Finding Feasible Solutions for Multi-Commodity Flow Problems

DAI Weibin[1], SUN Xiaoqian[1]* and WANDELT Sebastian[1]*

1. National Key Laboratory of CNS/ATM, School of Electronic and Information Engineering, Beihang University, Beijing, China
E-mail: daiweibin@buaa.edu.cn, sunxq@buaa.edu.cn, wandelt@informatik.hu-berlin.de

Abstract: Many transportation, communications, and logistics problems can be formulated as large multi-commodity flow problems, which are often solved using the traditional column generation method. During the application of the column generation, one needs to find initial solutions, so-called feasible columns in a first step. In order to determine these feasible columns and also solve dependent price problems within an acceptable time, this paper proposes several novel methods: Two methods by solving the global problem and two methods by finding the shortest and widest paths are proposed to compute the initial columns. Furthermore, we propose two methods by finding the shortest paths with commodities and sources (SPC and SPS) for solving price problems. Several well-known network instances are used as case studies in our evaluation to test the efficiency of each method. It is shown that the widest-path method is the best for finding initial columns. For solving the price problems, the preference of the two path-finding methods depend on the network structure.

Key Words: multi-commodity flow problem, column generation, initial columns, price problems

1 Introduction

The multi-commodity flow problem (MCFP) deals with the assignment of commodity flows from sources to destinations in a network. It can be applied in several areas, such as telecommunication and transportation [1]. MCFP has been studied by a number of researchers with a variety of methods, such as the column generation, Lagrangian relaxation, and Dantzig-Wolfe decomposition [1]. Tomlin (1966) proposed the column generation algorithm first [2] and this algorithm is one of the frequently-used methods for solving MCFP. He was also one of the early users of the Dantzig-Wolfe decomposition approach. In the next half a century, a number of new methods based on column generation idea and Dantzig-Wolfe decomposition have been presented. Barnhart et al. (2000) proposed a modified version of the branch-and-bound algorithm based on the column generation for origin-destination integer multi-commodity flow problems [3]. They improved the algorithm for the path-flow model by presenting a new branching rule and adding cuts. The cuts can significantly reduce the computational complexity in many real-world situations. However, since the restricted mater problem (RMP) changes a lot with the continuous column generation iteration, the solution often slowly convergences. In order to solve this problem, Gondzio et al. (2013,2014) proposed the primal-dual column generation method (PDCGM) [4][5][6]. PDCGM is a variant which relies on the suboptimal dual solutions of RMPs where the solutions are obtained with the primal-dual interior point method; initially proposed for solving large-scale convex optimization problems. The authors showed that their method is competitive and suitable in a wide context for optimization. In addition to column generation, the Dantzig-Wolfe decomposition and Lagrangian relaxation have also been improved by researchers. Karakostas (2008) proposed polynomial approximation approaches based on Dantzig-Wolfe decomposition for MCFP [7]. The schemes were based on some previous algorithms [8][9] and the computation time depended least on the number of commodities. Babonneau et al. (2006) proposed a new method based on the partial Lagrangian relaxation [10] and the relaxation is constrained to the set of arcs that are saturated at the optimum. This method can be used to solve large problems.

This work is supported by National Natural Science Foundation (NNSF) of China under Grant 61521091.

Based on the previous research, there are some modified versions of MCFP, such as bi-objective optimization problems. In order to solve such problem, Moradi et al. (2015) proposed a new column generation method. The method was based on the combination of the simplex method and Dantzig-Wolfe decomposition [11]. Similar to Karakostas's method, the evaluation showed that the average computing time does not increase with the number of commodities. Historically, models based on paths were mostly used to solve MCFP. However, Pierre-Olivier et al. (2013,2015) provided a new idea [12, 13]: Instead of generating paths for each commodity, they generated groups of paths in suitable ways. For instance, the paths can be aggregated into trees, into a single set, or into several trees. Compared with other models, their method showed reduced computation times.

