Using Circuit Variables to Accelerate Column Generation in Planar Multicommodity Flows

Filipe Alvelos [1], J. M. Valério de Carvalho [1]
[1] Departamento de Produção e Sistemas, Centro de Investigação Algoritmi
Address: Campus de Gualtar, Universidade do Minho, 4710-057 Braga, Portugal
Email: {vc,falvelos}@dps.uminho.pt

January 2005

Abstract

In this work we present a way of accelerating a column generation algorithm for the linear minimum cost multicommodity flow problem. We use a new model that, besides the usual variables corresponding to flows on paths, has a polynomial number of extra variables, corresponding to flows on circuits. Those extra variables are explicitly considered in the restricted master problem, from the beginning of the column generation process. The subproblem remains a set of shortest path problems, one for each commodity.

We present computational results for the comparison of this new approach with standard column generation, a bundle method and a general-purpose solver. For the tested instances, there is an effective improvement in computational time of the column generation method when the extended model is used.

keywords multicommodity flows, column generation, planar graphs.

1 Introduction

Multicommodity flow models have been used for several decades to deal with optimisation problems in transportation/distribution systems, telecommunications networks and production planning, among others.

In this work we consider the minimum cost linear multicommodity flow problem (MFP) defined in a planar network. This problem is defined over a network in which we want to route, with minimal cost, a set of commodities from their origin to their destination without exceeding the capacities of the arcs. We consider that, for each arc, the costs of all commodities are the same. This is a usual assumption in some applications of multicommodity flow models, as, for instance, in telecommunications networks, where the commodities are usually associated with streams of traffic between different pairs of users and the quality of the solution does not depend on the particular flow pattern of each commodity, but on the overall flow pattern.

A description of several applications of multicommodity flow models can be found in the surveys presented in [12], [4] and Chapter 17 of [1]. In the same surveys, classical methods are described. More recently, several approaches have been developed. Among them, we refer to specialized interior point methods [19] [6], bundle methods [9], analytic center cutting plane methods [10] and the hybridisation of integer and constraint logic programming techniques [18].

In this work we focus on the column generation method. Although almost 50 years has passed since its first application [7] (precisely on a multicommodity flow problem), in recent years a renewed interest on this method had appear, derived from its potential when embedded in integer programming methods and from the perception that there are several open roads to explore in order to improve its convergence properties [8, 15].

It is well known that column generation algorithms have poor convergence properties [14]. To deal with that issue several methods have been devised, usually taking a dual perspective, given that column generation can be seen as the Kelley's cutting plane algorithm applied to the dual problem [16].

Our present work is based on the method proposed by Valério de Carvalho [5]. The fundamental idea is that by adding extra variables (extra dual cuts), prior to the beginning of the column generation process, we can speed up the column generation algorithm and improve its practical convergence properties. The motivation is the following: the use of a tighter dual space restriction from the start may help in finding the optimal solution faster. Under certain conditions, the primal space is not relaxed, and it is possible to recover an optimal solution to the original problem from the optimal solution to the extended model. In this work we apply a similar idea to the MFP defined over planar graphs.

The extra variables added to the model must be tailored to the specific problem at hand. In the case of the MFP, we use variables associated with flows on circuits. An important issue is that the number of extra variables must be tractable by the master problem. We restrict our present work to the MFP defined on planar graphs, where it is possible to select a polynomial number of circuits, based on which all circuits of the network are implicitly considered. Our final aim is to generalise this approach to the MFP defined on general a graph.

In Section 2, we briefly review the standard column generation algorithm for the path formulation of the MFP. In Section 3, our approach is presented and discussed. Comparative computational results are the subject of Section 4. In Section 5, conclusions of this work are taken and future work is considered.

2 Path Formulation and Standard Column Generation

We start this section by defining the notation that will be used.

We consider a network formed by a set of *n* nodes, being *N* the set of their indices, and a set of *m* of arcs, being *A* the set of indices *ij* of the arcs (arc *ij* has origin in node *i* and destination in node *j*). We define a set *K* of indices of *h* commodities. Each commodity $k \in K$ is characterised by an origin node, a destination node and a demand, r_k. We also define an integer capacity u_{ij} and a unit cost, c_{ij}, both associated with each arc of the network.

