An Integer Programming Approach to the Path Selection Problems

Deokseong Kim[a], Kyungsik Lee[b], Kyungchul Park[c], and Sungsoo Park[d*]

[a] *Mobile Communication Division, SEC*
416 Maetan-3dong, Yeongtong-gu, Suwon-si, Gyeonggi-do 442-600, Republic of Korea
[b] *School of Industrial & Management Engineering, Hankuk University of Foreign Studies*
89 Wangsan-ri, Mohyeon-myon, Yongin-si, Gyeonggi-do 449-791, Republic of Korea
[c] *Department of Systems Management Engineering, Sungkyunkwan University*
300 Cheoncheon-dong, Jangan-gu, Suwon-si, Gyeonggi-do 440-746, Republic of Korea
[d] *Department of Industrial Engineering, KAIST*
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea

May, 2007

Abstract

We consider two types of path selection problems defined on arc-capacitated networks. Given an arc-capacitated network and a set of selected ordered pairs of nodes (commodity) each of which has a demand quantity, the first problem is to select a subset of commodities and setup one path for each chosen commodity to maximize profit, while satisfying the arc capacity constraints. The second is to setup one path for each of the given commodities so that the total cost is minimized under the same capacity constraints as the first problem.

We present new integer programming formulations for the two problems whose linear programming relaxations provide stronger bounds than earlier formulations. Since each of these integer programs has an exponentially many variables, column generation procedures are proposed to solve the linear programming relaxations. To get an integer optimum solution, we devise a branch-and-price procedure which incorporates an effective branching strategy. Computational experiments show that our algorithm can yield optimal solutions within reasonably small computing time. We also perform computational experiments to make comparisons with other existing approaches. The results show that our algorithm outperforms other approaches, mainly due to stronger bounds and smaller sizes of the enumeration trees.

Keywords: multicommodity flows; path selection problems; integer programming; column generation; branch-and-price

*Corresponding author. E-mail address: sspark@kaist.ac.kr

1 Introduction

We consider two types of path selection problems arising from transportation, communication, and production. They are defined over an arc-capacitated directed or undirected network. We are given a set of ordered pairs of nodes (commodity) each of which has associated demand quantity, revenue, and unit flow cost on each arc. The first problem is to choose a subset of given commodities and find one simple path for each chosen commodity to maximize total profit, while satisfying arc capacity constraints. We call this problem PSC (Path selection problem for a Subset of Commodities). The second problem, what we call PAC (Path selection problem for All Commodities) hereafter, is to find one simple path for each of all the given commodities so that total flow cost is minimized under the same arc capacity constraints as PSC. The problems are known to be NP-hard (Parker and Ryan (1994), Barnhart et al. (2000)). A number of example applications of these problems include bandwidth packing problems (Cox et al. 1991) and package flow problems. Barnhart et al. (2000) illustrated the example problems as follows.

- "Bandwidth packing problems require that bandwidth be allocated in telecommunications networks to maximize total revenue. The demands, or calls, on the networks are the commodities and the objective is to route the calls from their origin to their destination."

- "Package flow problems, such as those arising in express package delivery operations, require that shipments, each with specific origin and destination, be routed over a transportation network. Each set of packages with a common origin-destination pair can be considered as a commodity and often must be assigned to a single network path to facilitate operations and ensure customer satisfaction."

Among these problems, Laguna and Glover (1993) and Anderson et al. (1993) developed a tabu search method to approximately solve the bandwidth packing problem, which is an application of PSC. In addition, Laguna and Glover (1993) proposed an integer programming (IP) formulation of the problem and solved some instances using a commercial optimization package. To solve the IP formulation, they generated a fixed number of paths per commodity, and used them to formulate the problem. According to their report, an IP approach using a commercial optimization package is not practical since it takes too much CPU time to solve the (restricted) IP formulation.

Parker and Ryan (1994) proposed a column generation approach to the bandwidth packing problem. They incorporated the column generation technique in the branch-and-bound procedure to solve the linear programming (LP) relaxation of their IP formulation which has exponentially many path variables. However, they did not devise any other methods to strengthen the bounds of the LP relaxation. The solution approach they used has been applied successfully to solve many large-scale mixed integer programming problems, for instance, see Savelsbergh (1997) and Vance (1998). For a comprehensive survey of this approach, refer to Vanderbeck and Wolsey (1996) and Barnhart et al. (1998), where they called the approach the branch-and-price approach.

Park et al. (1996) proposed another branch-and-price algorithm for the bandwidth packing problem. They used the same IP formulation as that of Parker and Ryan (1994), but they also incorporated a strong cutting plane (row) generation technique in addition to the column generation technique to strengthen the bounds of the LP relaxation. The results show that their algorithm outperforms the Parker and Ryan's approach in terms of computing time. Moreover, the performance is even comparable to Laguna and Glover's tabu search heuristic algorithm.

After several years later, Barnhart et al. (2000) used the same approach as that of Park et al. (1996) to the integer multi-commodity problem, which is the same problem as PAC. Their algorithm, which incorporated combined column and row generation technique, is very similar to Park et al. (1996) but the algorithmic details including the branching rule are different. They named the algorithm the branch-and-price-and-cut algorithm. Other applications of this approach can be found in Nemhauser and Park (1991) and Lee et al. (1996).

In this paper, we present new integer programming formulations for PSC and PAC, which give stronger LP bounds both theoretically and computationally than the path-based formulations which are given in the next section. Our formulations are based on the concept of a commodity pattern, which is presented later in detail in the next section. Since there are exponentially many commodity patterns, the proposed integer programs have an exponentially many variables. To get an optimal solution to the problems, we devise a branch-and-price algorithm which incorporates a combined column and row generation procedure together with an effective branching strategy.

We test the performance of the proposed algorithm on several sets of problem instances. The sizes of test problems are comparable to those used in Laguna and Glover (1993) and

Park et al. (1996). Usually the total computation time does not exceed a few minutes on a Pentium PC (866MHz). For comparison with the previous approach proposed by Park et al. (1996) and Barnhart et al. (2000), each of which also used a combined column and row generation procedure using the path-based formulations, we implemented both of their algorithms and test their performance on the same problem instances. The results show that our algorithm outperforms other algorithms, mainly due to the stronger bounds and smaller sizes of the enumeration trees.

