in Proceedings of the
"International Network
Optimization Conference 2007",
Spa, Belgium, April 22-25, 2007,
(with refereeing)

A local search heuristic based on column generation applied to the binary multicommodity flow problem

Filipe Alvelos, *Departamento de Produção e Sistemas, Universidade do Minho, Portugal*
J. M. Valério de Carvalho, *Departamento de Produção e Sistemas, Universidade do Minho, Portugal*

Abstract

In this paper, we apply a multi-start local search heuristic based on column generation to the binary multicommodity flow problem. One solution is represented as a set of paths, one of each commodity. A neighbor solution is obtained by selecting a different path for one commodity. We discuss and present computational results for alternative ways of using primal and dual information from the linear programming (restricted) master problem in constructing and evaluating heuristic integer solutions.
The local search heuristic, although described and tested for the binary multicommodity problem, can be applied in general column generation models where the master variables are binary and there are convexity constraints, since it does not depend on the subproblem structure.

Keywords: local search, column generation, multicommodity flows.

1 Introduction

In this paper, we introduce a heuristic procedure for obtaining integer solutions in column generation models. The heuristic uses primal and dual information from the optimal solution of a restricted master problem (RMP) in order to search the solution space and can be easily incorporated in branch-and-price-and-cut algorithms. For a recent survey about column generation, see [8]. Recent surveys on local search and on metaheuristics can be found in [6, 11], respectively.
The heuristic was tested in the binary multicommodity flow problem (MFP). Other designations for the binary MFP are: non-bifurcated routing problem, traffic placement problem, single path routing problem, path selection problem or multiple source unsplittable MFP. Most of those designations refer to communication problems. Other applications, such as production planning and distribution / transportation may also be considered, whenever a multicommodity flow model is used and the commodities cannot be split. An introduction to MFPs can be found in [1].
Heuristics [5] and exact methods [2, 3, 4, 9] have been proposed for the binary MFP. We use the model where the decision variables are associated with paths as a starting point for defining the components of a multi-start local search heuristic (LSH). Following the classification of combinations of exact and heuristic methods of [10], our present method can be seen as a collaborative sequential method.

2 Column Generation and Branch-and-Price-and-Cut

The binary MFP is defined in a network with a set of n nodes, represented by N, and a set of m arcs, represented by A. We use an index $i=\{1,...,n\}$ to represent a node and a pair of indices ij to represent an arc which has origin in node i and destination in node j. We define a set K of h commodities, indexed by k. Each commodity k has an origin, o^k, a destination, d^k, and an integer demand, r^k, which is the number of units supplied at its origin and required at its destination. We also define an integer capacity u_{ij} associated with each arc of the network and a unit cost, c_{ij}^k, associated with the flow of commodity k on arc ij. We make the usual assumption, $c_{ij}^k \geq 0, \ \forall ij \in A, \ \forall k \in K$.

We denote the set of simple paths between the origin and the destination of commodity k by P^k. The parameter y_{ij}^{pk}, $\forall ij \in A$, $\forall k \in K$, $\forall p \in P^k$, assumes value *1* if arc ij belongs to the path p of commodity k, and *0* otherwise. The cost of one path, c^{pk}, is given by $c^{pk} = \sum_{ij \in A} y_{ij}^{pk} c_{ij}^k$. Finally, we define the decision variable associated with each path as λ^{pk}, $\forall k \in K$, $\forall p \in P^k$. The path model of the binary MFP is:

$$Min \sum_{k \in K} \sum_{p \in P^k} c^{pk} r^k \lambda^{pk}$$

$$s.t:$$

$$\sum_{p \in P^k} \lambda^{pk} = 1, \forall k \in K \qquad (1)$$

$$\sum_{k \in K} \sum_{p \in P^k} y_{ij}^{pk} r^k \lambda^{pk} \leq u_{ij}, \forall ij \in A \qquad (2)$$

$$\lambda^{pk} \in \{0,1\}, \forall k \in K, \forall p \in P^k .$$

Typically, when solving path models of multicommodity flow problems, a column generation approach is used, since the number of variables is extremely large. A detailed description of this method applied to the fractional multicommodity flow problem is given in [1]. In order to obtain binary solutions, a combination of column generation, the use of cuts, and branch-and-bound (branch-and-price-and-cut) was first developed in [3]. A branch-and-price-and-cut (BPC) algorithm with a different branching strategy, where the branching constraints are kept in the master problem and their duals are integrated in the objective function of the subproblem(s), is described in [2]. In this way, solving the problem associated with one node of the search tree is similar to solving the problem of the root node, in the sense that the subproblem structure is the same. The same happens if cuts are introduced in the master problem, i.e., the subproblem structure does not change. The starting point of the heuristic, presented in the next Section, is any RMP (possibly including branching constraints and cuts) along with an optimal fractional primal solution and an optimal dual solution.

