Using Branch-and-Price to Solve Multicommodity k-Splittable Flow Problems

Jérôme Truffot[1], Christophe Duhamel[1], Philippe Mahey[1]
[1]Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes
Address: UMR 6158 - CNRS, Université Blaise Pascal, Clermont-Ferrand, France
Email: {jerome.truffot,christophe.duhamel,philippe.mahey}@isima.fr

Abstract

In this paper we present a column generation model and a branch-and-price algorithm for a multicommodity k-splittable flow problem. The k-splittable flow is an extension of the unsplittable flow. It has been introduced by Baier, Köhler and Skutella in [4]. This flow can be split into at most k paths. The k-splittable flow is used to study path limitation constraints in classical flow problems. In this paper we consider only the maximal k-splittable flow problem. First we formulate this problem with two sets of variables for each one of the k paths: path design variables and flow variables. This leads to a large linear mixed integer program with two types of variables. The subproblem is not exactly an unsplittable flow problem since the amount of flow is not known. This degree of freedom complexifies this problem. Moreover the flow conservation constraint alone does not ensure that the decision variables define a single path since any set of directed cycles satisfies this constraint. So an additional constraint is put on each nodes cocycle. It allows only sets of disjoint directed cycles.

Next we present a path decomposition on this model, which gives us an exponential edge-path model. Then a Branch-and-Price algorithm is applied. The branching strategy is based on the work of Barnhart and al. for the integer multicommodity flow problem [5]. The column generation subproblem can be seen as a shortest path problem with "high" enough capacity. We propose a polynomial time algorithm to solve this subproblem. Computational results are improved by introducing strategies like variable ordering and storing paths in a pool. Finally numerical results are reported for the different strategies and with a direct MIP approach.

keywords: MPLS, mixed integer programming, Branch-and-Price, k-splittable flow

1 Introduction

Multiprotocol Label Switching (MPLS) is an efficient answer to the rise of Internet telecommunications. The main idea is to gather several IP packets and give them the same label. This aims at reducing the routing tables and improving the quality of service. On another hand the running cost increases with the number of LSP (*Label Switched Path*). Thus the operator must try to limit the number of paths to route each demand. This limitation on the number of paths is a recent constraint and it seems to introduce a big change on classical flow problems.

In order to study this new constraint, we add it to the classical maximal flow problem. This problem was first studied by Baier and al. in [4]. They introduced the notion of k-splittable flow problem (k-SFP): a k-splittable flow is a flow which can be splitted into at most k paths. This problem is proven to be NP-Hard even in the single commodity case and they propose approximation algorithms.

More recently, Martens and Skutella [9] present approximation algorithms and lower bounds for the minimal cost k-splittable flow problem. It can be seen as an extension of the unsplittable flow problem [7]. A quite similar problem is the k-disjoint flow problem introduced by Bagchi and al. in [3]. It can be seen as an extension of the edge-disjoint path problem [10].

In this paper, we present an exact algorithm approach for this problem. We use a Branch-and-Price algorithm using the seminal work of Barnhart and al. on unsplittable flow problem [5] (namely the adaptated branching rule for multicommodity flow problems). The Branch-and-Bound method was introduced by Land and Doig in [8] and Branch-and-Price by Barnhart and al. in [6]. A Branch-and-Price algorithm with another branching rule was proposed by Alvelos and de Carvalho in [2] for unsplittable flow problems too.

2 Models

The problem is defined over a capacitated digraph $G = (V, E)$ in which a maximal flow has to be routed for each commodity $k \in K$, without exceeding the arc capacities u_e, $e \in E$. Furthermore, the flow for each commodity k must be routed using less than H_k paths. For each h^{th} path of commodity k and for each edge e, we define a flow variable x_e^{hk} and a decision flow support variable y_e^{hk}. To simplify notations, a backward edge e_k is introduced with infinite capacity for each commodity. Therefore, maximisation is performed on the flow variables $x_{e_k}^{hk}$ of this backward edge. Moreover the flow constraints use the edge-node incidence matrix A which includes this backward edge.