This paper is organized as follows. In section 2, we present new formulations of PSC and PAC together with the path-based formulations. We also analyze the theoretical strength of the proposed formulations. A column generation procedure to solve the LP relaxation of the formulation is given in section 3. Section 4 presents the overview of the proposed algorithm with algorithmic details including a branching strategy. Computational test results are given in section 5. Finally, concluding remarks are given in section 6.

2 Formulations of the Problem

In this section, we first introduce integer programming formulations for PSC and PAC which were used in the previous works. Next, we present new integer programming formulations for those problems and discuss theoretical strength of the bounds provided by the linear programming relaxations of the formulations. The following are the notation and definitions used in the formulations.

V **:** set of nodes

E **:** set of arcs

K **:** set of commodities (ordered pairs of nodes)

$P(k)$ **:** set of (s,t)-paths, where s is the source and t is the destination of commodity k, where $k \in K$

$P(k;e)$ **:** set of paths in $P(k)$ that traverse arc e, where $e \in E, k \in K$

$E(p)$ **:** set of arcs contained in path p

c_e^k **:** unit flow cost of commodity k on arc e, where $k \in K, e \in E$

b_e **:** capacity of arc e, where $e \in E$

v_k **:** revenue of commodity k, where $k \in K$

r_k **:** demand quantity of commodity k, where $k \in K$

Using the above notation, PSC can be formulated as follows. For ease of later expositions, we call this formulation PSCP.

$$
\begin{aligned}
\text{(PSCP)} \quad \max & \sum_{k \in K} \sum_{p \in P(k)} w_p^k y_p^k \\
\text{s.t.} & \sum_{p \in P(k)} y_p^k \leq 1, \text{ for all } k \in K & (1) \\
& \sum_{k \in K} \sum_{p \in P(k;e)} r_k y_p^k \leq b_e, \text{ for all } e \in E & (2) \\
& y_p^k \geq 0, \text{integer, for all } p \in P(k), k \in K.
\end{aligned}
$$

where $w_p^k = v_k - r_k \sum_{e \in E(p)} c_e^k$.

Without loss of generality, we assume all data are integral. Note that w_p^k is the profit obtained if we select commodity k and assign it to path p. y_p^k is 1 if commodity k is chosen and assigned to path p; otherwise, it is 0. Constraints (1) ensure that at most one path can be selected for each commodity. Note that we do not need to require explicitly the y variables to be binary in the above formulation due to constraints (1). Constraints (2) mean that the total amount of demand quantities assigned to each arc is less than or equal to the capacity of the arc.

Since exactly one path should be selected for each commodity, the sense of constraints (1) should be changed to equality in the formulation for PAC. Also, in this case, the sum of revenues is constant, so that the objective function reduces to minimizing the total flow cost. For ease of later expositions, we call this formulation PACP. The formulations PSCP and PACP have been used in the previous works, such as Parker and Ryan (1994), Park et al. (1996), and Barnhart et al. (2000). Those previous researches are successful in that the proposed algorithms are much more efficient and can solve larger problem instances than the algorithms using the traditional arc-flow based formulations. To further improve the capability to solve PSC and PAC, we propose new formulations of the problems.

We first define the concept of a commodity pattern. A *commodity pattern* g for an arc e is represented by a set $K(g) \subseteq K$ such that $\sum_{k \in K(g)} r_k \leq b_e$. Let $G(e)$ be the set of all possible commodity patterns for arc $e \in E$, also let $G(e;k)$ be the subset of $G(e)$ which consists of

commodity patterns that include commodity $k \in K$. Now, we present a new formulation (MPS) for the problem PSC.

$$\text{(MPS)} \quad \max \sum_{k \in K} \sum_{p \in P(k)} w_p^k y_p^k$$

$$\text{s.t.} \sum_{p \in P(k)} y_p^k \leq 1, \text{ for all } k \in K \tag{3}$$

$$\sum_{g \in G(e)} z_e^g \leq 1, \text{ for all } e \in E \tag{4}$$

$$\sum_{p \in P(k;\ e)} y_p^k \leq \sum_{g \in G(e;\ k)} z_e^g, \text{ for all } e \in E, k \in K \tag{5}$$

$$y_p^k \geq 0, \text{integer, for all } p \in P(k), k \in K$$

$$z_e^g \geq 0, \text{integer, for all } g \in G(e), e \in E.$$

The decision variable z_e^g is equal to 1 if commodity pattern $g \in G(e)$ is selected, 0 otherwise. Constraints (4) ensure that at most one commodity pattern can be selected for each arc. Note that we do not need to require explicitly the z variables to be binary in the above formulation due to constraints (4). Constraints (5) mean that a path p for commodity k which traverses arc e can be set up only if a commodity pattern $g \in G(e)$ such that $k \in K(g)$ is selected for arc e. For the case of PAC, we only need to change the sense of constraints (3) to equalities and change the objective function to get the corresponding formulation, which we call MPA from now on.

Generally, there is an exponential number of paths and commodity patterns for an instance, that is MPS (MPA) has an exponential number of decision variables. It is thus impractical to enumerate all the possible paths and commodity patterns beforehand. However, the linear programming (LP) relaxations of MPS and MPA can be solved efficiently by using column generation technique of Gilmore and Gomory (1961). For the details of this technique, refer to Gilmore and Gomory (1961) and Barnhart et al. (1998). In the next section, we present a column generation procedure to solve the LP relaxations of MPS and MPA in detail.

Before closing this section, we are to investigate the strength of the bounds provided by the LP relaxations of the proposed formulations. In the following, we mainly discuss the analysis results for the case of MPS, however, the results are also valid for the case of MPA.

