The Minimum Connected Dominating Set Problem: Formulation, Valid Inequalities and a Branch-and-Cut Algorithm

Luidi Simonetti[1], Alexandre Salles da Cunha[2], and Abilio Lucena[3]

[1] Universidade Federal Fluminense, Instituto de Computação
luidi@ic.uff.br
[2] Universidade Federal de Minas Gerais, Departamento de Ciência da Computação
acunha@dcc.ufmg.br
[3] Universidade Federal do Rio de Janeiro, Programa de Engenharia de Sistemas e Computação
abiliolucena@globo.com

Abstract. We consider the minimum connected dominating set problem. We present an integer programming formulation and new valid inequalities. A branch-and-cut algorithm based on the reinforced formulation is also provided. Computational results indicate that the reinforced lower bounds are always stronger than the bounds implied by the formulation from which resulted one of the best known exact algorithms for the problem. In some cases, the reinforced lower bounds are stronger than those implied by the strongest known formulation to date. For dense graphs, our algorithm provides the best results in the literature. For sparse instances, known to be harder, our method is outperformed by another one. We discuss reasons for that and how to improve our current computational results. One possible way to achieve such goals is to devise specific separation algorithms for some classes of valid inequalities introduced here.

1 Introduction

Let $G = (V, E)$ be a connected undirected graph with a set of vertices $V = \{1, \ldots, n\}$ and edges E ($m = |E|$). Given $i \in V$, assume that $\Gamma_i \subseteq V$ denotes the union of $\{i\}$ with the set of vertices of V that share an edge with i. A set $W \subseteq V$ is a dominating set of G if, for every $i \in V$, there is $k \in \Gamma_i \cap W$. Given any set W, let $E(W) = \{\{i, j\} \in E : i, j \in W\}$ be the subset of edges of E with both endpoints in W. A dominating set W is connected if the subgraph $G_W = (W, E(W))$ of G is connected. In the Minimum Connected Dominating Set Problem (MCDSP), one wishes to find a connected dominating set of G with minimum cardinality. MCDSP was shown to be NP-hard in [6].

MCDSP is closely related to the Maximum Leaf Spanning Tree Problem [7] (MLSTP), which consists in finding a spanning tree of G with as many leaves as possible. It should be clear that, given a dominating set W, a spanning tree $T(W) = (W, E_T)$ of G_W could be found efficiently. Such a tree could be enlarged into a spanning tree T of G, where all vertices in $V \setminus W$ are leaves. Thus, for every connected dominating set W of G, a spanning tree of G with at least $n - |W|$ leaves could be efficiently found. In particular, if W is a minimum connected dominating set, a spanning tree of G with the maximum possible number of leaves results from the procedure outlined above.

Applications for MCDSP (and MLSTP, too) arise, e.g., in the design of ad-hoc wireless sensor networks, where network topologies may change dynamically [1] as well as the design of fiber optics networks where regenerators of information may be required at some vertices [3]. Polyhedral investigations for MLSTP were conducted in [4] and exact algorithms have been proposed in [5,9]. Two *integer programming* (IP) formulations were proposed for MLSTP in [9]. MCDSP has been tackled by approximation algorithms in [7,11]. Although not explicitly stated, the underlying problem associated with the Regenerator Location Problem (RLP) in [3] is a MCDSP.

Here we present an IP formulation, valid inequalities and a *branch-and-cut* (BC) algorithm [12] for MCDSP. Exploring specific properties of MCDSP, new valid inequalities are proposed. Lower bounds obtained by the reinforced formulation significantly improves on the original bounds. For some instances, such bounds are stronger than the best known bounds reported in [9]. Preliminary results obtained with the BC algorithm implemented here suggest that the formulation might be promising. Though, our current BC implementation is outperformed by the overall best exact algorithm for MLSTP in [9], for sparse instances.

The paper is organized as follows. In Section 2, we present the IP formulation for MCDS. We discuss a BC algorithm in Section 3. Computational results are reported in Section 4. We conclude the paper in Section 5.