For the unsplittable flow, flow constraints and binary variables are sufficient to guarantee a single path from the source to the target of the commodity. In fact, these constraints allow a path plus any set of directed cycles that do not improve the amount of flow for this problem. For the k-splittable flow these constraints are not sufficient. The relative freedom between flow and flow support complexifies the model. Indeed the flow can use a directed cycle of flow support to split and to route flow on more than one path. Figure 1 illustrates this problem: the couple of values on each arc are the flow and the flow support variables. The flow support cycle is shown by dashed edges.

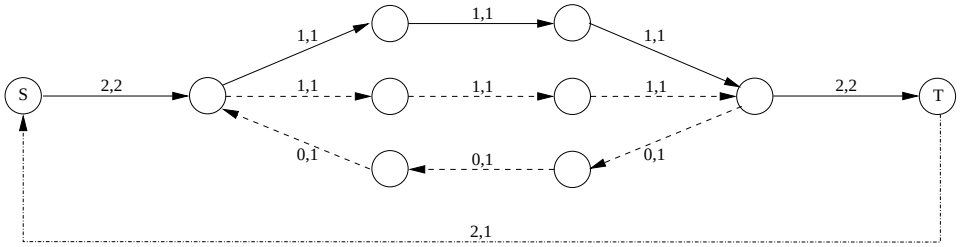

Figure 1: Flow split on a directed cycle.

To prevent such situation a constraint on incoming cocycle $\omega^-(v)$ of nodes $v \in V$ is introduced. Limitation on at most one incoming arc for each node forces directed cycles to be disjoint for the s,t-path. Finally, our problem is as follows:

$$
(LP1)\begin{cases}
\max \displaystyle\sum_{k\in K}\sum_{h=1}^{H_k} x_{e_k}^{hk} & & & & \\
s.t. & & & & \\
\quad Ax^{hk} = 0 & \forall k \in K & \forall h = 1 \dots H_k & & (a)\\
\quad Ay^{hk} = 0 & \forall k \in K & \forall h = 1 \dots H_k & & (b)\\
\quad \displaystyle\sum_{k\in K}\sum_{h=1}^{H_k} x_e^{hk} \le u_e & \forall e \in E & & & (c)\\
\quad x_e^{hk} - u_e y_e^{hk} \le 0 & \forall k \in K & \forall h = 1 \dots H_k & \forall e \in E & (d)\\
\quad \displaystyle\sum_{e\in\omega^-(v)} y_e^{hk} \le 1 & \forall k \in K & \forall h = 1 \dots H_k & \forall v \in V & (e)\\
\quad x_e^{hk} \ge 0 & \forall k \in K & \forall h = 1 \dots H_k & \forall e \in E & (f)\\
\quad y_e^{hk} \in \{0,1\} & \forall k \in K & \forall h = 1 \dots H_k & \forall e \in E & (g)
\end{cases} \tag{1}
$$

Another way to model this problem is to use the flow path decomposition. For each commodity k a set P^k of elementary paths from source to target is defined. For each path p of P^k we note $u_p = \max_{e\in p} u_e$ the path's capacity and $\delta_e^p = 1$ if the path p uses edge e and 0 otherwise. For each h^{th} path of commodity k, we define the flow variable x_p^{hk} and the flow support variable y_p^{hk}. Then the problem can be stated in the following extended formulation:

$$
(LP2)\begin{cases}
\max \sum_{k\in K}\sum_{H=1}^{H_k}\sum_{p\in P^k} x_p^{hk} & & & & \\
s.t. & & & & \\
\quad \sum_{k\in K}\sum_{h=1}^{H_k}\sum_{p\in P^k} \delta_e^p x_p^{hk} \le u_e & \forall e\in E & & & (a)\\
\quad x_p^{hk} - u_p y_p^{hk} \le 0 & \forall k\in K & \forall h=1\dots H_k & \forall p\in P^k & (b)\\
\quad \sum_{p\in P^k} y_p^{hk} \le 1 & \forall k\in K & \forall h=1\dots H_k & & (c)\\
\quad x_p^{hk} \ge 0 & \forall k\in K & \forall h=1\dots H_k & \forall p\in P^k & (d)\\
\quad y_p^{hk} \in \{0,1\} & \forall k\in K & \forall h=1\dots H_k & \forall p\in P^k & (e)
\end{cases} \tag{2}
$$

The number of paths in set P^k grows exponentially with the size of the graph. Thus this model can not be solved directly with a general MIP solver and a dedicated Branch-and-Price algorithm is proposed in the next section.

3 Branch-and-Price Algorithm

Branch-and-Price is an efficient strategy to solve hard combinatorial problems, especially when the number of variables is exponential. As shown by Barnhart and al. [5, 6], the branching scheme is a key point to the efficiency. Then we need an efficient branching rule to build the decision tree. On a first hand, we look for a rule that balances the decision tree, like Ryan and Foster branching rule [11]. On the other hand, we need the rule to keep a fast column generation. So we have chosen the branching rule proposed by Barnhart and al. in [5]. After solving the linear relaxation, we use the edge-node formulation to find the *first divergence node* d of a h^{th} path of a commodity k. This node is defined by the two next conditions: first there is a integer path from the source to d. Second there are at least two edges in the outgoing cocycle $\omega^+(d)$ of d, and the corresponding flow support variables are fractional. We note a and b those two edges. Then $\omega^+(d)$ is split into two balanced sets $\omega^+(d,a)$ and $\omega^+(d,b)$ that include respectively arc a and b. The branching rule is defined by equation 3:

$$
\left(\sum_{e\in\omega^+(d,a)} y_e^{hk} = 0\right) \quad \text{vs.} \quad \left(\sum_{e\in\omega^+(d,b)} y_e^{hk} = 0\right) \tag{3}
$$

The relaxation of (LP2) is solved by column generation. The optimal solution may be fractional. So we apply the branching rule but it works on the arc-node model. Then this branching rule is projected on the arc-path space. First notice that if flow support variables are fractional, then there are usually a lot of optimal solutions and some of them satisfy equation 4:

$$
y_p^{hk} = \frac{x_p^{hk}}{u_p} \qquad \forall k\in K \quad \forall h=1\dots H^k \quad \forall p\in P^k \tag{4}
$$

Then the relaxed (LP2) can be reduced by substituting these y_p^{hk} by $\frac{x_p^{hk}}{u_p}$. We obtain the linear program (LP3).

$$
(LP3)\begin{cases}
\max \sum_{k\in K}\sum_{h=1}^{H^k}\sum_{p\in P^k} x_p^{hk} & & & & \\
s.t. & & & & \\
\quad \sum_{k\in K}\sum_{h=1}^{H^k}\sum_{p\in P^k} \delta_e^p x_p^{hk} \le u_e & \forall e\in E & & & (a)\\
\quad \sum_{p\in P^k} \frac{x_p^{hk}}{u_p} \le 1 & \forall k\in K & \forall h=1\dots H^k & & (b)\\
\quad x_p^{hk} \ge 0 & \forall k\in K & \forall h=1\dots H^k & \forall p\in P^k & (c)
\end{cases} \tag{5}
$$

