Solving the Quorumcast Routing Problem as a Mixed Integer Program

Quoc Trung BUI[1], Quang Dung PHAM[2], and Yves DEVILLE[1]

[1] ICTEAM, Université catholique de Louvain, Belgium
{quoc.bui, Yves.Deville}@uclouvain.be
[2] SoICT, Hanoi University of Science and Technology, Vietnam
dungpq@soict.hust.edu.vn

Abstract. The quorumcast routing problem is a generalization of multicasting which arises in many distributed applications. It consists of finding a minimum cost tree that spans the source node and at least q out of m specified nodes on a given undirected weighted graph. In this paper, we solve this problem as a mixed integer program. The experimental results show that our four approaches outperform the state of the art. A sensitivity analysis is also performed on values of q and m.

1 Introduction

Multicasting is the problem of delivering a message from a source to a given subset of nodes, called the *multicast* nodes, in a network. Suppose given an undirected graph $G = (V, E, c)$, i.e., V, E are, respectively, the set of nodes and the set of edges. Suppose further that each edge $(i, j) \in E$ is associated with a positive cost $c_{ij} \in \mathbb{R}^+$. Now, given a set of multicast nodes $S \subseteq V$, an integral value $q \leq |S|$, and a root node r (without loss the generality, we may assume that $r \in S$), the objective of the quorumcast routing problem (QRP) is to find a minimum cost tree T that spans r and at least q nodes of S [5, 24, 12, 32, 29]. QRP is NP-hard, as it reduces to the Steiner tree problem [14] when $q = |S|$. QRP appears in many distributed applications, for example, distributed synchronization and updating a replicated resource (see [5] for more details).

For solving QRP, various incomplete approaches to computing an approximation of the optimal solution have been proposed in [5, 12, 32, 29], in which the constraint-based local search algorithm in [29] is currently the state of the art incomplete algorithm. In addition, two exact algorithms in [24, 29] have been proposed for solving this problem to optimality. In [24], a partial solution is defined to be a set of sub-trees that spans the root and some multicast nodes; a partial solution is extended by adding one edge at each step until a feasible solution is constructed; a *Confined Area Pruning* scheme was introduced that allows reducing that search space. The Constraint Programming (CP) approach in [29] is currently th state of the art exact algorithm.

Contributions In this paper, we propose four mathematical formulations for QRP and use them to solve QRP as a mixed integer program. These approaches outperform the state of the art approach based on CP. In addition, through the experimental results, we show the effect of the values q and $|S|$ on the performance of the proposed approaches.

2 Mathematical models

In this section, we propose four mathematical models for QRP. One is proposed directly on the undirected graph G, and the others are proposed on the corresponding directed graph of G that is formed by replacing each edge of G by two opposite arcs with the same cost as the original edge.

These models can exploit the properties of QRP solutions. Let T be a solution of $QRP(q, m)$ on a graph G. One can easily show that (1) all leaf nodes of T are multicast nodes [24], and (2) T spans exactly q multicast nodes.

All the models use the binary variables x_{ij} stating whether edge (i, j) is in the solution tree T. (In the undirected graph, we use the convention that $i < j$). All the models aim at minimizing $\sum_{(i,j)\in E} c_{ij}x_{ij}$.

2.1 Natural formulation: Model 1

In this section, we propose a formulation on the undirected graph $G = (V, E)$, called "the natural formulation." Many problems have been modeled by similar formulations [3, 27, 16, 13, 26, 1]. This model introduces binary variables y_i stating whether node i is in T.

$$\sum_{i,j\in C:(i,j)\in E} x_{ij} \leq |C| - 1, \forall C \subset V, 2 \leq |C| \leq |V| - 1 \tag{1a}$$

$$\sum_{(i,j)\in E} x_{ij} + \sum_{(j,i)\in E} x_{ji} \geq y_i, \forall i \in V \tag{1b}$$

$$\sum_{(i,j)\in E} x_{ij} + \sum_{(j,i)\in E} x_{ji} \leq (|V| - 1)y_i, \forall i \in V \tag{1c}$$

$$1 + \sum_{(i,j)\in E} x_{ij} = \sum_{v\in V} y_v \tag{1d}$$

$$y_r = 1 \tag{1e}$$

$$\sum_{v\in S} y_v = q \tag{1f}$$

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in E \tag{1g}$$

$$y_i \in \{0, 1\}, \forall i \in V \tag{1h}$$

In this model, the constraints (1a) are connectivity constraints (or subtour elimination constraints). The constraints (1b) and (1c) ensure that if $v \in T$, then $y_v = 1$, and if $v \notin T$, then $y_v = 0$. The constraint (1d) presents a basic property of a tree that requires the relation between the number of nodes and the number of edges. The constraints (1f) ensure that T includes exactly q multicast nodes (Property (2)). Finally, the constraint (1e) ensures that the root r is always in T. Notice that Property (1), stating that all leaf nodes are multicast nodes, could also be included, but experimental results have shown that it is useless here as well as in all subsequent models. It is therefore not considered.