Let P and G be $\bigcup_{k\in K} P(k)$ and $\bigcup_{e\in E} G(e)$, respectively. Now, consider the following sets S_e and Q_e for each $e \in E$.

$$S_e = \{y \in Z_+^{|P|} : \sum_{k\in K} \sum_{p\in P(k;e)} r_k y_p^k \le b_e, \sum_{p\in P(k;e)} y_p^k \le 1, \text{for all } k \in K\},$$

$$Q_e = \{z \in R_+^{|G|}, y \in R_+^{|P|} : \sum_{g\in G(e)} z_e^g \le 1, \sum_{p\in P(k;e)} y_p^k \le \sum_{g\in G(e;k)} z_e^g, \text{for all } k \in K\}$$

Let $Proj_y(Q_e)$ be the projection of Q_e onto y-space, and let $conv(S_e)$ be the convex hull of S_e. Then, the following lemma holds.

Lemma 1. *$Proj_y(Q_e) = conv(S_e)$, for all $e \in E$.*

Proof. The projection of Q_e, for each $e \in E$ can be expressed as follows (see Theorem 4.10 in page 98, Nemhauser and Wolsey (1988)):

$$Proj_y(Q_e) = \{y \in R_+^{|P|} : \sum_{k\in K} \sum_{p\in P(k;e)} \pi_k^t y_p^k \le \pi_0^t, \text{for all } t \in T\},$$

where T is the set of indices of extreme rays of $\Pi_e = \{\pi \in R_+^{|K|+1} : \sum_{k\in K(g)} \pi_k \le \pi_0, \text{for all } g \in G(e)\}$. We first note that every point of $conv(S_e)$ is in $Proj_y(Q_e)$. Now we show that every facet-defining inequality of $conv(S_e)$ is one of the defining inequalities of $Proj_y(Q_e)$.

We first note that if an inequality $\sum_{k\in K} \sum_{p\in P(k)} \mu_p^k y_p^k \le \mu_0$ is valid to $conv(S_e)$, then $\mu_0 \ge 0$ and $\mu_p^k = 0$ for all $p \in P(k) \setminus P(k;e)$ and $k \in K$. Suppose that an inequality defines a facet of $conv(S_e)$. Then, there exist $|P|$ linearly independent points $\hat{y}(i), i = 1, \ldots, |P|$, in S_e that satisfy the above inequality at equality. Also it can be easily verified that if $\mu_p^k < 0$ for some $p \in P(k;e)$ or $\mu_p^k \ne \mu_q^k$ for some $p, q \in P(k;e)$, then the inequality is dominated by a valid inequality with $\mu_p^k \ge 0$ for all $p \in P(k;e)$ and $\mu_p^k = \mu_q^k$ for every pair of $p, q \in P(k;e)$. These facts imply that μ_p^k for all $p \in P(k)$ and $k \in K$ is a solution to the following system of linear equalities whose rank is $|K|$.

$$\sum_{k\in K} \sum_{p\in P(k)} \hat{y}_p^k(i)\mu_p^k = \mu_0, \text{ for all } i = 1, \ldots, |P|$$

$$\mu_p^k = 0, \text{ for all } p \in P(k) \setminus P(k;e)$$

$$\mu_p^k - \mu_q^k = 0, \text{ for all } p, q \in P(k;e) \text{ and } k \in K$$

In other words, $\mu_p^k = \mu_k$ for all $p \in P(k;e)$ and $k \in K$, where the nonnegative vector $\mu = (\mu_0, \mu_1, \ldots, \mu_{|K|})$ is a solution to the following system of linear equalities whose rank is

also $|K|$.

$$\sum_{k \in K(g_i)} \mu_k = \mu_0, \text{for all } g_i \in G(e), i = 1, \ldots, |P|,$$

where $K(g_i) = \{k \in K |\ \hat{y}_p^k(i) = 1 \text{ for some } p \in P(k; e)\}$, for all $\hat{y}(i), i = 1, \ldots, |P|$. Therefore, μ is an extreme ray of Π_e. □

Now, consider the following linear program (LP1).

$$\text{(LP1)} \quad \max \sum_{k \in K} \sum_{p \in P(k)} w_p^k y_p^k$$
$$\text{s.t. (1) and } y \in \bigcap_{e \in E} conv(S_e)$$

Proposition 1. *The LP relaxation of MPS (MPL) and LP1 have the same optimal value.*

Proof. If (y', z') is a solution to MPL then y' satisfies (1) and $y' \in \bigcap_{e \in E} Proj_y(Q_e)$, that is , y' is a solution to LP1 by lemma 1. Conversely, by lemma 1, for a given solution y' to LP1, there exists $z' \in R_+^{|G|}$ such that (y', z') is a solution to MPL. □

Note that every feasible solution to LP1 is also a feasible solution to the LP relaxation of PSCP, but the converse is not necessarily true. This fact and proposition 1 imply that the bound provided by MPL is at least as strong as the bound provided by the LP relaxation of PSCP. In fact, lifted cover inequalities for the knapsack polytope were used in Park et al. (1996) and Barnhart et al. (2000) to obtain tighter description of $conv(S_e)$.

Since complete description of the knapsack polytope is unknown, proposition 1 implies that the bound obtained by solving MPL is stronger than the bound obtained in Park et al. (1996) and Barnhart et al. (2000).

By the same reasoning as in the case of MPS, we can get the same results for MPA.

3 Column Generation Problems

In this section, we present column generation procedures to solve LP relaxations of the formulations MPS and MPA given in the previous section. We discuss the details for the case of MPS, however, the procedure can be applied to the case of MPA.

Let P and G be $\bigcup_{k \in K} P(k)$ and $\bigcup_{e \in E} G(e)$, respectively. Assume that $P' \subseteq P$ and $G' \subseteq G$ are given. Replacing P and G by P' and G' in MPL yields the restricted linear program MPL$'$, whose solutions are suboptimal to MPL. Then, we solve MPL$'$ by using the

simplex method (Nemhauser and Wolsey (1988)), a well-known method for solving linear programming problems, which yields an optimal solution to MPL$'$ together with an optimal dual solution. Then, using the optimal dual solution, we search for profitable columns (paths and/or commodity patterns) whose addition to MPL$'$ may result in an increase of the optimal objective value of MPL$'$. If there are no such columns, the solution at hand is an optimal solution to MPL. Otherwise, we add the columns to MPL$'$, and then repeat the above process.

