한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003 년 5 월 16 일-17 일 한동대학교(포항)

대역폭 분할 문제를 위한 Branch-and-Price 알고리듬

김덕성[a], 이경식[b], 박성수[a]
[a] 한국과학기술원 산업공학과
[b]한국외국어대학교 산업정보시스템공학부

A Branch-and-Price Algorithm for the Bandwidth Packing Problem

Deokseong Kim[a], Kyungsik Lee[b], Sungsoo Park[a]
[a]Dept. of Industrial Engineering, KAIST, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
[b]School of Industrial Information & Systems Engineering, Hankuk University of Foreign Studies, 89 Wangsan-ri, Mohyeon-myon, Yongin-si, Kyongki-do 449-791, Republic of Korea

Abstract

We consider the bandwidth packing problem arising from telecommunication networks. The problem is to determine the set of calls to be routed and an assignment of them to the paths in an arc capacitated network. The objective is to maximize profit.

We formulate the problem as an integer programming and propose an algorithm to solve it. Column generation technique to solve the linear programming relaxation is proposed with two types of columns. In addition, to obtain an optimum integer solution, we consider a new branching strategy. Computational experiments show that the algorithm gives optimal solutions within reasonably small time limits.

1. Introduction

We consider the bandwidth packing problem (BWP), which was first introduced by Cox et al. (1991). This problem occurs in the area of telecommunications. It is defined in terms of an arc-capacitated undirected network with a unit flow cost on each arc. When we are given a set of calls with their revenues and static bandwidth requirements, we need to select a subset of calls to be routed and assign them to node-simple paths with the objective of maximizing profit (Laguna and Glover 1993).

To solve this problem, several researches have been performed. Laguna and Glover (9993) and Anderson et al. (1993) developed a tabu search method to solve the problem. In addition, Laguna and Glover (1993) proposed an integer programming (IP) formulation of the problem and solved some instances using a commercial optimization package. According to their report, 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 delayed column generation algorithm to solve the problem without any use of cutting plane. To obtain an integer solution, they used a branch-and-price technique with a branching strategy different from us. They reported that their algorithm could find near optimal solutions within small time bound.

Park et al (1996) proposed an integer programming approach to solve the problem. They used the delayed column generation technique and strong cutting plane (row) generation technique together to strengthen the LP relaxation of the problem. To obtain an integer optimal solution, they used branch-and-bound and branch-and-cut technique with anther branching strategy different from us.

In this study, we use the delayed column generation technique to solve the linear programming relaxation of the problem as above, but with two different kinds of column types. This approach strengthens the LP relaxation bound at least as tight as strong cutting plane approach. When the approach fails to obtain an integer solution, we use the branch-and-price approach as similar to above approaches. During the branching stage, we also apply the column generation approach. In addition, we propose another branching strategy different from the approaches of the above researches.

We test the procedure on several problem instances. The sizes of test problem are comparable to those used in Laguna and Glover (1993) and Park et al (1996). The results show that our approach performs very well to find the optimal solutions within small time limits.

The paper is structured as follows. Section 2 presents the formulation of the problem. In section 3, we present the column generation procedure. Section 4 describes the overview of our algorithm. Computational results are shown in section 5. Finally, conclusions are given in Section 6.

2. Formulation

In this section, we present the formulation of BWP. Given an arc-capacitated network $G=(V,E)$, the set of calls, the bandwidth requirement for each call, and the revenue associated to each call, BWP finds the set of calls and an assignment of the selected calls to the paths in the network to maximize profit while satisfying the bandwidth capacity restrictions on each arc.

First, we give some notations to be used in the formulation of the problem.