We represent the set of indices of all simple paths between the origin and the destination of commodity *k* by P^k. If arc *ij* belongs to path *p* of commodity *k*, then d_{ij}^{pk} equals *1*, and *0*, otherwise. The unit flow cost of path *p* of commodity *k*, is $c^{pk} = \sum_{ij \in A} d_{ij}^{pk} c_{ij}$, $\forall k \in K$, $\forall p \in P^k$. When we refer to a path of a commodity we mean a simple path that begins at the origin of a commodity and ends at its destination.

Defining decision variables, y^{pk}, as the flow on each path of all commodities, we obtain the path formulation:

$$\begin{aligned}
&\text{Minimise} \sum_{k \in K} \sum_{p \in P^k} c^{pk} y^{pk} \\
&\text{s.t.} \sum_{p \in P^k} y^{pk} = r^k, \ \forall k \in K \qquad (1) \\
&\sum_{k \in K} \sum_{p \in P^k} d_{ij}^{pk} y^{pk} \le u_{ij}, \ \forall ij \in A \quad (2) \\
&y^{pk} \ge 0, \ \forall k \in K, \ \forall p \in P^k.
\end{aligned}$$

Constraints *(1)* ensure that the all the demand of each commodity is routed from its origin to its destination. Constraints *(2)* ensure that the capacities of all the arcs are respected. The exponential number of decision variables of this model inhibits the possibility of the use of direct methods (such as simplex).

Applying column generation, a restricted master problem (RMP), that is a problem where not all paths are considered, is first optimised. After optimising the RMP, the attractiveness of the paths that are not present in the RMP is evaluated by solving a subproblem that uses the values of the dual variables of the RMP. That subproblem consists in determining the shortest path between their origin and their destination for all commodities in a network with (non negative) costs. After inserting the attractive paths in the RMP, the procedure is repeated until no attractive paths are returned by the subproblem. A detailed exposition of this procedure is presented in [1].

3 Accelerating Column Generation for Planar Problems

3.1 An extended model with circuits

The main idea of our procedure to accelerate the column generation method to solve the linear multicommodity flow problem is to add variables corresponding to circuits to the path formulation, thus obtaining a new model. The circuits that we consider have no arc repetitions, are oriented and formed by, at least, three arcs. One of the arcs of the circuit is traversed in the opposite direction of its orientation (backward arc) and all the remaining arcs are traversed in the same direction as their orientation (forward arcs).

We consider the set D formed by all the circuits such as the ones defined above, indexed by $s=1,...,|D|$. From now on, when referring to circuits, we mean circuits belonging to D, except when explicitly stated otherwise.

We define a parameter g_{ij}^{s}, "*ij* Î*A*, "*s* Î*D*, that is equal to 1, if arc ij is a forward arc of circuit s; equal to -1 if arc ij is the backward arc of circuit s; equal 0, if arc ij does not belong to circuit s. We associate with each circuit s, "*s* Î*D*, a variable d^s ³$0$, which corresponds to the flow in circuit s. The unit flow cost of circuit s is $c^s = \sum_{ij\in A} g_{ij}^{s} c_{ij}$, "*s* Î*D*.

By using the circuits defined above, it may be possible to represent one solution of the path model by a set of paths and circuits with flow, since some paths with flow can be represented as the sum of other paths and circuits. We note that the definition of the unitary costs of the paths and of the circuits implies that the cost of the solution is the same in both representations.

As an example, consider a (partial) solution in which there is a flow of commodity k in the two paths as represented in Figure 1. For easiness of explanation, suppose that y^{1k} ³ y^{2k}. The same (partial) solution can be represented as showed in Figure 2, where the new flow on path 1, denoted by y^{1k*}, is $y^{1k*} = y^{1k} + y^{2k}$ and $d^s = y^{2k}$, since the sum of path 1 and circuit s gives path 2.