In this model, there are $|E|+|V|$ variables and an exponential number of constraints.

2.2 Formulation based on multi-commodity flows: Model 2

In this section, we propose a multi-commodity flow formulation on the corresponding directed graph $G' = (V, A)$. In the literature, many problems have been modeled using multi-commodity flows [16, 10, 7, 17]. However, this problem is slightly more complex, as we do not know which multicast nodes are spanned.

This model introduces variables $y_{ij}^k \in \mathbb{R}^+$ measuring the flow, through arc $(i, j) \in A$, from the root node r to a node $k \in V \setminus \{r\}$.

$$\sum_{(r,i)\in A} (y_{ri}^k - y_{ir}^k) \leq 1, \forall k \in V \tag{2a}$$

$$\sum_{k\in S, k\neq r, (r,i)\in A} (y_{ri}^k - y_{ir}^k) = q - 1 \tag{2b}$$

$$\sum_{(k,i)\in A} (y_{ki}^k - y_{ik}^k) \geq -1, \forall k \in V \tag{2c}$$

$$\sum_{k\in S, k\neq r, (k,i)\in A} (y_{ki}^k - y_{ik}^k) = -(q - 1) \tag{2d}$$

$$\sum_{(j,i)\in A} (y_{ij}^k - y_{ji}^k) = 0, \forall k \in V, i \in V \setminus \{k, r\} \tag{2e}$$

$$y_{ij}^k \leq x_{ij}, \forall (i, j) \in A, \forall k \in V \cup \{r\} \tag{2f}$$

$$y_{ij}^k \geq 0, \forall (i, j) \in A, \forall k \in V \cup \{r\} \tag{2g}$$

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in V \tag{2h}$$

In this model, the constraints (2f) ensure that a flow is sent along an arc only if the arc is traversed. The constraints (2b), (2d) and (2e) ensure that there exists a flow from the root r to q nodes in S (note that we assumed that r is a multicast node). The constraints (2a), (2c) and (2e) are flow conserving constraints.

In this model, there are $|A| \times |S|$ variables and a polynomial number of constraints.

2.3 Classical formulation: Model 3

In this model, we propose a formulation, called "the classical formulation," on the corresponding directed graph $G' = (V, A)$. In the literature, many similar formulations have been proposed for problems related to finding a spanning tree [15, 22, 26, 29].

$$x_{ir} = 0, \forall (i, r) \in A \tag{3a}$$

$$\sum_{(r,i)\in A} x_{ri} \geq 1 \tag{3b}$$

$$\sum_{u\notin C, v\in C, (u,v)\in A} x_{uv} \geq \sum_{(j,i)\in A} x_{ji}, \forall C \subset V, 2 \leq |C| \leq |V| - 1, \forall i \in C, r \notin C \tag{3c}$$

$$\sum_{(i,j)\in A} x_{ij} \leq 1, \forall j \in V \tag{3d}$$

$$\sum_{i\in S, i\neq r, (j,i)\in A} x_{ji} = q - 1 \tag{3e}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{3f}$$

In this model, the constraints (3a) indicate that there are no arcs arriving at r. The constraint (3b) ensures that there exists at least one arc leaving r. The constraints (3c) are connectivity constraints (see [11, 6, 19, 22] for more details about the connectivity constraints). The constraint (3e) ensures that the optimal tree T' includes exactly q multicast nodes.

In this model, there are $|A|$ variables corresponding to the number of arcs in the corresponding directed graph G', and an exponential number of constraints in (3c).

2.4 Miller–Tucker–Zemlin formulation: Model 4

In this model, we propose a formulation using Miller–Tucker–Zemlin constraints as connectivity constraints on the corresponding directed graph $G' = (V, A)$ [28, 21].