Notation

V : set of nodes,
E : set of arcs,
K : set of candidate calls,
$P(k)$: set of (s, t)-paths, where s is the source and t is the destination of call k, where $k \in K$,
$P(k;e)$: set of paths in $P(k)$ that pass through arcs e, where $e \in E$ and $k \in K$,
$E(p)$: set of arcs of which path p consists,
c_e : cost of the unit flow on arc e, for all $e \in E$,
b_e : capacity of arc e, where $e \in E$,
v_k : revenue of call k, where $k \in K$,
r_k : bandwidth requirement of call k, where $k \in K$,

Additionally, we define a concept of a *call-set pattern*. A call-set pattern is a grouping configuration of calls that paths though an arc e and satisfies bandwidth capacity restriction on the arc e. Let $K(g)$ be the set of calls of call-pattern g. Then g consists of the set $K(g)$ of calls that satisfy $\sum_{k \in K(g)} r_k \le b_e$ for arc e, where $e \in E$. Let $G(e)$ be the set of call-set patterns for arc $e \in E$ and $G(e;k)$ be the set of call-set patterns in $G(e)$ that contain call k. Then, we can formulate BWP using above notations as follows:

(MP)

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

$$\text{s.t.} \quad \sum_{p \in P(k)} y_p^k \le 1 \qquad \forall k \in K \qquad (1)$$

$$\sum_{g \in G(e)} z_e^g \le 1, \qquad \forall e \in E \qquad (2)$$

$$\sum_{p \in P(k,e)} y_p^k \le \sum_{g \in G(e,k)} z_e^g, \forall e \in E, \forall k \in K \quad (3)$$

$$z_e^g \in \{0, 1\}, \forall g \in G(e), \forall e \in E,$$

$$y_p^k \in \{0,1\}, \forall p \in P(k), k \in K.$$

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

Without of loss of generality, we assume all data are integral. Note that w_p^k is the profit obtained if we select call k and assign it to path p. The decision variable y_p^k is 1 if call k is chosen and assigned to path p, 0 otherwise. The decision variable z_e^g is 1, if call-set pattern g is selected on arc e, 0 otherwise. Constraints (1) ensure that at most one path can be selected. Constraints (2) ensure that at most one call-set pattern can be selected for each arc. Constraints (3) mean that call k can path through arc e of path p if call-set pattern g, containing call k, is selected for arc e.

The bandwidth packing problem is NP-hard. This can be seen by reducing a 0-1 knapsack problem to a bandwidth packing problem. For more details of reducing, refer Parker and Ryan (1994).

Generally, there are an exponential number of paths and an exponential number of call-set patterns for MP. It is thus impractical to enumerate all the possible paths and call-set pattern to solve LP relaxation of MP with all the decision variables at hand. However, LP relaxation of MP can be solved efficiently by using the column generation technique of Gilmore and Gomory (1961). For more details of this method, refer to Gilmore and Gomory (1961) and Barnhart et al (1998).

3. Column Generation Problems

In this section, we give an explanation of column generation problems and the corresponding algorithms to solve the problems. Let MPL be the LP relaxation of MP.

Let U be the union of all possible paths ($P = \bigcup_{k \in K} P(k)$) and call-set patterns ($G = \bigcup_{e \in E} G(e)$). We assume that a subset U' of the set U is given. Replacing U by U' in MPL yields the restricted linear programming MPL′ whose solutions are suboptimal to MPL. Then, using the optimal dual solution returned by the simplex method in the current MPL′, we search for profitable columns 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 column to MPL′, and then repeat the above process.

Now, we mention how to find the profitable columns. Given a feasible basis to MPL, we need to generate columns to enter the basis. Let $(\bar{\alpha}, \bar{\beta}, \bar{\gamma})$ be the optimal solution of the dual problem of MPL′. Then, we may write the optimality condition for MPL as follows:

$$\min\{\sum_{e\in E(p)}\overline{\gamma}_e^k - w_p^k \mid p\in P\} \geq -\overline{\alpha}_k, \quad (4)$$

$$\max\{\sum_{k\in K(g)}\overline{\gamma}_k^e \mid g\in G\} \leq \overline{\beta}_e. \quad (5)$$

Using condition (4), we can derive the first column generation problem (SP1) associated to call k. The problem finds the shortest path from node s to node t, where s is the source and t is the destination of call k, respectively. Since the arc weights are nonnegative, SP1 can be solved efficiently by Dijkstra's algorithm (Bondy and Murty 1976). If the resulting length of the shortest path is less than $-\overline{\alpha}_k$, the path can be added to the current formulation. Otherwise, no column is generated with respect to call k.