The key step of the above process is to check whether there are profitable paths and/or commodity patterns. We can formalize this issue as other optimization problems, the so-called column generation sub-problems. Let α_k for all $k \in K$, β_e for all $e \in E$, and γ_e^k for all $k \in K$ and for all $e \in E$ be the dual variables associated with constraints (3), (4) and (5), respectively. Then the constraints in the dual of MPL are

$$\alpha_k + \sum_{e \in E(p)} \gamma_e^k \geq w_p^k, \text{ for all } p \in P(k), k \in K$$

$$\beta_e - \sum_{k \in K(g)} \gamma_e^k \geq 0, \text{ for all } g \in G(e), e \in E$$

$$\alpha_k, \beta_e, \gamma_e^k \geq 0, \text{ for all } k \in K, e \in E$$

Let $(\bar{\alpha}, \bar{\beta}, \bar{\gamma})$ be the optimal solution of the dual problem of MPL$'$. Then it is optimal to the dual of MPL if and only if

$$\bar{\alpha}_k + \sum_{e \in E(p)} \bar{\gamma}_e^k \geq w_p^k, \text{ for all } p \in P(k), k \in K$$

$$\bar{\beta}_e - \sum_{k \in K(g)} \bar{\gamma}_e^k \geq 0, \text{ for all } g \in G(e), e \in E$$

Therefore, we may write the optimality condition for MPL as follows:

$$\min\{ \sum_{e \in E(p)} \bar{\gamma}_e^k - w_p^k | p \in P(k)\} \geq -\bar{\alpha}_k, \text{ for all } k \in K \tag{6}$$

$$\max\{ \sum_{k \in K(g)} \bar{\gamma}_e^k | g \in G(e)\} \leq \bar{\beta}_e, \text{ for all } e \in E \tag{7}$$

Using condition (6) together with the fact that $w_p^k = v_k - r_k \sum_{e \in E(p)} c_e^k$, we can derive the first column generation problem (SP1(k)) to check if there is any profitable path for each commodity $k \in K$ as follows.

$$\text{(SP1}(k)\text{)} \quad \min \sum_{e \in E(p)} (\bar{\gamma}_e^k + r_k c_e)$$

$$\text{s.t. } p \in P(k)$$

For a given commodity $k \in K$, SP1(k) is the shortest path problem from node s to node t, where s is the source and t is the destination node of the commodity k, respectively. Since the arc weights are nonnegative, SP1(k) can be solved efficiently by Dijkstra's algorithm (Bondy and Murty (1976)). If the resulting length of the shortest path is less than $v_k - \bar{\alpha}_k$, the path can be added to MPL$'$. Otherwise, no column is added to MPL$'$ with respect to the commodity k.

For each arc $e \in E$, we can define the second column generation problem (SP2(e)) to check if there is any profitable commodity pattern by using condition (7) as follows.

$$\begin{aligned} (\text{SP2}(e)) \quad \max & \sum_{k \in K} \bar{\gamma}_e^k t_k \\ \text{s.t.} & \sum_{k \in K} r_k t_k \leq b_e \\ & t_k \in \{0, 1\}, \text{ for all } k \in K \end{aligned}$$

For each arc $e \in E$, the above SP2(e) is a well-known 0-1 knapsack problem, which is known to be NP-hard. However, the problem can be solved well by using the dynamic programming algorithm (Nemhauser and Wolsey (1988)). If the resulting optimal objective value of SP2(e), $e \in E$, is greater than $\bar{\beta}_e$, the commodity pattern corresponding to the obtained 0-1 solution can be added to MPL$'$. Otherwise, no column is added to MPL$'$ with respected to the arc e.

4 Overview of the Algorithm

4.1 Overview

In this section, we give a brief and overall description of our algorithm to solve PSC and PAC. Since our algorithm for PSC which uses the formulation MPS is the same as that for PAC except that the latter uses the formulation MPA, we only present the details for the case of PSC.

Our algorithm can be outlined as follows: We first construct an initial formulation of MPL$'$ (the restricted linear program explained in section 3) as described in section 4.2. Note that the number of constraints (5) becomes larger as $|K|$ and $|E|$ increase. To reduce the overhead of solving MPL$'$, we initially include a small number of constraints (5) and thereafter add the constraints as cutting planes when they are needed.

After we get the initial formulation of MPL′, we solve MPL′ using the column generation procedure presented in section 4.3. The resulting solution may violate some of the constraints (5). We find those violated constraints and add them to MPL′ as described in section 4.3. If no columns and violated constraints are found, we are done with an optimal solution to MPL (the LP relaxation of MPS). Otherwise, we go thorough the same process as we do after we get the initial formulation of MPL′.

Then we check if the obtained solution to MPL is integral. If it is the case, we are finished with an optimal integral solution to MPS. Otherwise, we go into the branch-and-price phase with the branching rule described in section 4.4. The procedure incorporates the column and cut generation procedure described in section 4.3 within the branch-and-bound scheme. The procedure yields an optimal solution to MPS. We used the best bound rule (Nemhauser and Wolsey (1988)) as the node selection rule in the branch-and-bound tree.

4.2 Initial Formulation of MPL′

To construct the initial formulation of MPL′, we first construct the set $P'(k) \subseteq P(k)$ for each $k \in K$ with one path corresponding to the shortest path for commodity k. We also initialize $G'(e) \subseteq G(e)$, as $G'(e) = \emptyset$ for each $e \in E$. Let $E' \subseteq E$ be the set of arcs each of which is used by at least one path in $\bigcup_{k \in K} P'(k)$. Then the initial formulation is as follows.

$$
\begin{aligned}
(\text{MPL}') \quad \max & \sum_{k \in K} \sum_{p \in P'(k)} w_p^k y_p^k - M \sum_{k \in K} a_k \\
\text{s.t. } & a_k + \sum_{p \in P'(k)} y_p^k \leq 1, \text{ for all } k \in K && (8) \\
& \sum_{g \in G'(e)} z_e^g \leq 1, \text{ for all } e \in E && (9) \\
& \sum_{p \in P'(k;\ e)} y_p^k \leq \sum_{g \in G'(e;\ k)} z_e^g, \text{ for all } e \in E', k \in K && (10) \\
& y_p^k \geq 0, \text{ for all } p \in P'(k), k \in K \\
& z_e^g \geq 0, \text{ for all } g \in G'(e), e \in E \\
& a_k \geq 0, \text{ for all } k \in K.
\end{aligned}
$$

Note that we add one artificial variable a_k for each $k \in K$ with a sufficiently large positive penalty M to ensure the feasibility of the initial linear program. Also note that those artificial variables are needed even if we have a feasible basis to guarantee the feasibility after branching in the branch-and-price phase.