This model introduces variables t_i constrained by $t_i \leq t_j$ if arc $(i, j) \in T$. These constraints prevent subtours in the solution. The variables p_i state whether or not node i is in T'.

$$p_r = 1 \tag{4a}$$

$$x_{ir} = 0, \forall (i,r) \in A \tag{4b}$$

$$\sum_{(r,i)\in A} x_{ri} \geq 1 \tag{4c}$$

$$\sum_{(i,j)\in A} x_{ij} = p_j, \forall j \in V \setminus \{r\} \tag{4d}$$

$$x_{ij} \leq p_i, \forall (i,j) \in A \tag{4e}$$

$$|V| x_{ij} + t_i + 1 \leq t_j + |V|, \forall (i,j) \in A \tag{4f}$$

$$1 + \sum_{(i,j)\in A} x_{ij} = \sum_{v\in V} p_v \tag{4g}$$

$$\sum_{i\in S} p_i = q \tag{4h}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A \tag{4i}$$

$$p_i \in \{0,1\}, \forall i \in V \tag{4j}$$

$$t_i \in \{1 \ldots |V|\}, \forall i \in V \tag{4k}$$

In this model, the constraints (4f) present the relative position of the nodes in the tree. They state that if $x_{ij} = 1$, then $t_i < t_j$. This prevents the solution from containing subtours. The constraints (4c) ensure that the root node r must connect to other nodes in the arborescence tree.

In this model, there are $(|A| + 2 \times |V|)$ variables and a polynomial number of constraints.

3 Solving QRP as a mixed integer program

In this section, we propose four different approaches, based on the above models, to solve QRP. Model 2 and Model 4 have a polynomial numbers of variables and constraints. They can be directly used in a MIP solver (CPLEX). These approaches will be denoted *Mod2_B&B* and *Mod4_B&B*. Model 1 and Model 3 have an exponential number of constraints. The constraints are relaxed, and Branch & Cut approaches are employed.

3.1 Lazy constraint approach

This approach is applicable to Model 1 and Model 3, where connectivity constraints (1a) and (3c) are considered as lazy constraints. The linear programming relaxation of the initial model without lazy constraints is solved. All isolated components are identified for every feasible integral solution that is not yet feasible. If a solution to a linear programming relaxation is feasible, then there is no isolated component. To check for isolated components, we use the union-find data structure [9, 18]. If there is only one component, then the solution is feasible. Otherwise, there is a cycle in some components, a lazy constraint is then added for each component as follows. For Model 3, C is the set of all nodes of the component and i is a random node in C; for Model 1, C is the set of all nodes of the component. All these lazy constraints are then added directly to the model, and the linear programming relaxation of the current model is reoptimized. This procedure is repeatedly executed until an optimal solution has been found. The two corresponding approaches will be denoted *Mod1_B&C_lazy* and *Mod3_B&C_lazy*.

3.2 Dynamic constraint separation approach

This approach can be applied to Model 3, where connectivity constraints (3c) are dynamically separated [11]. This approach, denoted by *Mod3_B&C_dyn*, finds violated connectivity constraints on the support graph. Given a solution x^* to a linear programming relaxation (containing all connectivity constraints separated so far), the support graph G^* of x^* has all nodes V and edges $\{(i,j) \in E : x^*_{ij} > 0\}$[4].

This approach consists of two stages. First, the support graph is checked for isolated components not connected to the root node. For each isolated component, only constraint (3c) is added to the current model, where C is the set of all nodes in the isolated component and i is a random node in C. Second, if the support graph has only one component that includes the root node r, a maximum $r-v-$flow/minimum $r-v-$cut problem is solved for each node $v \neq r$ in the component. To solve maximum-flow/minimum-cut problems, we use code written by Skorobohatyj [30]. A maximum flow that is less than the absolute inflow to v indicates a violated connectivity constraint (3c), in which C are all nodes on the same side of the $r-v$ cut as v (of course node i in the constraint is the node v).

3.3 Preprocessing

Different reduction checks have been proposed for the minimum Steiner tree problem and others [22, 19, 25, 31, 23, 2]. In the preprocessing of QRP, the following *check for*

useless nodes is useful. It is performed on the undirected graph G. If a node is not a multicast node and its degree is only one, then this node (and its edge) can be removed from the graph G. If a node v is not a multicast node with exactly two neighbors u and w, then the node v and the edges (u, v) and (v, w) can be removed. If there exists an edge (u, w) with cost c_{uw}, this cost is updated to $min(c_{uw}, c_{uv} + c_{vw})$. Otherwise, an edge (u, w) is added with cost $c_{uv} + c_{vw}$. These checks can be applied iteratively until the graph remains unchanged. In practice, we limit ourselves to three iterations. Other reductions were considered, but they had only very marginal impact.