2 Integer Programming Formulation

In order to present an IP formulation for MCDSP, let us use the following decision variables: $y_i \in \{0,1\}$, $i \in V$: to select which vertices are to be included ($y_i = 1$) or not ($y_i = 0$) in the dominating set; $x_e \in \{0,1\}$, $e \in E$: to choose edges that guarantee that the dominating set is indeed connected. In what follows, assume that $\mathbb{B} = \{0,1\}$ and that $\mathbb{R}$ denotes the set of real numbers. An IP formulation for MCDSP is:

$$\min \left\{ \sum_{i \in V} y_i : (x,y) \in \mathscr{R}_0 \cap (\mathbb{R}^m_+, \mathbb{B}^n) \right\}, \tag{1}$$

where polyhedral region $\mathscr{R}_0$ is implied by:

$$\sum_{e \in E} x_e = \sum_{i \in V} y_i - 1, \tag{2a}$$

$$\sum_{e \in E(S)} x_e \le \sum_{i \in S \setminus \{j\}} y_i, \quad S \subset V, j \in S \tag{2b}$$

$$\sum_{j \in \Gamma_i} y_j \ge 1, \quad i \in V \tag{2c}$$

$$x_e \ge 0, \quad e \in E \tag{2d}$$

$$0 \le y_i \le 1, \quad i \in V. \tag{2e}$$

The idea behind the formulation above is to use variables x to select edges that guarantee that a spanning tree must be found in the subgraph $G_W = (W, E(W))$, implied by a dominating set W. More precisely, constraint (2a) guarantees that the number of selected edges is exactly one unity less than the number of vertices in a connected dominating set. Generalized Subtour Breaking Constraints (GSEC) (2b) guarantee that the selected edges imply a tree. Constraints (2c), on the other hand, make sure that the set of vertices selected in a solution is dominating.

Formulation $\mathscr{R}_0$ embeds two basic structures. On the one hand, constraints (2a), (2b), (2d) – (2e) fully characterize the tree polytope of G [10]. The other structure, implied by constraints (2c) and (2e), is that of the Covering Problem. Facet defining inequalities for the covering polytope are widely recognized as difficult to separate [2]. In the sequence, we show how formulation $\mathscr{R}_0$ can be strengthened by other means, using problem specific arguments.

Firstly, we claim that (2c) can be lifted to

$$\sum_{j\in\Gamma_i} y_j - \sum_{e\in E(\Gamma_i)} x_e \geq 1, \forall i \in V. \tag{3}$$

To show that (3) is valid for MCDSP, consider a connected dominating set W. Since $|W \cap \Gamma_i| \geq 1$ and since the edges in $E(W)$ selected to span the set must imply a tree, we have that the number of selected edges in $E(\Gamma_i)$ must be at most $|\Gamma_i| - 1$. Otherwise, at least one cycle must be included in the solution.

Constraints (3) can also be viewed as a strengthened version of the Generalized Subtour Breaking constraints (2b). To verify that, let $S = \Gamma_i$, for which the corresponding GSEC reads as: $\sum_{e\in E(\Gamma_i)} x_e \leq \sum_{j\in\Gamma_i\setminus\{k\}} y_j, k \in \Gamma_i$. Since at least one vertex in Γ_i must be chosen, the latter can be replaced by the stronger form $\sum_{e\in E(\Gamma_i)} x_e \leq \sum_{j\in\Gamma_i} y_j - 1$, which is precisely (3). The previous observation leads to a lifting of (2b) as follows. Assume that, for any given way, we can certify that, out of those vertices in a particular given set $C \subset V$, at least one vertex must be included in a connected dominating set. Clearly, for sets satisfying such property, GSECs (2b) can be replaced by the stronger version:

$$\sum_{e\in E(C)} x_e \leq \sum_{j\in C} y_j - 1. \tag{4}$$

To present another valid inequality for MCDSP, assume that given $S \subset V$, $S \neq \emptyset$, $\Gamma(S) := \bigcup_{i\in S} \Gamma_i$ and that $(S, V\setminus S) := \{\{i,j\} \in E : i \in S, j \notin S\}$ denotes the edges in the cut implied by S. Whenever $\Gamma(S) \neq V$ and $\Gamma(V\setminus S) \neq V$, at least one edge in $(\Gamma(S), V \setminus \Gamma(S))$ must be chosen. This is true since the vertices in a connected dominating set cannot be exclusively confined to S or to $V\setminus S$. Mathematically, we have that:

$$\sum_{e\in(S,V\setminus S)} x_e \geq 1, \forall S \subset V : \Gamma(S) \neq V, \Gamma(V\setminus S) \neq V. \tag{5}$$

Therefore, a strengthened formulation $\mathscr{R}_1$ for MCDSP can be obtained by constraints (2a)-(2b),(2d),(2e), (3),(4) and (5).