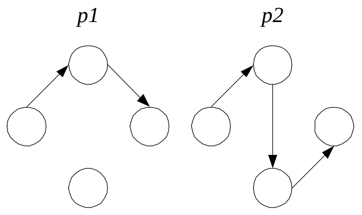

Figure 1: One solution represented with two paths.

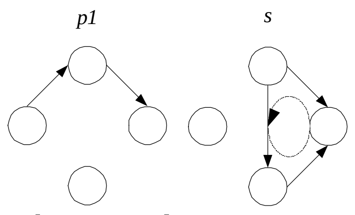

Figure 2: The same solution represented as one path and one circuit.

We now present an extended model that, besides the path variables, also includes the above defined circuit variables:

$$\text{Minimise} \sum_{k\in K}\sum_{p\in P^k} c^{pk} y^{pk} + \sum_{s\in D} c^s d^s$$

$$\text{s.t.} \sum_{p\in P^k} y^{pk} = r^k \text{ , "kÎK} \qquad (3)$$

$$\sum_{k\in K}\sum_{p\in P^k} d_{ij}^{pk} y^{pk} + \sum_{s\in D} g_{ij}^{s} d^s \text{ £ } u_{ij} \text{ , "ijÎA} \qquad (4)$$

$$\sum_{k\in K}\sum_{p\in P^k} d_{ij}^{pk} y^{pk} + \sum_{s\in D} g_{ij}^{s} d^s \text{ ³ } 0 \text{ , "ijÎA} \qquad (5)$$

$$y^{pk} \text{ ³ } 0, \text{ "kÎK, "pÎ}P^k$$

$$d^s \text{ ³ } 0, \text{ "sÎD.}$$

We can see this model as giving an optimal solution on paths and then, through the circuits, redirect their flow to other paths. Circuit variables are not present in constraints *(3)* since they respect the flow conservation of all nodes.

We do not associate commodities with the circuit variables because they all have the same cost and the circuits only have one backward arc. Thus, the choice of the commodity for which its flow (or part of it) is redirected is irrelevant. In this way, we keep the number of new variables and constraints small. Furthermore, *(4)* and *(5)* can be joined into a ranged constraint.

The new set of constraints *(5)* state that it is only possible to redirect flow through a circuit if that flow exists. If we had not constrained the total flow of the arcs to be nonnegative we could get an unbounded solution (since some circuits may have a negative cost), or a solution that was impossible to transform into a feasible solution to the arc-path formulation, since it had a smaller value. In other words, the set of constraints *(5)* assure that the original space is not relaxed: only positive flows are still allowed.

With this methodology, we are adding more variables and turning a set of constraints into a set of ranged constraints. In turn, with the same set of paths in the RMP and with the circuits, we are considering implicitly a much larger number of extreme points of the subproblems, since, besides the paths that are present in the RMP, we are considering all the paths that can be obtained by adding circuits to them. We note that the same circuit, when combined with different paths, may allow the implicitly consideration of several paths.

So far, we neglected the fact that the number of circuit variables can be very large. In fact, in a general network, it is exponential with respect to the dimension of the network. Here we just present results for instances defined in planar graphs. In this case, we can identify a polynomial number of simple circuits such that all the graph circuits can be represented as nonnegative combinations of them. Based on those circuits, we can define a polynomial number of circuit variables (see [3] for a formal a proof).

3.2 Dealing with negative cost cycles in the subproblem

The dual variables of constraints *(5)* contribute with a negative value to the modified cost of the arcs in the subproblem objective function, leading to arcs with a negative modified cost, and even to negative cost cycles. Note that this never happens in the column generation procedure for the path formulation, because it has not any greater than or equal to constraints.

Therefore, the subproblem of a commodity k still is a shortest path problem but now in a network that can have negative cycles (all arcs traversed in the direction of their orientation), which is NP-hard. In order to avoid solving a NP-hard problem, we can overcome this issue by solving a shortest path problem with a label correcting algorithm, and, when a negative cycle is detected, we introduce the associated variable in the RMP. After reoptimising the RMP, the negative cost cycle previously detected will never be generated again.