4 Computational experiments

In [29], our approach based on CP was tested on 960 random instances, with the largest graph having 60 nodes. It was shown that the CP approach was better than the existing state of the art complete approaches. We reuse these instances. We collect these instances into a class called **C1**.

We also collect 2500 instances in a class **C2**, generated from 100 undirected graphs of 160 nodes and 25 couples $\langle q, |S| \rangle$, ranging from $\{\langle 3, 20 \rangle$ to $\langle 119, 140 \rangle \}$. These 100 undirected graphs were extracted from 100 minimum Steiner tree instances of test set **I160** in the library SteinLib [20]. The multicast nodes were randomly chosen.

All MIP approaches were implemented in C++, using IBM Ilog Cplex Concert Technology, version 12.4. The standard Cplex cuts were automatically added. The CP approach in [29] was implemented in Comet [8]. Finally, all experiments were performed on XEN virtual machines with 1 core of a CPU Intel Core2 Quad Q6600 @2.40GHz and 1GB of RAM. The time limit for each execution of an algorithm was 30 minutes.

4.1 Comparing the approaches

We first compare the MIP approaches as well as the CP approach from [29]. Figure 1 gives a summary of the experimental results. The columns have the following meanings: $\%opt$ is the percentage of instances solved to optimality within the time limit of 30 minutes; $\overline{I}$ is the average number of iterations; $\overline{N}$ is the average of the number of nodes in the branch-and-bound tree; $\overline{C}$ is the average of the number of separated constraints; $\overline{T}$ is the average computational time in seconds (on the solved instances). Figure 2 shows the evolution of the percentage of solved instances in **C2** with respect to the time limit.

It is clear that all the MIP approaches significantly outperform the CP approach. In the class **C1**, the MIP approaches are two orders of magnitude faster than CP. In the class **C2**, CP only solved two instances out of 2500, while *Mod4_B&B* solved 95.2% of the instances. In Figures 1 and 2, there is no major difference between *Mod3_B&C_lazy* and *Mod4_B&B*, nor between *Mod1_B&C_lazy* and *Mod3_B&C_dyn*. It is clear that *Mod3_B&C_lazy* and *Mod4_B&B* are the best two approaches, both in the percentage of solved instances and in the execution time. The approach *Mod2_B&B* is the worst among the MIP approaches, although it develops few nodes. The low number of nodes results from the fact that the integer relaxation version of this model is quite close to the optimal solution [17]. It is worth noting that *Mod3_B&C_dyn* has the smallest number of iterations. This mainly comes from the dynamic constraint separation approach, which

Class	Approach	$\%opt$	$\overline{I}$	$\overline{N}$	$\overline{C}$	$\overline{T}$
C1	*CP*	97.1	na.	na.	na.	80.47
	Mod1_B&C_lazy	100	441.7	50.43	6.58	0.57
	Mod2_B&B	100	2159	1.38	na.	1.06
	Mod3_B&C_lazy	100	398.8	77.51	81.42	0.35
	Mod3_B&C_dyn	100	308.2	3.69	158.1	1.94
	Mod4_B&B	100	506.2	55.9	na.	0.64
C2	*CP*	0.08	na.	na.	na.	9.74
	Mod1_B&C_lazy	78.6	92804	8507	1110	150.9
	Mod2_B&B	60.2	66716	7.05	na.	318.6
	Mod3_B&C_lazy	94.4	30938	2312	1077	97.6
	Mod3_B&C_dyn	77.2	8408	54.64	6089	153.0
	Mod4_B&B	95.2	50327	6138	na.	92.8

Fig. 1. A summary of computational results for two classes of instances

produces smaller and thinner branch-and-bound trees. However, the number of added constraints is much larger.