3 Branch-and-Cut Algorithm

In this section, we provide the main implementation details on a preliminary BC algorithm for MCDSP, based on formulation $\mathscr{R}_1$. The algorithm was implemented with calls to XPRESS MIP solver (release 19.00) callback routines. The algorithm implements a *best-first* search strategy. All other default XPRESS settings were used.

The algorithm starts solving the LP relaxation

$$\min \sum_{i \in V} y_i : \ (x,y) \in \mathscr{P}, \tag{6}$$

where polytope $\mathscr{P}$ is given by (2a),(2d),(2e), (3) and $x_{ij} \leq y_i, \{i,j\} \in E$, as well as $x_{ij} \leq y_j, \{i,j\} \in E$.

Let $(\overline{x},\overline{y}) \in \mathscr{P}$ be the solution to (6) and $\overline{G} = (\overline{V},\overline{E})$ be the subgraph of G implied by $(\overline{x},\overline{y})$ ($\overline{V} := \{i \in V : \overline{y}_i > 0\}$ and $\overline{E} := \{\{i,j\} \in E : \overline{x}_{ij} > 0\}$). If $(\overline{x},\overline{y})$ is integer and if there is no GSEC (2b) violated by $(\overline{x},\overline{y})$, $(\overline{x},\overline{y})$ solves (1). Otherwise, we attempt to reinforce $\mathscr{P}$, appending violated valid inequalities to it.

The exact separation of GSECs can be carried out efficiently, through max-flow (min-cut) computations, in $O(n^4)$ [13]. Despite that, in our current implementation, we do not use any exact separation algorithm. Here, the algorithm of [13] is only used to present a tighter approximation of the bounds implied by $\mathscr{R}_1$. In practice, in our BC method, we found advantageous to separate GSECs only through the following heuristic. We sort the edges in $\overline{E}$ in non-increasing order of their $\overline{x}_{ij}$ values. Then, we find a forest of maximum cardinality of $\overline{G}$, using Kruskal's algorithm [8], giving preference to include edges with higher values of $\overline{x}_{ij}$. Each edge included during Kruskal's method merges two sets of vertices into a new connected component in the forest being built. We check for violated GSECs for the connected components generated after each edge inclusion, until a forest of maximum cardinality has been found. If no violated GSECs are found with the separation heuristic, we branch on y variables.

Whenever (with the GSEC separation heuristic outlined above) we find a set C whose corresponding GSEC is violated, we attempt to lift the inequality to (4), by checking whether at least one vertex in C must be in a connected dominating set for G. One simple test allowing to conclude so is the following. Whenever $\Gamma(V \setminus C) \neq V$, (4) holds for C. Apart from that, specific separation algorithms for inequalities (4) are not yet implemented as well as separation algorithms for inequalities (5).

Initial valid upper bounds for MCDSP are computed with the dynamic greedy heuristic introduced in [9]. It is oriented towards generating spanning trees of G with as many leaves as possible or, equivalently, to enforce that these trees contain as few inner vertices as possible. The heuristic works with two sets: $\mathscr{D}$, to represent vertices in a connected dominating set and $\mathscr{L}$, to represent those vertices which have at least one neighbor in $\mathscr{D}$. The procedure is initialized by $\mathscr{D} = \{v\}$ and $\mathscr{L} = \Gamma_v \setminus \{v\}$ for any $v \in V$. The basic operation is to try to push vertices from $\mathscr{L}$ into $\mathscr{D}$, until a connected dominating set is found. Assuming that i is moved from $\mathscr{L}$ to $\mathscr{D}$, in the next iteration we have: $\mathscr{D} \leftarrow \mathscr{D} \cup \{i\}$ and $\mathscr{L} \leftarrow \mathscr{L} \setminus \{i\} \cup (\Gamma_i \setminus \mathscr{D})$. Preference is given to include in $\mathscr{D}$ vertices with as many neighbors as possible, not already included in $\mathscr{L}$. The heuristic stops when $V = \mathscr{L} \cup \mathscr{D}$, when $\mathscr{D}$ represents a connected dominating set and $\mathscr{L}$ denotes the set of leaf implying vertices in the tree.