As our summary of related works shows, the column generation method has been widely used to solved MCFP. However, in previous methods, there is an often neglected problem of getting initially feasible columns. Many authors only state that the initial feasible solutions can be found by a selection of methods, but do not show how it is actually performed. In this paper, the theory and steps for identifying initial feasible solutions for MCFP are discussed. In order to get initially feasible columns and solve price problems within an acceptable time, this paper proposes a set of distinct methods. It is necessary to make sure that the initial columns are feasible, and the solution of the global problem must be feasible. Thus, two methods for getting a feasible solution of the global problem are proposed. In addition, solving the problem at hand is slow. However, if we can solve each single-commodity problem step-by-step and also consider the interactions between them, we might gain feasibility and reduced running times. Therefore, two methods by finding the shortest and widest paths are proposed to get initial columns. In order to solve the price problems, we propose two methods by finding the shortest paths with commodities and sources (SPC and SPS) In total, we compare a set of 12 competitors.

To test the efficiency of our methods, we evaluate them on a set of well-known MCFP datasets as a case study. For each dataset, we evaluate instances with different sizes, which allows us to discuss the scalability of the techniques. Our findings suggest that shortest-path based methods are fast, yet often lead to infeasible solutions. Widest-path based methods, have similar computational properties, but find a feasible solution for all problems in our evaluation. Our research contributes to the important topic of solving MCFP instances efficiently.

The remainder of this paper is organized as follows. Section 2 provides an introduction for the multi-commodity flow problem and the column generation algorithm. We propose novel methods for finding initial columns in Section 3.1-3.4. Section 3.5 presents the modified methods for solving pricing problems. In order to test the methods, well-known network instances are used as case studies in Section 4. The paper concludes with Section 5.

2 Background

2.1 Multi-commodity flow problem

Let $G = (V, E)$ be a directed graph with n nodes and m edges, where V and E are the sets of nodes and edges, respectively. Each edge has a cost and capacity. In total, t commodities need to be transported between some nodes. For each such commodity k, $(k = 1, 2, ..., t)$, $O(k)$ and $D(k)$ represent its origin and destination of k, respectively. d^k is the travel demand of commodity k. The goal of MCFP is to find a flow assignment that satisfies the travel demand and capacity constraints. We assume that $X^k = (x^k_1, x^k_2, ..., x^k_m)^T, (k = 1, 2, ..., t)$ is the flow of the kth commodity on each edge. Then the MCFP can be formulated as follows:

$$minimize\ z(x) = c^T \sum_{k=1}^{t} X^k \tag{1}$$

$$subject\ to\ \sum_{k=1}^{t} X^k \leq cap \tag{2}$$

$$BX^k = b^k,\ k = 1, 2, ..., t \tag{3}$$

$$X^k \geq 0,\ k = 1, 2, ..., t \tag{4}$$

where $c = (c_1, c_2, ..., c_t)^T$ and $cap = (cap_1, cap_2, ..., cap_t)^T$ are the vectors of cost and capacity on each edge, respectively. Moreover, $b^k = (b^k_1, b^k_2, ..., b^k_n)^T$, $(k = 1, 2, ..., t)$ and is defined as follows:

$$b_i = \begin{cases} d^k,\ if\ i = O(k) \\ -d^k,\ if\ i = D(k) \\ 0,\ otherwise \end{cases} \tag{5}$$

$B = (B_{il})_{n*m}$ is the incidence matrix between nodes and edges and it has a size of $n * m$. We sort all edges as a list. Then for the l-th edge (i, j), we set $B_{il} = 1$ and $B_{jl} = -1$.

2.2 Column generation

In Section 2.1, it is shown that MCFP formulations have a special structure that can be exploited in order to find optimal solutions. The constraint (3) can be divided into t independent equations. The interactions between all commodities are only due to the capacity constraints (2). Thus, with Dantzig-Wolfe decomposition, the column generation algorithm is a commonly-used method for solving multi-commodity flow problem. In this section, the steps of column generation are introduced.