4.3 Column and Cut Generation Procedure

After we get the initial formulation of MPL$'$ as presented in section 4.2, we apply the follwing column and cut generation procedure to get an optimal solution to MPL. We also use the procedure to solve MPL at each node of the enumeration tree in the branch-and-price phase.

Stage 1 : LP solution

Step 1 : Solve the current formulation of MPL$'$.

Step 2 : Let $(\bar{y}, \bar{z})$ be the obtained optimal solution and let $\bar{\alpha}_k$ for all $k \in K$, $\bar{\beta}_e$ for all $e \in E$, and $\bar{\gamma}_e^k$ for all $k \in K$ and $e \in E'$ be the corresponding dual solution associated with constraints (8), (9) and (10). Go to Stage 2.

Stage 2 : Path generation

Step 1 : For each $k \in K$, solve SP1(k) and add the obtained path to $P'(k)$ if the length of the path is less than $(v_k - \bar{\alpha}_k)$ - i.e. condition (6) is violated.

Step 2 : If a new path is added for some $k \in K$ in step 1, go to Stage 1. Otherwise, go to Stage 3.

Stage 3 : Commodity pattern generation

Step 1 : For each $e \in E$, solve SP2(e) and add the obtained commodity pattern to $G'(e)$ if the resulting objective value of SP2(e) is greater than $\bar{\beta}_e$ - i.e. condition (7) is violated.

Step 2 : If a new commodity pattern is added for some $e \in E$ in step 1, go to Stage 1. Otherwise, go to Stage 4.

Stage 4 : Cut generation

Step 1 : Recall that for each arc e in $E' \subseteq E$, corresponding constraints (5) are included in the current formulation. For each $e \in E \setminus E'$, add corresponding constraints (5) for all $k \in K$ and set $E' = E' \cup \{e\}$ if one of those inequalities is violated by the current solution $(\bar{y}, \bar{z})$.

Step 2 : If for some $e \in E$, constraints (5) for all $k \in K$ are added in step 1, go to Stage 1. Otherwise, the current solution is optimal to MPL. Stop.

4.4 Branching Strategy

One of the keys to devising a branch-and-price procedure is identifying a branching rule that eliminates a fractional solution to MPL at hand without making column generation problems intractable after branching. In the following, we present a branching rule that does not change the characteristics of the column generation problems given in section 3 after branching. We first note that, for a given extreme point solution $(\bar{y}, \bar{z})$ to MPL such that $\bar{y}$ is integral, either $\bar{z}$ is integral or there exists a solution $(\bar{y}, \hat{z})$ with the same objective value such that $\hat{z}$ is integral. This result directly follows from the fact that if $\bar{z}$ is not integral, we can construct $\hat{z}$ such that for each $e \in E$, $\hat{z}_e^g = 1$ if and only if $K(g) = \{k \in K |\ \bar{y}_p^k = 1 \text{ for some } p \in P(k; e)\}$. This implies that we only need to check the integrality of path variables.

Now, we present our branching rule which consists of two stages. Recall that PSC and PAC can be defined on both directed and undirected networks. The following is the branching rule for PSC and PAC defined on undirected networks. We mention that stage 1 is not necessary for the case of PAC. Stage 2 is a modified version of the branching rule proposed by Barnhart et al. (2000) for the case of PAC defined on directed networks. For the problems defined on directed networks, stage 2 can be replaced by Barnhart et al. (2000)'s rule.

Branching rule for PSC and PAC on on undirected network

Let (y, z) be a fractional solution to MPL.

Stage 1 : Commodity branching

Step 1 : If there is no such $k \in K$ such that $\sum_{p \in P(k)} y_p^k$ is fractional, go to Stage 2. Otherwise, select a commodity $k \in K$ with the lowest index such that $k = argmin_{k \in K}\{|0.5 - \sum_{p \in P(k)} y_p^k|\}$.

Step 2 : Create 2 nodes in the enumeration tree, one with $\sum_{p \in P(k)} y_p^k = 0$ and the other with $\sum_{p \in P(k)} y_p^k = 1$. To satisfy the condition at each node, we change the senses and the right-hand-side constants of constraints (8) accordingly. Stop.

Stage 2 : Path branching

Step 1 : If there is no such $k \in K$ such that $0 < y_p^k < 1$ for all $p \in P(k)$, stop.

Otherwise, select $k \in K$ with the lowest index such that $0 < y_p^k < 1$ for some $p \in P(k)$.

Step 2 : Let $Q(k) = \{p \in P(k) | y_p^k > 0\}$ and let $(s, v_1, v_2, \cdots v_{i-1}, v_i)$ be a longest path such that the path is contained in every path in $Q(k)$, where s is the source of commodity k. Let $A(v_i)$ be the set of arcs incident to node v_i other than arc $\varepsilon = (v_{i-1}, v_i)$. Identify the arcs $e1 \in A(v_i)$ and $e2 \in A(v_i)$ such that the $\sum_{p \in P(k;\ e1)} y_p^k \geq \sum_{p \in P(k;\ e2)} y_p^k \geq max_{e \in A(v_i) \setminus \{e1,\ e2\}} \sum_{p \in P(k;e)} y_p^k$.

Step 3 : Construct the set $A(v_i, e1)$ containing $e1$ and the set $A(v_i, e2) = A(v_i) \setminus A(v_i, e1)$ with arc $e2$ according to the Step 4 of Barnhart et al. (2000)'s branching rule - i.e. let the size of the two sets roughly equal. Create two nodes in the enumeration tree, one with the condition that the arcs in $A(v_i, e1)$ are forbidden for commodity k, and the other with the condition that the arcs in $A(v_i, e2)$ are forbidden for commodity k. To satisfy the conditions imposed by branching, for each path variable that is already generated, we first set the upper bound of the variable to 0 if the corresponding path passes through an arc in $A(v_i, e1)$ at the first node. We perform a similar bound setting at the second node. Furthermore, at the first node, we perform the column generation procedure for SP1(k) over the network obtained by removing arcs $e \in A(v_i, e1)$ from the given network. The same scheme is applied to the second node.