The column generation subproblem (LP4) works with dual variables π_e and λ^{hk} corresponding to constraints (a) and (b). It is defined as follows:

$$(LP4)\begin{cases} \min \sum_{e\in E} u_e\pi_e + \sum_{k\in K}\sum_{h=1}^{H^k}\lambda^{hk} & & & & \\ s.t. & & & & \\ \quad \sum_{e\in E}\delta_e^p\pi_e + \dfrac{\lambda^{hk}}{u_p} \geq 1 & \forall k\in K & \forall h = 1\dots H^k & \forall p\in P^i & (a) \\ \quad u_e \geq 0 & \forall e\in E & & & (b) \\ \quad \lambda^{hk}\geq 0 & \forall k\in K & \forall h = 1\dots H^k & & (c) \end{cases} \tag{6}$$

So the reduced cost c_p^{hk} of path p in P^k for each h^{th} path of commodity k is defined by:

$$c_p^{hk} = 1 - \sum_{e\in E}\delta_e^p\pi_e - \frac{\lambda^{hk}}{u_p} \tag{7}$$

The new column will be the path with the highest reduced cost. It is a compromise between shortest path and highest capacity path. In order to find it, we use a modified version of Dijkstra algorithm [1, pp 108–113] to compute the highest capacity path among the shortest path. Moreover we add a lower capacity $\underline{u}$ constraint on this path. We start with $\underline{u} = 0$ and $\underline{u}$ is iteratively increased to find another shortest path which satisfies this capacity and such that its reduced cost is maximal. We prove that this algorithm is polynomial and solves our subproblem.

Some other ideas may allow us to further speed up the path generation procedure. The first one is to build a pool of paths for the decision tree. Indeed many paths are generated for a lot of nodes. The pool allows to use previously generated paths for the remaining decision nodes. Thus a lot of path generations are avoided (at the cost of bigger problems at each node) and the speed of the whole optimization procedure is increased. On the other hand, symmetry in the linear programs is known to increase the computational time. Then we chose to apply a *variable ordering* method and to break this symmetry with constraint 8.

$$\sum_{p\in P^k} x_p^{(h+1)k} - \sum_{p\in P^k} x_p^{hk} \leq 0 \qquad \forall k\in K \quad \forall h = 1\dots H^k - 1 \tag{8}$$

4 Computational Results

For the testing, the instances have been randomly generated using a procedure which guarantees the existence of a flow between source and target of each commodity. In this paper we propose some results for the single commodity case. Multicommodity case will be tested in the future. Tests were performed on an Intel Xeon 2Ghz processor, 2GB of RAM, running Gentoo linux distibution.

CPU times in seconds and number of visited nodes of the decision tree are reported in table 1. We use the following notation: 'C' stands for the cplex mixed integer solver on (LP1), 'BB-V' for the Branch-and-Bound and variable ordering, 'BP' for the Branch-and-Price on (LP2), 'BP-V' for Branch-and-Price with variable ordering on (LP2) and 'BP-VP' for Branch-and-Price with variable ordering and paths pool management on (LP2). For each instance H is the maximum number of paths, z^* is the optimal solution.

Graph	H	z^*	Number of nodes				CPU Time (s)				
			BB-V	BP	BP-V	BP-VP	C	BB-V	BP	BP-V	BP-VP
5-70	1	88	1	1	1	1	0.01	0.01	0.01	0.00	0.00
	2	170	1	1	1	1	0.02	0.02	0.01	0.00	0.01
	3	234	63	6	31	9	0.03	0.22	0.02	0.05	0.01
	4	291	118	90	49	38	0.28	0.26	0.22	0.14	0.03
	5	334	6895	20243	1889	1541	4.98	38.02	58.03	6.57	1.46
	6	375	32547	476461	5299	5454	1459.91	396.33	1704.37	24.35	7.40
	7	412	112401	-	21639	21192	-	1317.92	-	125.31	43.66
	8	447	-	-	46981	51813	-	-	-	322.94	156.82
	9	481	-	-	163096	97275	-	-	-	1391.98	396.22
10-80	1	63	1	1	1	1	0.01	0.01	0.00	0.00	0.00
	2	118	78	2	2	5	0.36	0.24	0.01	0.00	0.01
	3	164	5185	49	22	20	6.22	23.55	0.07	0.04	0.01
	4	199	103553	3697	321	169	2198.27	749.10	6.85	0.73	0.10
	5	232	-	910205	4196	1920	-	-	2239.20	12.14	1.71

Table 1: Comparing CPU Times.