4 Preliminary Computational Results

The algorithms introduced here were implemented in C and all computational testings were conducted in a Intel XEON machine running at 2Ghz, with 8 Gbytes of RAM

Table 1. MCDSP lower bounds and Branch-and-cut results

		Lower Bounds				OPT	Branch-and-cut			$t_{DGR}(s)$
n	density (%)	$\mathscr{R}_0$	$\mathscr{R}_1$	DGR	STR		$t(s)$	BLB	BUB	
30	10	8.60	14.40	14.12	14.34	15	0.01	15	15	0.01
	20	5.09	6.18	5.68	6.49	7	0.02	7	7	0.10
	30	2.92	3.60	3.05	3.50	4	0.05	4	4	0.03
	50	1.95	2.38	1.86	2.11	3	0.04	3	3	0.08
	70	1.36	1.83	1.30	2.00	2	0.02	2	2	0.01
50	5	15.55	31.00	31.00	31.00	31	0.02	31	31	0.01
	10	9.17	10.68	10.37	11.15	12	0.42	12	12	0.36
	20	4.76	5.24	4.88	5.52	7	0.66	7	7	1.32
	30	3.28	3.69	3.26	3.93	5	0.25	5	5	1.21
	50	1.98	2.44	1.82	2.20	3	0.25	3	3	0.51
	70	1.45	1.84	1.31	2.00	2	0.29	2	2	0.04
70	5	17.10	26.31	25.29	26.44	27	1.42	27	27	0.26
	10	9.82	11.23	10.90	11.40	13	34.29	13	13	4.73
	20	4.92	5.37	5.12	5.63	7	2.16	7	7	16.30
	30	3.27	3.62	3.20	3.86	5	1.00	5	5	2.90
	50	2.05	2.44	1.95	2.05	3	0.70	3	3	1.33
	70	1.43	1.91	1.35	2.00	2	0.79	2	2	1.92
100	5	18.00	21.63	20.79	22.04	24	342.25	24	24	12.50
	10	10.05	10.98	10.62	11.07	13	32.11	13	13	9.36
	20	5.24	5.52	5.15	5.62	8	174.93	8	8	86.16
	30	3.37	3.74	3.33	-	6	193.65	6	6	258.15
	50	2.10	2.51	1.97	-	4	35.41	4	4	132.55
	70	1.45	1.94	1.36	2.05	3	12.03	3	3	154.10
120	5	19.12	22.74	22.48	22.87	35	-	24.34	26	2.65
	10	9.79	10.66	10.33	10.87	13	-	12.79	15	65.49
	20	5.14	5.35	5.07	-	8	610.89	8	8	393.47
	30	3.40	3.76	3.31	-	6	475.54	6	6	653.70
	50	1.99	2.49	1.37	2.15	4	168.55	4	4	815.64
	70	1.44	1.92	-	-	3	31.67	3	3	356.31
150	5	19.60	21.72	21.35	21.94	26	-	23.74	27	2954.00
	10	10.27	10.69	10.56	10.84	14	-	12.47	15	3247.89
	20	5.05	5.37	4.95	-	9	-	6.97	9	-
	30	3.42	3.81	3.33	-	6	1954.00	6	6	2317.35
	50	1.98	2.47	1.90	-	4	481.61	4	4	2756.36
	70	1.44	1.99	1.37	-	3	43.75	3	3	1828.86
200	5	20.35	22.52	22.17	22.69	-	-	23.78	29	-
	10	10.16	10.53	10.39	-	-	-	11.9	16	-
	20	4.95	5.26	4.87	-	-	-	6.41	9	-
	30	3.35	3.77	3.23	-	-	-	4.56	7	-
	50	2.01	2.53	1.93	-	4	2249.43	4	4	20155.00
	70	1.44	2.00	1.37	2.03	3	271.91	3	3	8154.13

memory. In order to evaluate BC and the quality of the lower bounds of the proposed reinforced model, we considered those MCDSP instances introduced in [9], for which $n \in \{30, 50, 70, 100, 120, 150, 200\}$ and graph densities range from 5% to 70%. For each value of n and graph density, BC was allowed to run for at most 3600 seconds, after which, the problem is left unsolved.