5 Computational Results

We performed computational experiments in two cases. In the first case, we applied our algorithm to 10 randomly generated problem instances of PSC defined on each of the 4 randomly generated undirected networks. We also tested our algorithm on 10 randomly generated problem instances of PAC defined on each of the 4 randomly generated undirected networks. For computational tests, we used CPLEX callable library (version 7.0) as an LP solver. The tests were performed on a Pentium PC (866 MHz CPU).

5.1 Characteristics of Problem Instances

To generate problem instances of PSC, we first generate 2 networks with 30 nodes and 2 networks with 35 nodes. These networks are generated based on the construction method

given in Park et al. (1996) to reflect the characteristics of real-world networks. Given the number of nodes, we randomly generated the position of each node of each network so that they are uniformly distributed on a 50 by 50 Euclidean square. For each of the 4 networks, the number of arcs is randomly chosen in the range of $[1.5|V|, 3.5|V|]$. Given the number of arcs of a network, each arc is randomly chosen so that nearer nodes are most likely to be connected than distant nodes. The length of each arc is set to the Euclidean distance between the two nodes of the arc. For the instances of PAC, we used the same generation rule as above.

For each generated network, we generated 10 problem instances. The capacity of each arc is randomly chosen in the range of [10, 50] in the case of PSC. In the case of PAC arc capacities are chosen in the range of [20, 70]. For the instances of PSC, the number of commodities is set to a randomly chosen number in the range of $[3|V|, 4|V|]$. For the case of PAC, the number of commodities is randomly determined in the range of $[|V|, 3|V|]$. The demand quantity for each commodity is generated in the range of [1, 20] and the revenue of each commodity is set to $10 \times R$, where R is a randomly generated integer in the range of [10, 100]. The unit flow cost of commodity $k \in K$ on arc $e \in E$ is set to the length of the arc.

5.2 Summary of Test Results

Computational results for the instances of PSC are summarized in Tables 1 - 4. Tables 5 - 8 show the results for the case of PAC.

>> Insert Table 1-8 here <<

In those tables, the headings $|K|$, #COLP, #COLC, #TCUT, and #LP refer to the number of commodities, the total number of generated path variables, the total number of generated commodity pattern variables, the total number of constraints (5) added, and the number of calls to LP solver during the execution of our algorithm, respectively. Let LP be the optimal objective value of the LP relaxation of MPS(MPA) at the root node of the enumeration tree and let OPT be that of an optimal integral solution to MPS(MPA). GAP(%) is defined as follows:

$$\text{GAP}(\%) = |\text{LP} - \text{OPT}|/\text{OPT} \times 100$$

The heading #BB refers to the number of nodes of the enumeration tree generated in the branch-and-price procedure. Finally, TIME refers to the accumulated CPU time in seconds needed to solve an instance.

From the results, we can see that the number of generated path variables does not exceed 5 times the number of commodities ($|K|$) in average. The number of commodity pattern variables also does not exceed twice the number of generated path variables in average. GAP is very small and usually within 0.2% in average, so that the number of generated nodes in the branch-and-price procedure is small and CPU time needed to solve an instance does not exceed a few minutes.

5.3 Comparison with Other Approaches

We made comparisons with other approaches. For the case of PSC, we compare the performance of our algorithm with the performance of the algorithm presented by Park et al. (1996). We implemented their algorithm and applied it to the test instances of PSC defined on 2 networks given in section 5.2. Tables 9 - 10 show the results. In those tables, KLP refers to our algorithm and PKP1996 refers to the algorithm proposed by Park et al. (1996). From the results, we can see that the average GAP reduction by our algorithm is about 50%. Our algorithm also dramatically reduces the number of nodes in the branch-and-price procedure and CPU time needed to get optimal solutions.

We also compare the performance of our algorithm with that of Barnhart et al. (2000) in the case of PAC using the test instances of PAC defined on 2 networks given in section 5.2. Tables 11 - 12 show the results. In those tables, BHV2000 refer to the algorithm proposed by Barnhart et al. (2000). From the results, we can see that our algorithm outperforms BHV2000 as in the first comparison with PKP1996. We can also see that our algorithm consistently shows more reliable performance than BHV2000 on some difficult instances.

>> Insert Table 9-12 here <<

6 Concluding Remarks

In this paper, we proposed new integer programming formulations for PSC and PAC which incorporate commodity pattern variables in addition to path variables. Those formulations give stronger LP bounds than the previous models.

Though the number of variables in the formulations is huge, we devised efficient algorithms to solve them using combined column and row generation technique. The computational results show that our algorithm can solve the problems to optimality within a short time and outperforms existing algorithms proposed so far.

We expect that the approach proposed in this paper can be applied to other potential applications including routing problems in telecommunication networks and logistics networks.

Acknowledgements

"This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD)" (KRF-2004-041-D00784).

References

Branch-and-Price-Cut To Solve Origin-Destination Integer Multicommodity Flow Problems. *Operations Research* **48** 318–326.

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

Bondy, J. A., U. S. R. Murty. 1976. Graph Theory with Applications. Elsevier, New York.

Cox, L.A., I. Davis, Y. Qie. 1991. Dynamic Anticipatory Routing in Circuit-Switched Telecommunications Networks, in L. Davis, eds. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

Gilmore, P. C., R. E. Gomory. 1961. A Linear Programming Approach to the Cutting-stock Problem. *Operations Research* **9** 849–859.

Laguna, M., F. Glover. 1993. Bandwidth Packing: A Tabu Search Approach. *Management Science* **39** 492–500.

Lee, K, K. Park, S. Park. 1996. Design of Capacitated Networks with Tree Configurations. *Telecommunication Systems* **6** 1–19.

Nemhauser, G. L., S. Park. 1991. A Polyhedral Approach to Edge Coloring. *Operations Research Letters* **10** 315–322.

Nemhauser, G. L., L. A. Wolsey. 1998. Integer and Combinatorial Optimization. John Wiley & Sons, New York.