In the application of the heuristic to the binary MCF, we used the RMP of the root node. In this case, the optimised RMP provides a set of paths, their weights (optimal fractional primal solution) and the dual solution associated with the constraints (including lifted cover inequalities).

3 Local Search based on Column Generation

A solution of the LSH is represented by a set of h paths, one for each commodity. A solution may be infeasible, since capacity constraints (2) may be violated. A neighbor solution is obtained by selecting a different path for one commodity. With this neighborhood structure, any solution of the search space can be reached starting in any other one. Denoting by Q^k the set of paths of commodity k included in the RMP, the

number of solutions considered by the LSH is $\prod_{k=1}^{h}\left|Q^k\right|$ and the size of the neighborhood is $\sum_{k=1}^{h}\left(\left|Q^k\right|-1\right)$.

The sets of solutions involved in the LSH are illustrated in Figure 1: S'_M is the relaxed solution (it may violate the capacity constraints) space of the (full) master problem; S'_{RMP} is the relaxed solution space of the RMP; S_M is the space of feasible solutions (includes the optimal solution); and S_{RMP} is the space of feasible solutions of the RMP. Note that, if one path that belongs to the optimal solution is not included in the RMP, the optimal solution may not be found in the space searched. Also note that the search space (S'_{RMP}) may not include any feasible solution when $S'_{RMP} \cap S_{RMP} = \varnothing$.

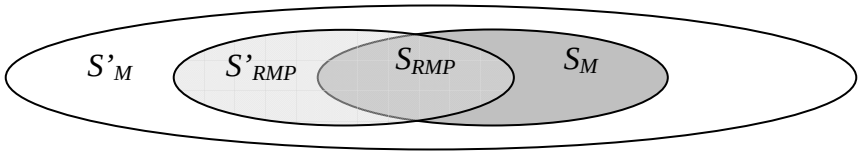

Figure 1 Illustration of the relation between solution spaces.

In order to construct the initial solution, we considered three alternatives. In the first one, for each commodity, the path with largest weight is selected. In the second and third alternatives, one path of each commodity is selected randomly. In the second alternative the weight of a path is taken as the probability of its selection. In the third alternative, the probabilities for a given commodity are uniform. We note that in the first two alternatives, primal information from the RMP is used, and that, in the first alternative, since the procedure is deterministic, the heuristic solution generated is always the same. In preliminary computational tests, the second alternative clearly gave the best results. As an example, for one instance (*bl11*), with four different evaluation functions and ten runs each one (each run with 20 starts), the second alternative always obtained a feasible solution, while the first and third alternatives did not obtain any.

We separate the evaluation of a solution in two components: one value associated with the cost of the paths (feasibility value) and one value associated with the violation of the capacity constraints (infeasibility value). When comparing two (in)feasible solutions, only the (in)feasibility values are used. When comparing one feasible solution with one infeasible solution, the former is always better.

For simplicity of notation, in the discussion that follows, we consider that there are neither branching constraints nor cuts in the RMP.

We considered two alternatives for the feasibility values: original costs and modified costs. In the first alternative, the feasibility value of one solution is obtained by summing up the original costs of its paths. In the second alternative, the feasibility value of one solution is obtained by summing up the modified costs of its paths, where the modified cost of a path is given by $\bar{c}^{pk} = r^k \sum_{ij \in A}\left(c_{ij}^k + w_{ij}\right)y_{ij}^k$, where w_{ij} is the (nonnegative) dual variable associated with the capacity constraint (2) of arc *ij*. For a basic path *p* of commodity *k*, $\bar{c}^{pk} = \pi^k$, and π^k is the dual variable associated with the convexity constraint (1) of commodity *k*. For a non basic path *q* of the same commodity *k*,

$\bar{c}^{qk} \geq \pi^k$. Using modified costs allows the implicit consideration of the capacity constraints.