2.2.1 Preparation: Dantzig-Wolfe decomposition

In order to solve MCFP with column generation, we need to use Dantzig-Wolfe decomposition first. In general, for many instances, solved via a simplex-like algorithm, most variables (called columns) do not contribute to the basis. The idea of Dantzig-Wolfe decomposition is to split the original problem into a master problem (containing a superset of the currently active columns) and a a set of sub-problems (which generate new columns to-be-added into the basis to improve the objective function). In our case, the original MCFP is reformulated into a master problem (MP) and t sub-problems. The sub-problems is formulated as follows:

$$minimize\ z(Y^k) = c^T Y^k \tag{6}$$

$$subject\ to\ Y^k \leq cap \tag{7}$$

$$BY^k = b^k \tag{8}$$

$$Y^k \geq 0 \tag{9}$$

where $Y^k = (x^k_1, ..., x^k_m)^T$, $k = 1, 2, ..., t$ is the vector of flow of commodity k on each edge. In fact, each sub-problem is just a single-commodity flow problem. By solving them, we can get several feasible solutions $Y^k_1, Y^k_2, ..., Y^k_{n_k}$, $(k = 1, 2, ..., t)$ and these Y^k_j are called 'columns'.

Thus, we decompose X^k as $X^k = \sum_{j=1}^{n_k} \lambda^k_j Y^k_j$, where λ^k_j is coefficient and it satisfies $\sum_{j=1}^{n_k} \lambda^k_j = 1$ for each $k = 1, 2, ..., t$. Therefore, the master problem (MP) can be formulated as follows:

$$minimize\ z(\lambda) = \sum_{k=1}^{t} \sum_{j=1}^{n_k} (c^T Y^k_j) \lambda^k_j \tag{10}$$

$$subject\ to\ \sum_{k=1}^{t} \sum_{j=1}^{n_k} Y^k_j \lambda^k_j \leq cap \tag{11}$$

$$\sum_{j=1}^{n_k} \lambda^k_j = 1,\ k = 1, 2, ..., t \tag{12}$$

$$\lambda^k_j \geq 0,\ j = 1, 2, ..., n_k, k = 1, 2, ..., t \tag{13}$$

If the master problem is solved, the optimal solution based on this set of columns can be obtained. However, until that point, it is not clear whether the obtained solution is also a global optimum. Therefore, we need to solve the dual problem and price problems as well.

2.2.2 Dual problem

The *dual problem* is the dual problem of master problem. The master problem (10)-(13) can be formulated with matrices as follows:

$$minimize\ z(\lambda) = c^T Y \lambda \tag{14}$$
$$subject\ to\ Y\lambda \le cap \tag{15}$$
$$N\lambda = \mathbf{1}_t \tag{16}$$
$$\lambda \ge \mathbf{0}_{num} \tag{17}$$

where $Y = (Y_1^1, Y_2^1, ..., Y_{n_1}^1, Y_1^2, ..., Y_{n_t}^t)$ is the matrix of columns and its size is $m * num$ (here num is the number of columns). $\lambda = (\lambda_1^1, \lambda_2^1, ..., \lambda_{n_1}^1, \lambda_1^2, ..., \lambda_{n_t}^t)^T$. $\mathbf{1}_t = (1, 1, ..., 1)_t^T$ and $\mathbf{0}_{num} = (0, 0, ..., 0)_{num}^T$

$N = (N_{ij})_{t*num}$ is the incidence matrix between k and λ_j^k. It can be defined as follows (for instance, $n_1 = 3$, $n_2 = 2$ and $n_3 = 3$...)

$$N = \begin{pmatrix} 1\,1\,1\,0\,0\,0\,0\,0\,...\,0 \\ 0\,0\,0\,1\,1\,0\,0\,0\,...\,0 \\ 0\,0\,0\,0\,0\,1\,1\,1\,...\,0 \\ ... \\ 0\,0\,0\,0\,0\,0\,0\,0\,...\,1 \end{pmatrix} \tag{18}$$