In this way subproblems may suggest paths or cycles to the RMP. This approach is presented in [2] in the context of a branch-and-price algorithm for the integer multicommodity flow problem.

Note that these cycle variables are not related with the extra circuit variables that are present from the very beginning in the RMP. All the arcs of the cycles are forward arcs.

3.3 Obtaining the optimal solution of the path formulation

If there are no circuit variables with positive values then the actual solution is optimal to the arc-path model. Otherwise, we developed a procedure, fully described in [3], to perform the conversion of one optimal solution of the extended model to one optimal solution to the path model, with the same cost. The main idea is redirecting all the flow on circuits to paths, forcing circuit variables to have a null value one after the other.

4 Computational Tests

We compared an implementation of our proposed method (ACG) with an implementation of standard column generation (CG), with other specialized code for multicommodity flow problems based on a bundle method [9] (B) and Cplex 6.6 [11] (C) solving directly the arc formulation. Our code was developed in C++ using the programming environment Microsoft Visual Studio 6.0. We used the class library LEDA 4.1 [17] for identifying the circuits, to solve the shortest path subproblems and to randomly generate of the tested instances. We used Cplex 6.6 to solve the restricted master problems. The Bundle code was provided to us by Professor Antonio Frangioni. We note that this code can be used with more general instances (namely instances with multiple origins and destinations). We also note that the adjustment of its parameters was, by no means, exhaustive and not done by an experienced user. In spite of the theoretical closeness of the two methods (column generation and bundle), the implementations tested have a major difference: the column generation uses a disaggregated approach and the bundle an aggregated one.

We tested three groups of instances: A, B and Planar. We randomly generated A and B instances. The values presented for these instances in Table are average values for sets of three instances with the same characteristics. The other group of instances comes from [13] and was obtained in the address (maintained by Antonio Frangioni) *http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html*. We refer to this last set of instances as the planar instances, in order to keep their original designation, but noting that all the instances in all sets are planar.

The results given in Table 1 were obtained on a personal computer with a Pentium 4, 2 GHz processor, 1 GB of RAM, running Windows XP Professional Edition. All the time values are expressed in seconds. Last column gives the relative difference (in percentage) between standard column generation and accelerated column generation.

Table 1: Computational results.

Instance(s)	*n*	*m*	*h*	C	B	CG	CGA	%Δ
A100dl	100	569	600	195.2	18.2	1.2	1.1	-10.2
A100ds	100	572	300	29.7	9.4	0.6	0.7	31.9
A100sl	100	400	600	77.1	5	0.8	0.8	-4.1
A100ss	100	400	300	23.8	2.9	0.5	0.5	5.3
A600dl	600	3562	3600	-	-	154.4	52.8	-65.8
A600ds	600	3557	1800	-	-	82.7	30.9	-62.6
A600sl	600	2400	3600	-	-	173.5	98.4	-43.3
A600ss	600	2400	1800	-	-	54.9	34.4	-37.4
B100dl	100	285	600	20.1	38.3	2.1	1.2	-44.9
B100ds	100	286	300	7.2	5.3	0.7	0.5	-31.1
B100sl	100	200	600	3.5	10.6	0.6	0.4	-38.2
B100ss	100	200	300	1.5	2.2	0.3	0.2	-1.3
B600dl	600	1771	3600	-	-	1470.9	175.1	-88.1
B600ds	600	1778	1800	-	-	685.3	166	-75.8
B600sl	600	1200	3600	-	-	255.1	78.9	-69.1
B600ss	600	1200	1800	-	-	75.9	26.9	-64.6
p30	30	150	92	0.2	-	0.1	0.1	-14.3
p50	50	250	267	3.9	-	0.2	0.2	11.8
p80	80	440	543	73	-	1.3	0.8	-38.4
p100	100	532	1085	291.2	-	3.1	2	-35.5
p150	150	850	2239	-	-	88.6	43.6	-50.7

The computational tests clearly show the effectiveness of method proposed in this work to accelerate column generation. In fact, for all groups of instances that took more than two seconds to be solved by standard column generation, the relative improvement is always greater than 35% and frequently greater than 60%.