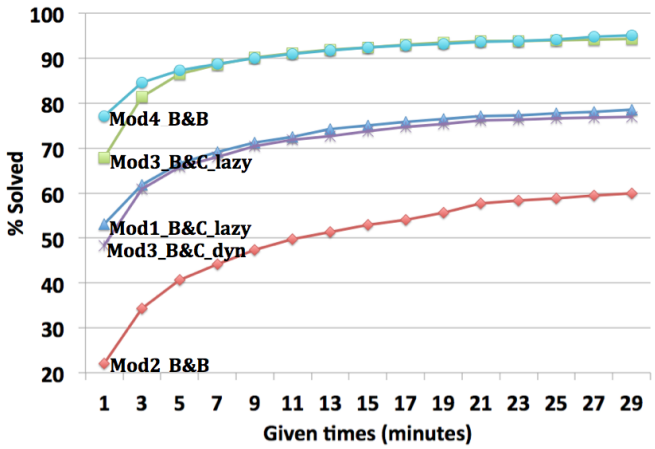

Fig. 2. Percentage of solved instances in **C2** in given times

Although not reported here for reasons of space, the experimental results also showed that Property (2) of the QRP solutions (Section 2) is very useful in all the models proposed in this paper. For example, this property helps this model to solve 6.2% more instances of class **C2**.

4.2 Effect of the values of q and $|S|$ on the performance of the approaches

We analyze the sensitivity of the performance to the value of q and the size of the multicast node set. We divided the instances of class **C2** into two sets of groups. In the

first set, a group contains instances of the same size of multicast node set. The groups *G20* (resp. *G50, G80, G110* and *G140*) consist of all instances with $|S| = 20$ (resp. $50, 80, 110$ and 140). In the second set, a group contains 500 instances with similar $\frac{q}{|S|}$. The groups *G1* through *G5* split the instances from the smallest to the largest values of $\frac{q}{|S|}$.

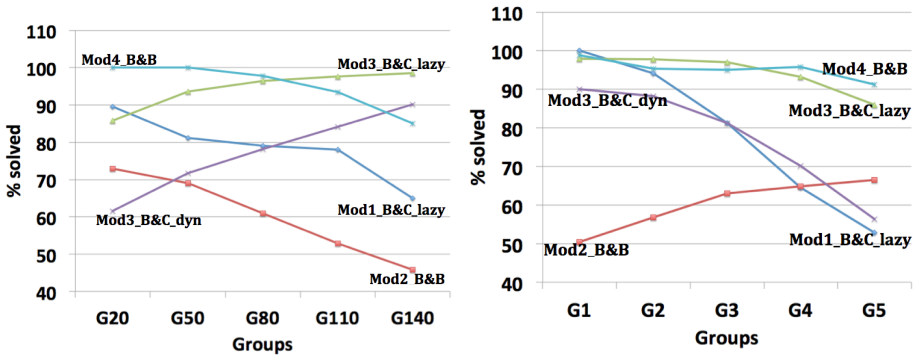

Fig. 3. Comparing MIP approaches in solving groups of instances with respect to the percentage of solved instances

The experimental results for each group are given in Figure 3. First, the different approaches have differing sensitivities to q. The instances in group *G5* are more difficult to solve than those in the other groups, except for *Mod2_B&B*. When considering the ratio $\frac{q}{|S|}$, *Mod3_B&_lazy* and *Mod3_B&_dyn* are better for high value of this ratio, while the other approaches are worse. These results also confirm that *Mod3_B&C_lazy* and *Mod4_B&B* are the two best approaches. However, there is a significant difference between these two approaches when the number of multicast nodes varies. A portfolio approach could be used to select *Mod4_B&B* for low values of $|S|$, and *Mod3_B&C_lazy* for high values.

5 Conclusion

This paper solved the quorumcast routing problem to optimality as a mixed integer program. In this paper, we proposed four mathematical formulations for QRP. We then solved QRP to optimality as a mixed integer program, introducing two constraint relaxations. The computational results showed that the MIP approaches are much more efficient than the state of the art approach (which is based on Constraint Programming). In addition, we showed that two of the approaches, *Mod3_B&C_lazy* and *Mod4_B&B*, are the two best ones. Finally, experimental results pointed out that the different approaches have different sensitivities to the parameters q and the size of the multicast node set. As future research, new separation constraints could be investigated.

Acknowledgments The authors want to thank the anonymous reviewers for their helpful comments. This research is partially supported by the FRFC project 2.4504.10 of the Belgian FNRS and the UCLouvain Action de Recherche Concertée ICTM22C1.