We also considered two alternatives to calculate the infeasibility value of a solution. In the first one, the infeasibility value is given by $\sum_{ij \in A} Max\left\{0, \sum_{k \in K} y_{ij}^{p'k} r^k - u_{ij}\right\}$, where *p'* is the path of commodity *k* in the solution. In second alternative, the values associated with the dual variables of the capacity constraints are used to weight the violation of the different constraints. For each arc $ij \in A$, we define $\overline{w}_{ij} = w^{max}$, if $w_{ij} = 0$ and $\overline{w}_{ij} = w_{ij}$, if $w_{ij} > 0$, where w^{max} is the largest value of a dual variable w_{ij}. The infeasibility value is given by $\sum_{ij \in A} Max\left\{0, \overline{w}_{ij}\left(\sum_{k \in K} y_{ij}^{p'k} r^k - u_{ij}\right)\right\}$. We note that this second alternative is commonly used (for example, in [11]), but with the $\overline{w}_{ij}$ values not related to the duals.

In the next section, we present computational results for the four combinations of these alternatives.

4 Computational Results

All the reported results were obtained on a personal computer with a Pentium 4, 2.80 GHz processor, 1 GB of RAM, running Windows XP Professional Edition. We used the largest *32 Carbin* test instances (available at *www.dps.uminho.pt/pessoais/falvelos/*). We also tested one instance (*planar50*) from [7] (obtained at *www.di.unipi.it/~frangio/*).

In Table 1, the number of arcs (column *m*) and the number of commodities (column *h*) for each instance are given. All *Carbin* instances have *32* nodes and the *planar50* has *50* nodes. The difference between each group of four consecutive *Carbin* instances lies in the cost structure (the arc costs vary by commodity – second and fourth instances – or not – first and third instances, and their values are between 1 and 1000 – first and second instances – or between 1 and 10 – third and fourth instances). The BPC algorithm is based on the branching rule introduced in [2], and uses general lifted cover inequalities (LCIs), a depth search strategy and columns with reduced cost greater than the duality gap are removed every five iterations. We implemented the L–2queue algorithm to solve shortest path (sub)problems, and used the COIN (*www.coin-or.org/*) implementation of the Horowitz-Sahni algorithm to solve the knapsack problems when constructing LCIs. We used the Cplex 8.1 (*www.ilog.com*) dual simplex algorithm to solve the RMPs. The LSH is executed after the optimal solution of the BPC root node is obtained (thus, the RMP may include cuts, but no branching constraints) with 200 starts.

We first compare the LSH alternatives. Alternative ND provided the best results: feasible integer solutions for 20 out of the 33 instances, and the best heuristic solution 14 times. The second best was alternative DN (20 feasible solutions, 4 best heuristic solutions; in two of them, it was the only alternative that obtained feasible solutions).

We now compare the quality of the solutions of the best LSH alternative with BPC, noting the very significant difference between the execution times. The number of integer feasible solutions found was approximately the same with both algorithms (20 vs. 21). The LSH provided integer solutions better than the obtained by the BPC algorithm (either the optimal solution or the incumbent solution) in one hour in 10 out of 33 instances (the

opposite happened 16 times). For five of the largest instances (*bl18*, *bl19*, *bs17*, *bs18* and *bs21*), the results were even more impressive: the local search heuristic found one feasible solution in a few minutes, while BPC was not able to do so in one hour.

Instance	*m*	*h*	**RT**	**BPC**		**LSH**							
						NN		**DN**		**ND**		**DD**	
				V	T	V	T	V	T	V	T	V	T
bl09	96	192	2.8	**	**	***	42	***	43	***	39	***	39
bl10	96	192	1.3	**	**	***	41	***	41	***	44	***	44
bl11	96	192	0.7	69018	470	69141	8	69151	8	69143	8	69145	8
bl12	96	192	1.5	66019	*	***	3	67244	29	***	31	***	30
bl13	320	192	7.5	3155673	*	3143701	44	3142417	43	3144458	48	3144687	48
bl14	320	192	2.3	2433011	301	2452994	41	2456951	42	2452720	46	2453219	47
bl15	320	192	2.9	34274	316	34362	24	34345	23	34302	24	34302	24
bl16	320	192	1.8	28074	37	***	25	***	24	***	25	***	24
bl17	96	320	3.0	13324233	*	13343031	184	13297281	180	13274926	189	13317324	202
bl18	96	320	5.0	**	**	10561044	178	10551262	181	10540177	199	10546784	204
bl19	96	320	6.2	**	**	110059	238	110410	235	110350	253	110571	241
bl20	96	320	4.5	**	**	***	218	***	235	***	235	***	229
bl21	320	320	16.7	5837994	*	5843286	202	5854770	205	5828677	218	5847143	209
bl22	320	320	10.0	4217172	*	4244190	184	4260610	180	4238529	191	4254038	187
bl23	320	320	11.6	56970	*	57669	159	57684	156	57312	171	57525	170
bl24	320	320	6.5	51081	*	48164	143	48103	140	48316	146	48231	148
bs09	96	192	1.7	6308373	*	6331112	45	***	41	6364750	46	6369392	48
bs10	96	192	2.0	**	**	***	45	7255730	45	***	48	***	49
bs11	96	192	4.9	**	**	***	66	***	60	***	65	***	66
bs12	96	192	2.1	**	**	***	34	***	34	***	36	***	35
bs13	320	192	8.4	3615375	*	***	83	3716193	83	3654204	89	3729077	85
bs14	320	192	12.5	3181860	*	***	153	***	154	***	158	***	159
bs15	320	192	3.7	38533	117	38755	30	38764	30	38695	30	39022	29
bs16	320	192	3.3	31124	335	***	63	***	64	***	63	***	65
bs17	96	320	4.9	**	**	11559349	148	11555234	145	11548436	158	11564789	162
bs18	96	320	4.3	**	**	***	202	***	200	10603149	215	***	210
bs19	96	320	5.9	106369	*	106770	100	106542	106	106600	113	106433	112
bs20	96	320	4.6	**	**	***	184	***	184	***	197	108679	205
bs21	320	320	64.0	**	**	5653280	284	5650572	281	5645452	298	5665752	299
bs22	320	320	36.5	4796079	*	***	501	***	506	***	542	***	541
bs23	320	320	30.9	57821	*	58273	210	58656	210	58194	234	58396	224
bs24	320	320	15.9	51045	*	***	277	***	274	***	289	***	291
planar50	250	267	1.0	123226335	*	122543523	203	122707247	140	122514873	189	122687405	139