The best relative improvements were achieved in the larger instances, and, in particular, in the instances with a large number of commodities defined in dense networks.

5 Conclusions

In this work, we presented a way of accelerating column generation for the linear multicommodity flow problem in planar graphs. The method used is based on a new model, which includes a polynomial number of extra variables corresponding to flows on circuits.

Computational tests for three different sets of instances showed the efficiency of this new approach.

The presented approach poses no theoretical difficulties when applied to related multicommodity flow problems. In particular, it can be used in instances of the same problem with multiple origins and destinations for each commodity, or in the multicommodity maximal flow problem. Future work will include the exploration of its use has a component of a branch-and-price algorithm. Another natural development of this work is to apply the same approach on multicommodity problems in general networks. The main question that arises is how to control the (exponential) number of extra variables, or, saying it in another way, how to select an effective subset of extra variables such they do not make the master problem too large.

Acknowledgements

We thank Carina Pimentel for programming the random instance generator used and for the realization of the computational tests. We thank Professor Antonio Frangioni for sharing with us his bundle code (and for the support given on running it) allowing the comparison that was made. We also thank his valuable comments that helped us improving this work. This work was partially supported by Fundação para a Ciência e Tecnologia (Projecto POSI /1999 /SRI/35568) and Centro Algoritmi da Universidade do Minho.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, "Network Flows". Prentice Hall, 1993.
[2] F. Alvelos, J.M.V.d. Carvalho, "Aplicação do método de partição e geração de colunas ao problema do fluxo multicomodidade", Investigação Operacional, 21, 71-92, 2001.
[3] F. Alvelos, J.M.V.d. Carvalho, "Accelerating column generation for planar multicommodity flow problems", Universidade do Minho, Cadernos do DPS-08/2003, 2003.
[4] A.A. Assad, "Multicommodity network flows - A survey", Networks, 8, 37-91, 1978.
[5] J.M.V.d. Carvalho, "Using extra dual cuts to accelerate column generation", to appear in INFORMS Journal on Computing, 2000.
[6] J. Castro, "A specialized interior-point algorithm for multicommodity network flows", SIAM Journal on Optimization, 10, 852-877, 2000.
[7] L.R. Ford, D.R. Fulkerson, "A suggested computation for maximal multicommodity network flows", Management Science, 5, 97-101, 1958.
[8] A. Frangioni, "About lagrangian methods in integer optimization", Università di Pisa, Dipartimento di Informatica, 2004.
[9] A. Frangioni, G. Gallo, "A bundle type dual-ascent approach to linear multicommodity min cost flow problems", INFORMS Journal on Computing, 11, 370-393, 1999.
[10] J.L. Goffin, J. Gondzio, R. Sarkissian, J.P. Vial, "Solving nonlinear multicommodity flow problems by the analytic center cutting plane method", Mathematical Programming, 76, 131-154, 1996.
[11] ILOG, "CPLEX 6.5, User's Manual". 1999.

[12] J.L. Kennington, "Survey of linear cost multicommodity network flows", Operations Research, 26, 209-236, 1978.

[13] T. Larsson, D. Yuan, "An augmented lagrangian algorithm for large scale multicommodity routing", Department of Mathematics, Linköpings Universitet, Research report, 2001.

[14] C. Lemaréchal, "The omnipresence of Lagrange", 4OR Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1, 7-25, 2003.

[15] M.E. Lübbecke, J. Desrosiers, "Selected topics in column generation", Les Cahiers de GERAD G-2002-64, 2002.

[16] R.E. Marsten, W.W. Hogan, J.W. Blankenship, "The boxstep method for large-scale optimization", Operations Research, 23, 389-405, 1975.

[17] K. Mehlhorn, S. Näher, "LEDA - A platform for combinatorial and geometric computing". Cambridge University Press, 1999.

[18] W. Ouaja, B. Richards, "A hybrid multicommodity routing algorithm for traffic engineering", Networks, 43, 125-140, 2004.

[19] G.L. Schultz, R.R. Meyer, "An interior point method for block angular optimization", SIAM Journal on Optimization, 1, 583-602, 1991.