Let $w = (w_1, w_2, ..., w_m)^T$ and $\alpha = (\alpha_1, ..., \alpha_t)^T$ be the vectors of dual variables for (18) and (19), separately. The size of them are $m * 1$ and $t * 1$. Thus, the dual problem can be formulated as follows:

$$maximize\ z_d(w, \alpha) = cap^T w + \mathbf{1}_t^T \alpha \tag{19}$$
$$subject\ to\ Y^T w + N^T \alpha \le Y^T c \tag{20}$$
$$w \ge \mathbf{0}_m \tag{21}$$

2.2.3 Price problem

After solving the dual problem, we can obtain the optimal value of w and α. Then we can construct price problems and solve them. For each $k,\ k = 1, 2, ..., t$, the price problem is formulated as follows:

$$minimize\ u(x^k) = (c - w)^T x^k - \alpha_k \tag{22}$$
$$subject\ to\ Bx^k = b^k \tag{23}$$
$$x^k \ge 0 \tag{24}$$

Once we obtained the optimal objective value u^k for each k, we can check them. It is proved that if $u^k \ge 0, \forall k = 1, ..., t$, then the solution of the master problem (13)-(16) is optimal. If $u^k < 0, k = k_1, k_2, ..., k_p$, then we need to add $x^k, k = k_1, k_2, ..., k_p$ into the set of columns, reconstruct and solve the new dual problem and price problems, until $u^k \ge 0, \forall k = 1, ..., t$.

With the phases above, we can solve the multi-commodity flow problem based on the column generation algorithm. However, there are some open problems, which are addresses below.

3 Method for finding initial columns and modification for solving price problems

Before using the column generation method, a set of initially feasible columns (=initial solutions) are needed. In Section 2.2.1, they are obtained by solving t sub-problems (6)-(9). In fact, these sub-problems are single-commodity flow problems for each commodity. If we have a set of feasible solutions $\{Y_j^k,\ j = 1, 2, ..., n_k,\ k = 1, 2, ..., t\}$, we may get the optimal solution of the master problem in the form of $X^k = \sum_{j=1}^{n_k} \lambda_j^j Y_j^k$ $(\sum_{j=1}^{n_k} \lambda_j^k = 1,\ k = 1, 2, ..., t)$ in most cases.

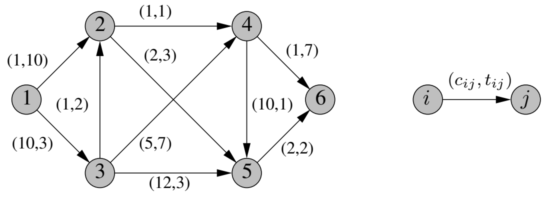

Fig. 1: small example I

Table 1: Example commodities

commodity	O	D	travel demand
1	1	4	1
2	1	5	4
3	1	6	2

3.1 Two-phase method

In general, the sub-problems are solved with two-phase method for the feasibility of solutions. If we solved them directly, only one solution can be obtained for each sub-problem. These solutions are feasible to each sub-problem, but it is very likely to be infeasible while solving the master problem based on them. Therefore, the two-phase method is used to get another feasible solution before getting the optimal solution. In the first phase, it adds several artificial variables to the linear program the phase-I problem can be formulated as follows:

$$minimize\ z(Y^k, s^k) = \mathbf{1}_n s^k \tag{25}$$
$$subject\ to\ Y^k \le cap \tag{26}$$
$$BY^k + s^k = b^k \tag{27}$$
$$Y^k, s^k \ge 0 \tag{28}$$

where $s^k = (s_1^k, ..., s_n^k)^T$ is the vector of artificial variables. If linear program (25)-(28) has an optimal solution with $s^k = \mathbf{0}_n$, then the previous sub-problem has a feasible solution. The solution Y^k obtained in this phase can be added into the initial column set. The second phase is just to solve the previous sub-problem (6)-(9), and the second group of initial columns can be got. Therefore, we can solve the master problem based on these two groups of Y_j^k. However, this method has a limitation: In some cases, even though two groups of initial columns are obtained, the linear combination of these feasible solutions of sub-problems may also be infeasible to the master problem. For instance, there is a small example in Figure 1 (This graph is taken from [1]). There are 6 nodes and 10 edges in this network. At each edge, a pair of numbers represents the $(cost,\ capacity)$ value. There are also 3 commodities and their OD pairs (origin and destination) are shown in Table 1.