Using condition (5), we can derive the second column generation problem (SP2) associated to arc e. The problem is a 0-1 knapsack problem for the arc e. The problem can be solved using dynamic programming (Nemhauser and Wolsey 1988) with time complexity $O(nb)$ where n is the number of variables and b is the capacity of the knapsack. If the resulting weight of the knapsack is greater than $\overline{\beta}_e$, the call-pattern can be added to the current formulation. Otherwise, no column is generated with respect to arc e.

4. Overall Algorithm

4.1 Overview

In this section, we give a brief and overall explanation of our algorithm. First, we construct initial formulation of MPL without constraints (3), which we call INIT.

After solving the initial INIT, we decide if constraint (4) is satisfied. If it is not, new columns for path variables are generated and added to INIT. If the present solution satisfies constraint (4), i.e., no more columns for path variables need to be added to INIT, we proceed to find a subset of constraints (3) to be added if some violated inequalities are found by the current fractional solution. In this way, the augmented formulation ALP has been obtained.

After getting ALP, we solve ALP and decide if the constraint (5) is satisfied. If it is not, new columns for call set-patterns are generated and added to ALP. If the present solution satisfies constraint (5), no more columns for call set-patterns needs to be added to INIT.

Then, we go though the same procedure as we do after the initial formulation of MPL is obtained. If the solution of ALP is dual feasible, we generated needed columns until it is optimally solved.

When no more columns can be generated, we check if the solution obtained by solving the last LP is integral. If we have obtained an integral solution, we are done with an optimal solution of BWP. Otherwise, we have to initiate the branch-and-price procedure to find an integer optimal solution.

4.2 Branching Strategy

As the branching rule of this problem, we may consider about 3 kinds of rules. The first rule is that of Parker and Ryan (1994). And the second rule is that of Park et al. (1996). The third is as follows.

Our branching rule consists of 2 stages. In the first stage, we decide whether to select a call or not for each call that takes on fractional value. In the second stage, we select the path of the call whose flow on the network is split in any links. For the selection of the path, we modified the branching rule of Bahnhart et al. (2000) to be used in the undirected network model.

5. Computational Results

We tested the proposed algorithm on some randomly generated problems. The underlying networks are devised to reflect the characteristics of the realistic telecommunication networks according to the remarks shown in Anderson et al. (1993). In the following, we mention the characteristics of the general problems and then show the computational results.

5.1 Problem Characteristics

The nodes of a generated network are chosen randomly in two-dimensional Euclidean plane 50×50. These nodes are connected by arcs which are chosen randomly. The arcs are chosen such that nearer nodes are more likely to be connected than distant nodes (Anderson et al, 1993). All of the networks are generated as above such that it satisfies the factor of connectivity.

For each network, we generated 10 problem instances of BWP that have different arc capacities and call tables. The capacities of the arcs are randomly chosen in the range from 10 to 50. Call tables are also randomly generated. Each call has an origin, a destination, a bandwidth requirement, and revenue. Bandwidth requirement of each call was generated between 1 and 20. We managed revenue of a call to be from 100 to 1000 in scale of tens. The calls are randomly generated among all possible pairs of nodes. The number of calls is also randomly determined, proportional to the sizes of the network considered. The bandwidth requirement for each call is generated so that not all the calls can be selected.

For computational test, we used CPLEX callable library version 6.6 as an LP solver. The LP solution routine and the other routines for adding inequalities,

adding columns, and changing bounds of variables are used.

In the next section, we present the performance of the proposed algorithm on the test problems.

5.2 Computational Results

The computational results on 2 networks are summarized in table 1-2.