The paths pool strategy clearly improves the running time and reduces the number of nodes in the tree. In table 2, SP_0 and COL_0 are the number of calls to Dijkstra's algorithm and the number of paths at the root node. SP and COL are the average number of calls to Dijkstra's algorithm and paths generated at each node (except the root node). The results are reported for 'BP-V' and 'BP-VP' strategies.

Graph	H	z^*	SP_0	COL_0	SP		COL	
					BP-V	BP-VP	BP-V	BP-VP
5-70	1	88	8	2	0.00	0.00	0.00	0.00
	2	170	42	12	0.00	0.00	0.00	0.00
	3	234	102	27	89.48	20.78	18.06	1.22
	4	291	332	72	195.18	33.45	33.86	1.16
	5	334	735	125	289.62	27.72	37.42	0.18
	6	375	996	168	435.31	32.54	53.90	0.08
	7	412	1400	231	579.41	38.14	68.54	0.03
	8	447	1872	280	713.37	44.06	81.77	0.02
	9	481	2403	360	926.90	51.72	101.52	0.01
10-40	1	63	8	2	0.00	0.00	0.00	0.00
	2	118	48	12	5.00	12.00	1.00	1.00
	3	164	123	30	73.00	16.95	14.82	1.10
	4	199	312	68	145.62	20.02	22.96	0.40
	5	232	465	95	199.04	20.84	30.36	0.18

Table 2: Comparing column generation.

We can notice that the pool strategy leads to less generated path.

5 Conclusion

Our Branch-and-Price algorithm gives interesting results and is probably one of the most efficient way to solve problems with a constraint on the number of paths. The main difficulty is to control the computational time. For many instances, time varies from few seconds on k paths to several hours on $(k+1)$ paths. On the other hand, we are going to compare Barnhart and al. branching rule with others like Alvelos and de Carvalho branching rule [2].

The maximal flow is a good problem to study this number of paths limitation constraint but it has a few direct applications in telecom networks. So we want to apply our Branch-and-Price method to other problems. First we want to study the minimal cost k-splittable flow problem. The formulation of this problem and the column generation model seems to be pretty similar to the one presented here. Next we want to study the minimization of number of LSP in MPLS routing problems. Finally we may search approximation algorithms for these problems.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows - Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Filipe Alvelos and Jos Manuel Vasconcelos Valrio de Carvalho. Comparing Branch-and-price Algorithms for the Unsplittable Multicommodity Flow Problem. In *Proceedings of the International Network Optimization Conference*, pages 7–12, 2003.

[3] Amitabha Bagchi, Amitabh Chaudhary, Christian Scheideler, and Petr Kolman. Algorithms for fault-tolerant routing in circuit switched networks. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 265–274. ACM Press, 2002.

[4] Georg Baier, Ekkehard Köhler, and Martin Skutella. On the k-splittable flow problem. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 101–113. Springer-Verlag, 2002.

[5] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

[6] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W.P. Savelsbergh, and Pamela H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[7] Jon M. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1996.

[8] A. Land and A. Doig. An automatic method of solving discrete programming problems. *Econometrika*, 28(3):497–520, 1960.

[9] Maren Martens and Martin Skutella. Flows on Few Paths: Algorithms and Lower Bounds. In *proceedings of ESA 2004*, pages 520–531, 2004.

[10] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:97–115, 1927.

[11] David M. Ryan and Brian A. Foster. An integer programming approach to scheduling. In Anthony Wren, editor, *Computer Scheduling of Public Transport : Urban Passenger Vehicle and Crew Scheduling*, pages 269–280. North-Holland Publishing Company, 1981.