With the two-phase method, the two groups of initial columns are obtained and they are shown in Table 2.

Table 2: Y_j^k obtained by two-phase method

edge	Y_1^1	Y_2^1	Y_1^2	Y_2^2	Y_1^3	Y_2^3	capacity
(1,2)	1	1	3	4	1	2	10
(1,3)	0	0	1	0	1	0	3
(3,2)	0	0	0	0	0	0	2
(2,4)	1	1	0	1	1	1	1
(2,5)	0	0	3	3	0	1	3
(3,4)	0	0	0	0	1	0	7
(3,5)	0	0	1	0	0	0	3
(4,5)	0	0	0	1	0	0	1
(4,6)	0	0	0	0	2	1	7
(5,6)	0	0	0	0	0	1	2

Table 3: a feasible solution to the master problem

edge	(1,4)	(1,5)	(1,6)	cost	capacity
(1,2)	1	3	0	1	10
(1,3)	0	1	2	10	3
(3,2)	0	0	0	1	2
(2,4)	1	0	0	1	1
(2,5)	0	3	0	2	3
(3,4)	0	0	2	5	7
(3,5)	0	1	0	12	3
(4,5)	0	0	0	10	1
(4,6)	0	0	2	1	7
(5,6)	0	0	0	2	2

Notice the edge $(2,4)$, because $\sum_{j=1}^{n_k} \lambda_j^k = 1$, $(\forall k = 1, 2, ..., t)$, no matter how we assign λ_j^k, the capacity of this edge is not enough.

In fact, this is a feasible solution for this problem. As shown in Table 3, the 3 columns in the middle are the flows on each edge for each commodities, respectively. The right two columns are the cost and capacity of each edge. Because of the large cost, edge $(1,3)$, and $(3,4)$ is hardly chosen with the two-phase method in Table 2. However if we want to get feasible solution, these edges have to be used.

In general, because the two-phase method solves each single-commodity flow problem independently, it is very possible to be infeasible while solving the master problem. Therefore, it is necessary to try some other methods.

3.2 None-objective method

It is possible to get infeasible while solving each sub-problem independently. Thus, if the problem is solved globally, it must be feasible. However, the global problem is just the previous MCFP. Therefore, I set the objective function as 0 and just need a feasible solution. This method is called None-objective method and it can be formulated as follows:

$$minimize\ 0 \tag{29}$$

$$subject\ to\ \sum_{k=1}^{t} X^k \leq cap \tag{30}$$

$$BX^k = b^k,\ k = 1, 2, ..., t \tag{31}$$

$$X^k \geq 0,\ k = 1, 2, ..., t \tag{32}$$

This formulation is also equal to artificial-variable method (36)-(42).

$$minimize\ \sum_{i=t+1}^{2t} X^i \tag{33}$$

$$subject\ to\ \sum_{k=1}^{t} X^k \leq cap \tag{34}$$

$$BX^k + X^{k+t} = b^k,\ k = 1, 2, ..., t \tag{35}$$

$$X^k \geq 0,\ k = 1, 2, ..., 2t \tag{36}$$

Because this method does not need optimal condition, it may use less time to get an initial solution. However, the size of this method is equal to the previous MCFP, the reduction of time may be very little.

3.3 Edge-path method

This method is a combination of edge-flow model and path-flow model.

It only needs an initially feasible solution in the first step. Thus, it may be fast if we find a small set of paths and do the linear program based on these paths. What we need to guarantee is to satisfy all the travel demand and capacity constraints ('feasibility').