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
2. J.-F. Cordeau A.M. Costa and G. Laporte. Steiner tree problems with profits. *INFOR*, 44:99–115, 2006.
3. Kim Allan Andersen, Kurt Jrnsten, and Mikael Lind. On bicriterion minimal spanning trees: An approximation. *Computers and amp; Operations Research*, 23(12):1171 – 1182, 1996.
4. David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.
5. Shun Yan Cheung and A. Kumar. Efficient quorumcast routing algorithms. In *INFOCOM '94. Networking for Global Communications., 13th Proceedings IEEE*, pages 840 –847 vol.2, jun 1994.
6. Markus Chimani, Maria Kandyba, Ivana Ljubić, and Petra Mutzel. Obtaining optimal k-cardinality trees fast. *J. Exp. Algorithmics*, 14:5:2.5–5:2.23, January 2010.
7. S. Chopra and C.Y. Tsai. Polyhedral approaches for the steiner tree problem on graphs. In D.-Z. Du X. Cheng, editor, *Steiner Trees in Industries*, volume 11, page 175202. Kluwer Academic Publishers, 2001.
8. Comet. *Comet user manual, dynadec, 2011. http://dynadec.com/.*
9. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
10. Adelaide Cerveira Cristina Requejo, Agostinho Agra and Eullia Santos. Formulations for the weightconstrained minimum spanning tree problem. In *AIP Conf. Proc. 1281*, pages 2166–2169, 2010.
11. Michael Drexl and Stefan Irnich. Solving elementary shortest-path problems as mixed-integer programs. *OR Spectrum*, pages 1–16, 2012.
12. B. Du, J. Gu, D.H.K. Tsang, and W. Wang. Quorumcast routing by multispace search. In *Global Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity*, volume 2, pages 1069 –1073 vol.2, nov 1996.
13. Tetsuya Fujie. The maximum-leaf spanning tree problem: Formulations and facets. *Networks*, 43(4):212–223, 2004.
14. Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
15. Michel X. Goemans and Young-Soo Myung. A catalog of steiner tree formulations. *Networks*, 23(1):19–28, 1993.
16. Sebastian Tobias Henn. *Weight-Constrained Minimum Spanning Tree Problem*. PhD thesis, University of Kaiserslautern, 2007.
17. M.S. Ibrahim, N. Maculan, and M. Minoux. A strong flow-based formulation for the shortest path problem in digraphs with negative cycles. *International Transactions in Operational Research*, 16(3):361–369, 2009.
18. Wayne K. Union-find algorithms. http://www.cs.princeton.edu/rs/AlgsDS07/01UnionFind.pdf. 2008.
19. T. Koch and A. Martin. Solving steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.
20. T. Koch, A. Martin, and S. Voß. SteinLib: An updated library on steiner tree problems in graphs. Technical Report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin, 2000.
21. R.V. Kulkarni and P.R. Bhave. Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20(1):58 – 67, 1985.

22. Ivana Ljubi, Ren Weiskircher, Ulrich Pferschy, Gunnar W. Klau, Petra Mutzel, and Matteo Fischetti. An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. *Mathematical Programming*, 105:427–449, 2006.
23. Ivana Ljubi, Ren Weiskircher, Ulrich Pferschy, Gunnar W. Klau, Petra Mutzel, and Matteo Fischetti. An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. In *Mathematical Progamming, Series B*, 2006.
24. Chor Ping Low. A fast search algorithm for the quorumcast routing problem. *Inf. Process. Lett.*, 66(2):87–92, April 1998.
25. A. Lucena and J. E. Beasley. A branch and cut algorithm for the steiner problem in graphs. *Networks*, 31:39–59, 1998.
26. Abilio Lucena, Nelson Maculan, and Luidi Simonetti. Reformulations and solution algorithms for the maximum leaf spanning tree problem. *Computational Management Science*, 7:289–311, 2010.
27. Thomas L. Magnanti and Laurence A. Wolsey. Chapter 9 optimal trees. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503 – 615. Elsevier, 1995.
28. C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, October 1960.
29. QuangDung Pham and Yves Deville. Solving the quorumcast routing problem by constraint programming. *Constraints*, 17:409–431, 2012.
30. Georg Skorobohatyj. Finding a minimum cut between all pairs of nodes in an undirected graph. http://elib.zib.de/pub/Packages/mathprog/mincut/all-pairs/index.html. 2008.
31. Eduardo Uchoa. Reduction tests for the prize-collecting steiner problem. *Operations Research Letters*, 34(4):437 – 444, 2006.
32. B. Wang and J. C. Hou. An efficient QoS routing algorithm for quorumcast communication. *Computer Networks Journal*, 44(1):43–61, 2004.