Parker, M., J. Ryan. 1994. A Column Generation Algorithm for Bandwidth Packing. *Telecommunication Systems* **2** 185–196.

Park, K., S. Kang, S. Park. 1996. An integer programing approach to the bandwidth packing problem. *Management Science* **42** 1277–1291.

Savelsbergh, M. 1997. A Branch-and-Price algorithm for the generalized assignment problem. *Operations Research* **45** 831–841.

Vance, P. H. 1998. Branch and Price algorithms for the one-dimensional cutting stock problem. *Computational Optimization and Applications* **9** 211–228

Vanderbeck, F., L. A. Wolsey. 1996. An exact algorithm for IP column generation. *Operations Research Letters* **19** 151–159.

Table 1: Test results for PSC on Network 1 ($|N| = 30, |E| = 50$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	100	497	1723	3900	1225	0.45	162	88.23
2	102	339	721	3774	348	0.15	10	14.06
3	94	234	366	3008	186	0.05	14	4.91
4	84	161	199	2688	93	0.08	2	1.65
5	93	210	401	2790	330	0.21	34	6.98
6	99	207	254	3168	122	0.09	6	2.93
7	103	184	140	3193	42	0.00	0	1.01
8	84	118	157	1932	82	0.10	4	1.08
9	77	176	372	2387	288	0.72	42	5.18
10	79	185	221	3002	58	0.13	2	1.33
Avg	91.5	231.1	455.4	2984.2	277.4	0.20	27.6	12.74
Max	103	497	1723	3900	1225	0.72	162	88.23
Min	77	118	140	1932	42	0.00	0	1.01

Table 2: Test results for PSC on Network 2 ($|N| = 30, |E| = 70$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	97	625	1179	5141	551	0.19	54	54.07
2	100	287	286	4100	98	0.00	0	3.28
3	100	293	751	4000	546	0.43	82	17.45
4	84	168	130	2688	49	0.00	0	0.94
5	97	490	1281	4462	808	0.34	64	43
6	91	307	340	3276	127	0.22	2	4.13
7	100	464	749	5300	531	0.19	54	28.55
8	100	521	1550	4200	1085	0.40	86	71.47
9	95	247	261	3325	78	0.00	0	2.4
10	96	368	542	4704	178	0.05	6	9.26
Avg	96	377	706.9	4119.6	405.1	0.18	34.8	23.46
Max	100	625	1550	5300	1085	0.43	86	71.47
Min	84	168	130	2688	49	0.00	0	0.94

Table 3: Test results for PSC on Network 3 ($|N| = 35, |E| = 85$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	112	479	767	6608	276	0.11	22	20.6
2	120	619	1631	8040	1241	0.28	222	128.84
3	104	358	478	4992	168	0.21	4	8.2
4	111	463	778	6438	197	0.04	8	19.43
5	113	469	694	6554	296	0.09	22	19.48
6	92	204	139	3404	93	0.04	12	2.21
7	101	301	380	4141	318	0.11	46	10.39
8	101	318	309	4343	88	0.00	0	3.5
9	113	746	1324	6667	466	0.32	42	70.24
10	98	401	550	5782	228	0.25	20	11.77
Avg	106.5	435.8	705	5696.9	337.1	0.14	39.8	29.47
Max	120	746	1631	8040	1241	0.32	222	128.84
Min	92	204	139	3404	88	0.00	0	2.21

Table 4: Test results for PSC on Network 4 ($|N| = 35, |E| = 106$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	122	414	620	6588	275	0.12	24	17.29
2	117	561	591	7371	124	0.00	2	14.6
3	99	227	169	4554	54	0.00	0	1.83
4	120	398	495	6840	291	0.08	46	16.33
5	102	347	295	5610	87	0.00	0	4.11
6	125	406	522	7375	144	0.01	8	9.49
7	114	386	323	6270	90	0.00	2	5.31
8	96	308	325	4032	138	0.02	4	5.14
9	114	329	401	6270	123	0.05	4	6.66
10	119	345	416	5950	186	0.05	12	9.12
Avg	112.8	372.1	415.7	6086	151.2	0.03	10.2	8.99
Max	125	561	620	7375	291	0.12	46	17.29
Min	96	227	169	4032	54	0.00	0	1.83

Table 5: Test results for PAC on Network 5 ($|N| = 30, |E| = 78$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	41	279	542	1353	365	0.70	50	5.53
2	48	227	268	1536	88	0.00	2	1.21
3	37	121	136	703	71	0.00	0	0.45
4	49	128	140	980	124	0.25	8	1
5	43	240	286	1462	95	0.00	0	1.38
6	52	461	698	2756	253	0.10	8	11.24
7	34	75	44	340	31	0.00	0	0.13
8	41	212	310	1066	181	0.08	16	1.84
9	44	145	145	748	106	0.10	10	0.8
10	41	279	542	1353	365	0.70	50	5.32
Avg	43	216.7	311.1	1229.7	167.9	0.19	14.4	2.89
Max	52	461	698	2756	365	0.70	50	11.24
Min	34	75	44	340	31	0.00	0	0.13

Table 6: Test results for PAC on Network 6 ($|N| = 30, |E| = 92$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	56	226	343	2016	201	0.27	26	3.42
2	68	292	442	2992	142	0.15	8	4.05
3	63	260	239	2394	77	0.00	0	1.35
4	60	258	303	2340	121	0.10	10	2.2
5	55	113	70	1045	31	0.00	0	0.26
6	65	750	2067	3380	1321	0.34	206	108.78
7	60	504	842	4200	362	0.29	42	21.79
8	60	260	360	1920	117	0.03	2	2.56
9	48	111	52	768	29	0.00	0	0.2
10	66	553	1599	3630	1735	0.48	450	111.34
Avg	60.1	332.7	631.7	2468.5	413.6	0.17	74.4	25.595
Max	68	750	2067	4200	1735	0.48	450	111.34
Min	48	111	52	768	29	0.00	0	0.2