However, it may get infeasible solution in some cases of networks. Thus, we combine the edge-flow model with it: We set a parameter N_s. For each commodity, we find the shortest path between its OD pair. If the edge number of this path is less than N_s, we find a set of paths for this commodity. Otherwise, we use edge-flow model. In summary, the formulation of the edge-path model is shown as follows (we also just need a feasible solution, so the objective function is 0).

$$\min\ 0 \tag{37}$$

$$subject\ to\ \sum_{k\in P_1}\sum_{h\in H^k} f_h^k \delta_l^h + \sum_{k\in P_2} g_l^k \leq cap_l,\ \forall l \in E \tag{38}$$

$$\sum_{h\in H^k} f_h^k = d^k,\ \forall k \in P_1 \tag{39}$$

$$\sum_{l\in N_i^+} g_l^k - \sum_{l\in N_i^-} g_l^k = b_i^k,\ \forall i \in V, k \in P2 \tag{40}$$

$$f_h^k \geq 0,\ \forall h \in H^k, k \in P_1 \tag{41}$$

$$g_l^k \geq 0,\ \forall l \in E, k \in P_2 \tag{42}$$

where

- f_h^k and g_l^k are the flow of OD pair k on path h and edge l, respectively.
- P_1 and P_2 are the set of OD pairs that use path-flow model and edge-flow model, respectively.
- H^k is path set of OD pair k, $\forall k \in P_1$
- d^k is set travel demand of OD pair k
- N_i^+ and N_i^- are the set of edges that go out node i and go into node i, respectively.

$$b_i = \begin{cases} d^k,\ if\ i = O(k) \\ -d^k,\ if\ i = D(k) \\ 0,\ otherwise \end{cases} \quad (43)$$

3.4 Two finding-path methods

The infeasible case of two-phase method is caused by the independence of each sub-problem. On the other hand, solving the global problem is very time consuming. Thus, we solve each single-commodity flow problem step by step, and also consider the influence between them. For each commodity, we find the shortest path for that commodity first. Then we find the minimum capacity of an edge on this path. If this capacity is larger than the travel demand of this commodity, we reduce the capacities of all edges on the path by the travel demand. Otherwise, we just reduce the edge capacities by the minimum capacity and continue to find another shortest path. After satisfying one commodity's travel demand, it will continue to find shortest path for the next one in updated network. We call this method as **shortest-path method**. For instance, as shown in Figure 1, we find a shortest path $1- > 2- > 4$ for commodity 1 from node 1 to node 4 first. Because the travel demand is 1, the capacity of edge $(1, 2)$ and $(2, 4)$ are reduced to 9 and 0. Then we continue to find the shortest path of next commodity. With this method, we can get an initially feasible solution fast in most cases of networks. However, it does not work in some extreme cases. Therefore, anther finding-path method based on the capacity is proposed.

The most portion of steps of these two methods are the same. The only difference between them is the edge weight in the network. The weight of each edge is replaced with the reciprocal of its capacity (i.e. $\frac{1}{cap_{ij}}$, where cap_{ij} is the capacity of edge (i, j)). If the minimum-weight path for a commodity in this network is found, then the edge capacity in this path would tend to be large. It is more possible to be feasible than the finding-shortest-path method while solving the master problem. This method is called **widest-path method**.

3.5 Modification for solving price problems

In order to check whether the master problem's solution is the global optimum, we need to solve the price problems (22)-(24). It is a simple linear program, we can solve it directly (called linear program, LP). However, please note that the program is a problem for finding shortest path in fact. The only difference is just reduce the cost vector c by vector w. Thus, the second method for solving price problems is to find the shortest paths for each commodity k $(k = 1, 2, ..., t)$ (called shortest paths with commodities, SPC). For further analysis, we need to find the shortest paths for different commodities in the same network. Thus, there is another approach. We can assign the commodities with the same source in a group and find all shortest paths from this source (called shortest paths with sources, SPS). With this approach, the speed may be faster. For instances in Table 1, there are 3 commodities and we can find shortest paths for each of them (SPC). However, it is shown that all the commodities are started from node 1. Thus, it may be fastest to find all shortest paths from node 1 together (SPS).