Table 1 Computational results.

* An optimal solution was not obtained within one hour; the corresponding V is the value of the incumbent integer solution. ** A feasible integer solution was not obtained within one hour. *** The heuristic did not find a feasible solution. RT – Root CPU time including the generation of cuts (in seconds); BPC – Branch-and-price-and-cut; V – Value of the solution; T – CPU time excluding the root time (in seconds); LSH – Local search heuristic(s); NN – no dual information is used in evaluating solutions; DN – dual information is used in evaluating *feasible* solutions; ND – dual information is used in evaluating *infeasible* solutions; DD – dual information is used in evaluating feasible and infeasible solutions.

5 Conclusions

In this paper we presented a multi-start local search heuristic (LSH) that can be applied in column generation models with binary variables and convexity constraints.
The LSH uses primal and dual information from a restricted master problem when building an initial solution and when evaluating solutions, respectively.
When applied to the binary multicommodity flow problem, the local search heuristic yielded solution of quality similar to those of a branch-and-price-and-cut (BPC) algorithm, but using a small fraction of time. For some instances the local search heuristic

found one feasible solution in a few minutes, while BPC did not obtain any feasible solution in one hour.
In the present application, the starting point of the LSH was the RMP of the root node of a BPC tree. A natural extension is its use in different nodes of the branch-and-price-and-cut tree.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows: Theory, algorithms, and applications, Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] F. Alvelos and J.M.V.d. Carvalho, Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem, INOC - International Network Optimization Conference, Institut National des Télécommunications, France, 2003, pp. 7-12.

[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 48 (2000), 318-326.

[4] M. Belaidouni and W. Ben-Ameur, A superadditive approach to solve the minimum cost single path routing problem: preliminary results, INOC - International Network Optimization Conference, 2003, pp. 67-71.

[5] M.-C. Costa, A. Hertz, and M. Mittaz, Bounds and heuristics for the shortest capacitated paths problem, Journal of Heuristics 8 (2002), 449-465.

[6] M. Gendreau and J.-Y. Potvin, Metaheuristics in combinatorial optimization, Annals of Operations Research 140 (2005), 189-213.

[7] T. Larsson and D. Yuan, An augmented Lagrangian algorithm for large scale multicommodity routing, Computational Optimization and Applications 27 (2004), 187-215.

[8] M.E. Lübbecke and J. Desrosiers, Selected topics in column generation, Operations Research 53 (2005), 1007-1023.

[9] S. Park, D. Kim, and K. Lee, An integer programming approach to the path selection problems, International Network Optimization Conference, 2003, pp. 448-453.

[10] J. Puchinger and G.R. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification", Lecture Notes in Computer Science, J. Mira and J.R. Álvarez (Editors), Springer, 2005, Vol. 3562, pp. 41-53.

[11] M. Yagiura and T. Ibaraki, "Local Search", Handbook of Applied Optimization, P.M. Pardalos and M.G.C. Resende (Editors), Oxford University Press, 2002, pp. 104-123.