Table 1. Computational Results on Network (30, 50)

No	#Call	#C1	#C2	#CON	#LP	Gap (%)	#BB	Time
1	100	543	1799	4100	1297	0.45	218	105.8
2	102	335	688	3570	333	0.15	10	13.51
3	94	221	346	2914	189	0.05	22	5.05
4	84	177	251	2772	95	0.08	2	2.08
5	93	208	398	2790	328	0.21	34	7.07
6	99	216	286	3366	126	0.09	6	3.56
7	103	191	145	3193	49	0	0	1.24
8	84	120	152	1932	73	0.1	4	1.03
9	77	173	380	2310	285	0.72	42	5.16
10	79	188	249	3002	56	0.13	2	1.48
Avg	91.5	237.2	469.4	2995	283	0.2	34	14.6
Max	103	543	1799	4100	1297	0.72	218	105.8
Min	77	120	145	1932	49	0	0	1.03

Table 2. Computational Results on Network (35, 85)

No	#Call	#C1	#C2	#CON	#LP	Gap (%)	#BB	Time
1	112	504	820	6720	296	0.11	22	25.24
2	120	626	1697	8040	1379	0.28	252	164.3
3	104	336	432	4992	156	0.21	4	7.19
4	111	525	853	6882	218	0.04	8	25.87
5	113	401	629	6554	270	0.09	24	17.54
6	92	205	157	3404	109	0.04	12	2.59
7	101	289	397	4141	344	0.11	46	11.34
8	101	316	284	4444	88	0	0	3.54
9	113	596	1305	6441	450	0.32	42	64.97
10	98	414	592	5586	256	0.25	20	13.57
Avg	106.5	421.2	716.6	5720	357	0.14	43	33.61
Max	120	626	1697	8040	1379	0.32	252	164.3
Min	92	205	157	3404	88	0	0	2.59

In those tables, the headings #Call, #C1, #C2, #CON and #LPs refer to the number of calls, the number of generated columns of type 1, the number of generated columns of type 2, the number of constraints added from the constraints (3), and the number of calls to LP solver in the algorithm, respectively. All data are accumulated up to the end of the whole procedure. Let LP denote the objective value obtained by solving the final linear programming formulation before the branch-and-price procedure and OPT denote the value of the optimal integer solution. Gap(%) is defined as follow:

$$\text{Gap}(\%) = \frac{\text{LP} - \text{OPT}}{\text{OPT}} \times 100.$$

The headings #BB refer to the numbers of nodes generated in the branch-and-price procedure. Finally, Time refer to the accumulated execution times in seconds needed to solve the problem until branch-and-price procedure. All problems are solved on Pentium PC (866MHz).

Note that the number of columns of type 1 generated in the procedure does not exceed 4 times the number of calls in average. The number of columns of type 2 generated does not exceed twice the number of columns of type 1 in average.

The column Gap(%) shows that our proposed model gives tight bound in the LP relaxation model. The column #BB shows that it does not exceed 100 in the number of enumeration nodes in average.

6. Conclusions

In this paper, we proposed an algorithm to solve the bandwidth packing problem. We used the delayed column generation technique with two kinds of column types. The technique has been used to solve the LP relaxation of the problem efficiently. In the algorithm, we also consider a new branching rule to obtain an integer optimal solution.

The computational results showed that the proposed algorithm gives optimal solutions in small running time.

The approach used in this paper may be applied to other potential applications in telecommunication network design or path or route selection in the capacitated network.

References

Anderson, C.A., F. Fraughnaugh, M. Parker, and J. Ryan, Path Assignment for Call Routing: An Application of Tabu Search, in F. Glover et al. (Eds.), Annals of Operations Research, Vol. 41, 1993.

Barnhart, C., C. Hane, and P.H. Vance, Using Branch-and-Price-Cut To Solve Origin-Destination Integer Multicommodity Flow Problems, *Operations Research*, 48(2000), pp.318-326.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance, Branch-and-Price:

Column generation for solving huge integer programs, *Operations Research*, 46(1998), pp.316-329.

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

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

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

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

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

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

Park, K., S. Kang, and S. Park, An integer programing approach to the bandwidth packing problem, *Management Science*, 42(1996), pp.1277-1291.

Yen, J. Y., Finding the k Shortest Loopless Paths in a Network, *Management Science*, 17 (1971), 712-716