Table 4: Overview of network instances

instances	n	m	t
grid1	25	80	50
grid2	25	80	100
grid3	100	360	50
grid4	100	360	100
grid5	225	840	100
grid6	225	840	200
grid7	400	1520	400
grid8	625	2400	500
grid9	625	2400	1000
planar30	30	150	92
planar50	50	250	267
planar80	80	440	543
planar100	100	532	1085
planar150	150	850	2239

4 Evaluation

In order to test the feasibility and scalability of each method, the $grid\ networks$ and $planar\ networks$ are used as case studies. All datasets are available for download from `www.di.unipi.it/optimize/Data/MMCF.html`. Relevant properties of each network are shown in Table 4, where n, m, t are the numbers of nodes, edges and commodities, separately.

The computation time while using 4 methods for finding initial columns and 3 methods for solving price problems are shown in Table 5. In the second row, the 3 methods for solving the price problems, (i.e. solving linear program, finding shortest paths with commodities and finding shortest paths with sources) are denoted by LP, SPC and SPS, respectively.

In $grid\ networks$, the two finding-path methods (i.e. shortest-path and widest-path) are much faster than the other two methods for finding initial columns. Furthermore, although the shortest-path method is a little faster than the widest-path method, it may get infeasible sometimes. The frequency of infeasibility is much higher in the $planar\ networks$ with shortest-path method (It is all infeasible except planar30 and planar50.). Totally, the widest-path method has the best property on computation time and feasibility for finding initial columns.

For solving the price problems, SPC and SPS are both faster than LP. However, the speed comparison between SPC and SPS are different in the two kinds of networks. In the $grid\ networks$, SPC is faster than SPS, but it is opposite in the $planar\ networks$. From Table 4, it is shown that there are much more commodities in $planar\ networks$ while the network sizes are close. Thus, it is faster to find all shortest paths from one source together than to find paths for one commodity each step in $planar\ networks$. Therefore, SPC method is better for problems with less commodities and SPS method is better for problems with more commodities. It should be noted that all the methods above compute an optimal solution, once the partial solution is feasible. This is a very valuable property.

5 Conclusions

In this paper, we proposed and discussed several methods to find feasible columns and solve the price problems during the process of applying column generation algorithm to

Table 5: time for each method

Instances	None-objective method			Edge-path method			Shortest-path method			Widest-path method			Optimal
	LP	SPC	SPS	LP	SPC	SPS	LP	SPC	SPS	LP	SPC	SPS	
grid1	0.188	0.146	0.14	0.114	0.076	0.075	0.091	0.031	0.029	0.053	0.027	0.027	8.273e+5
grid2	0.747	0.609	0.597	0.278	0.17	0.153	infea	infea	infea	0.098	0.057	0.045	1.705e+6
grid3	0.691	0.574	0.594	0.49	0.335	0.385	0.215	0.095	0.121	0.167	0.084	0.104	1.525e+6
grid4	2.49	2.14	2.20	1.14	0.771	0.808	0.638	0.264	0.283	0.588	0.255	0.283	3.032e+6
grid5	8.71	7.09	7.07	5.07	3.65	3.89	1.54	0.63	0.834	2.07	0.645	0.855	5.050e+6
grid6	67.85	70.11	69.62	102.37	100.50	101.92	6.32	2.52	2.71	5.62	2.69	2.94	1.040e+7
grid7	495.31	467.32	468.56	>10min	>10min	>10min	25.47	7.98	8.79	24.98	8.96	9.72	2.586e+7
grid8	>10min	>10min	>10min	>10min	>10min	>10min	127.22	40.41	43.56	115.27	48.89	53.13	4.171e+7
grid9	>10min	>10min	>10min	>10min	>10min	>10min	277.99	146.92	146.50	310.17	163.48	163.64	8.265e+7
planar30	0.285	0.229	0.212	0.363	0.286	0.263	0.166	0.088	0.074	0.235	0.093	0.073	4.435e+7
planar50	3.59	2.94	2.70	2.10	1.43	1.31	1.18	0.575	0.403	1.07	0.624	0.425	1.222e+8
planar80	34.69	31.95	31.20	7.34	7.27	7.30	infea	infea	infea	7.13	3.94	3.10	1.824e+8
planar100	241.15	217.41	226.81	43.76	37.19	34.50	infea	infea	infea	14.51	7.19	5.22	2.313e+8
planar150	>10min	>10min	>10min	>10min	>10min	>10min	infea	infea	infea	179.60	131.07	121.08	5.481e+8