Table 7: Test results for PAC on Network 7 ($|N| = 35, |E| = 89$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	65	1072	3624	4680	2240	0.56	320	369.6
2	48	276	393	1968	208	0.09	26	4.1
3	67	273	189	2010	61	0.00	0	1.1
4	61	434	610	2928	212	0.12	8	8.57
5	62	554	1189	3534	726	0.36	102	39.45
6	59	275	364	2360	129	0.04	4	2.98
7	70	951	2532	5180	1260	0.41	106	184.5
8	49	173	230	1372	86	0.00	0	1.11
9	65	311	439	2990	100	0.00	0	4.21
10	65	1072	3624	4680	2240	0.56	320	369.71
Avg	61.1	539.1	1319.4	3170.2	726.2	0.21	88.6	98.533
Max	70	1072	3624	5180	2240	0.56	320	369.71
Min	48	173	189	1372	61	0.00	0	1.1

Table 8: Test results for PAC on Network 8 ($|N| = 35, |E| = 106$)

No	$\|K\|$	#COLP	#COLC	#TCUT	#LP	GAP(%)	#BB	TIME
1	73	693	853	4672	196	0.01	2	31.94
2	75	802	1704	5625	657	0.26	56	102.13
3	58	96	51	812	33	0.00	0	0.31
4	59	234	245	1770	103	0.00	4	1.69
5	70	559	811	3990	303	0.10	16	20.8
6	55	121	64	880	29	0.00	0	0.26
7	65	153	94	1365	34	0.00	0	0.4
8	62	253	283	2294	101	0.03	4	1.95
9	55	155	142	1155	51	0.00	0	0.53
10	58	298	542	2436	348	0.18	36	7.86
Avg	63	336.4	478.9	2499.9	185.5	0.06	11.8	16.787
Max	75	802	1704	5625	657	0.26	56	102.13
Min	55	96	51	812	29	0.00	0	0.26

Table 9: Comparison for PSC problem on Network 1 ($|N| = 30, |E| = 50$)

No	$\|K\|$	KLP			PKP1996		
		GAP(%)	#BB	TIME	GAP(%)	#BB	TIME
1	100	0.45	218	106.03	1.02	14944	*3600
2	102	0.15	10	14.06	0.18	50	1.86
3	94	0.05	14	4.91	0.18	280	6.01
4	84	0.08	2	1.65	0.08	2	0.2
5	93	0.21	34	6.98	0.38	584	12.22
6	99	0.09	6	2.93	0.28	56	1.89
7	103	0.00	0	1.01	0.00	0	0.14
8	84	0.10	4	1.08	0.70	84	1.55
9	77	0.72	42	5.18	0.78	372	6.7
10	79	0.13	2	1.33	0.60	404	7.41
Avg	91.5	0.20	33.2	14.516	0.42	1677.6	363.798
Max	103	0.72	218	106.03	1.02	14944	3600
Min	77	0.00	0	1.01	0.00	0	0.14

* : Unsolved within one hour

(GAP is calculated using an optimal solution obtained by KLP)

Table 10: Comparison for PSC problem on Network 3 ($|N| = 35, |E| = 85$)

No	$\|K\|$	KLP			PKP1996		
		GAP(%)	#BB	TIME	GAP(%)	#BB	TIME
1	112	0.11	22	20.6	0.39	4150	822.1
2	120	0.28	222	128.8	0.65	19140	*3600
3	104	0.21	4	8.2	0.37	3902	396.1
4	111	0.04	8	19.4	0.17	6020	1753.0
5	113	0.09	22	19.5	0.23	1920	164.7
6	92	0.04	12	2.2	0.09	6	0.4
7	101	0.11	46	10.4	0.18	2776	81.0
8	101	0.00	0	3.5	0.51	1010	75.6
9	113	0.32	42	70.2	0.51	16380	*3600
10	98	0.25	20	11.8	0.39	276	22.0
Avg	106.5	0.14	39.8	29.5	0.35	5558	1051.49
Max	120	0.32	222	128.8	0.65	19140	3600
Min	92	0.00	0	2.2	0.09	6	0.4

* : Unsolved within one hour

(GAP is calculated using an optimal solution obtained by KLP)

Table 11: Comparison for PAC problem on Network 6 ($|N| = 30, |E| = 92$)

No	$\|K\|$	KLP			BHV2000		
		GAP(%)	#BB	TIME	GAP(%)	#BB	TIME
1	56	0.27	26	3.4	0.39	250	6.6
2	68	0.15	8	4.1	0.31	38	1.1
3	63	0.00	0	1.4	0.05	10	0.4
4	60	0.10	10	2.2	0.30	66	1.9
5	55	0.00	0	0.3	0.03	2	0.1
6	65	0.34	206	108.8	0.94	22424	*3600
7	60	0.29	42	21.8	0.61	1880	88.5
8	60	0.03	2	2.6	0.11	14	0.6
9	48	0.00	0	0.2	0.01	2	0.1
10	66	0.48	450	111.3	0.80	31914	*3600
Avg	60.1	0.17	74.4	25.6	0.35	5660	729.93
Max	68	0.48	450	111.3	0.94	31914	3600
Min	48	0.00	0	0.2	0.01	2	0.1

* : Unsolved within one hour

(GAP is calculated using an optimal solution obtained by KLP)

Table 12: Comparison for PAC problem on Network 7 ($|N| = 35, |E| = 89$)

No	$\|K\|$	KLP			BHV2000		
		GAP(%)	#BB	TIME	GAP(%)	#BB	TIME
1	65	0.56	320	369.6	0.89	11360	1752.0
2	48	0.09	26	4.1	0.36	176	3.0
3	67	0.00	0	1.1	0.00	2	0.3
4	61	0.12	8	8.6	1.47	426	18.0
5	62	0.36	102	39.5	0.80	1128	39.7
6	59	0.04	4	3.0	0.06	14	0.4
7	70	0.41	106	184.5	0.85	704	100.2
8	49	0.00	0	1.1	0.16	20	0.6
9	65	0.00	0	4.2	0.41	132	4.7
10	65	0.56	320	369.7	0.89	11360	1773.5
Avg	61.1	0.21	88.6	98.5	0.59	2532.2	369.2
Max	70	0.56	320	369.7	1.47	11360	1773.5
Min	48	0.00	0	1.1	0.00	2	0.3