multi-commodity flow problems. After Dantzig-Wolfe decomposition, the original problem is divided into the master problem and several sub-problems. Although we can get feasible solutions for every sub-problems with the two-phase method, the linear combination of these solutions may be infeasible in the master problem.

Thus, four new methods were presented in this paper. The first two methods, the none-objective method and the edge-path method, are the methods by solving the global problem. They can guarantee the feasibility of the solutions, but their computation complexity is unacceptable. The other two methods, the shortest-path method and the widest-path method, are the approaches by finding paths for commodities. The former is to find the path with shortest cost, while the latter is to find the path with largest capacity.

The price problems are very important in each loop for the column generation. From the formulations, it can be found easily that the price problems are not only linear programs, but also problems for finding shortest paths in fact. Thus, two methods, finding shortest paths with commodities (SPC) and finding shortest paths with sources (SPS), were proposed.

In order to test the efficiency and speed of each method, the $grid\ networks$ and $planar\ networks$ were used as case studies. It is shown that although the shortest-path method is faster than the other three methods for finding initial columns, it might also become infeasible in some cases. The widest-path method is slightly slower, but it can get feasible columns in all the instances proposed in this paper. In summary, the widest-path method has the best property for finding initial columns. For solving the price problems, the SPC method is more suitable for problems with less commodities; while the SPS method is more suitable for problems with more commodities.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows," DTIC Document, Tech. Rep., 1988.

[2] J. Tomlin, "Minimum-cost multicommodity network flows," *Operations Research*, vol. 14, no. 1, pp. 45–51, 1966.

[3] C. Barnhart, C. A. Hane, and P. H. Vance, "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems," *Operations Research*, vol. 48, no. 2, pp. 318–326, 2000.

[4] J. Gondzio, P. Gonzlez-Brevis, and P. Munari, "Large-scale optimization with the primal-dual column generation method," *Mathematical Programming Computation*, pp. 1–36, 2013.

[5] ——, "New developments in the primalcdual column generation technique," *European Journal of Operational Research*, vol. 224, no. 1, pp. 41–51As the access to this document is restricted, you may want to look for a different version under "Related research" (further below) orfor a different version of it., 2013.

[6] J. Gondzio and P. Gonzlez-Brevis, "A new warmstarting strategy for the primal-dual column generation method," *Mathematical Programming*, vol. 152, pp. 1–34, 2014.

[7] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," *ACM Transactions on Algorithms (TALG)*, vol. 4, no. 1, p. 13, 2008.

[8] N. Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," 1998.

[9] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 4, pp. 505–520, 2000.

[10] F. Babonneau and J. P. Vial, "Solving large-scale linear multicommodity flow problems with an active set strategy and proximal-accpm," *Operations Research*, vol. 54, no. 1, pp. 184–197, 2006.

[11] S. Moradi, A. Raith, and M. Ehrgott, "A bi-objective column generation algorithm for the multi-commodity minimum cost flow problem," *European Journal of Operational Research*, vol. 244, no. 2, pp. 369–378, 2015.

[12] P.-O. Bauguion, W. Ben-Ameur, and E. Gourdin, "A new model for multicommodity flow problems, and a strongly polynomial algorithm for single-source maximum concurrent flow," *Electronic Notes in Discrete Mathematics*, vol. 41, pp. 311–318, 2013.

[13] ——, "Efficient algorithms for the maximum concurrent flow problem," *Networks*, vol. 65, no. 1, pp. 56–67, 2015.