Detailed computational results are reported in Table 1. In the first two columns of the table, we report n and the graph density. In the subsequent four columns, we present lower bounds for MCDSP: the lower bound implied by $\mathscr{R}_0$, an approximation of the lower bound implied by $\mathscr{R}_1$, followed by the lower bounds implied by the Directed Graph Reformulation (DGR) and by the Steiner Arborescence Reformulation (STR) in [9]. In the next column, we report the optimal objective function value (OPT) for the instance, whenever available (an entry "-" in that column indicates that the corresponding optimal value is yet unknown). In the next three columns, we present results for our BC method: $t(s)$, the time (in seconds) it takes to solve the instance and the best lower (BLB) and upper (BUB) bound found after the search. Whenever BC does not solve the instance within the imposed time limit, an entry "-" is given in that column. In the last column of the table, we report $t_{DGR}(s)$, the time (in seconds) needed by the BC algorithm in [9] (based on DGR) to solve the instance. These times were obtained in a machine whose speed is directly comparable to the one used in our testings. We do not compare our method with the BC algorithm in [3], since instances in that study were not available to us.

As can be seen from Table 1, optimizing over $\mathscr{R}_1$ allows significantly stronger lower bounds, when compared to the bounds implied by $\mathscr{R}_0$. We should recall that the bounds reported in column $\mathscr{R}_1$ are not the true bounds provided by that formulation, since inequalities (4) and (5) are not separated yet (only GSECs were separated exactly, for the sake of approximating the bounds given by $\mathscr{R}_1$). In many cases, these approximations are stronger than the best lower bounds in [9], given by STR. Note also that the new reinforced lower bounds are always at least as strong as the bounds implied by DGR, the formulation from which resulted the best exact algorithm for MSLTP in [9].

For small instances, with up to 70 vertices, the BC algorithm implemented here is always faster than the BC algorithm based on DGR. Similar conclusions can be drawn for larger dense instances. However, despite the potential benefits of optimizing over $\mathscr{R}_1$, for sparse larger instances, the BC procedure implemented here is outperformed by the one based on DGR. For example, note that when $n = 120$ and graph density is 10% or less, the algorithm implemented here failed on solving the problem within the imposed time limit, whereas the algorithm in [9] solved such instances quite easily, in less than 66 seconds. One possible explanation is the following. Cuts based on GSECs are usually denser than directed cutset counterparts. As a consequence, solving the LP relaxations involved in the formulation discussed here is likely to be more time consuming than solving DGR relaxations.

5 Conclusions

Our computational results indicate that the (approximated) lower bounds implied by the reinforced formulation are stronger than the bounds given by the formulation from which resulted one of the best exact solution approaches in MCDSP literature. Our method is at least as fast as the Branch-and-cut algorithm in [9] for dense graphs. Despite the potential benefits of optimizing over a better approximation of the convex hull of feasible solutions, our Branch-and-cut method is outperformed by the best solution algorithm in [9] for sparse problems.

We believe that our computational results could be improved significantly, if specific separation algorithms for the new valid inequalities were implemented. In addition, the valid inequalities suggested here for an undirected formulation for MCDSP have direct counterparts for MCDSP formulated in a directed graph. Such inequalities should be easier to separate over a directed structure. Consequently, a Branch-and-cut algorithm based on a directed version of the formulation provided here could also benefit from the stronger lower bounds introduced in this study. Benefits of such directed Branch-and-cut implementation could arise also from other two different sources. Firstly, directed cutsets are typically sparser than GSECs. Consequently, the time needed to solve the Linear Programming relaxations may decrease. Secondly, directed cutsets can be separated in $O(n^3)$ time, while GSECs take $O(n^4)$, in the worst case.

Acknowledgements

Alexandre Salles da Cunha is partially funded by CNPq grant 302276/2009-2 and FAPEMIG PRONEX APQ-01201-09. Abilio Lucena is partially funded by CNPq grant 310561/2009-4. Luidi Simonetti is partially funded by FAPERJ grant E-26/111.819/2010 and CNPq grant 483.243/2010-8.

References

1. Balasundaram, B., Butenko, S.: Graph domination, coloring and cliques in telecommunications. In: Handbook of Optimization in Telecommunications, pp. 865–890. Springer, Heidelberg (2006)
2. Borndörfer, R.: Aspects of Set Packing, Partitioning and Covering. Ph.D. thesis, Konrad-Zuse-Zentrum fur Informationstechnik, Berlin (1998)
3. Chen, S., Ljubić, I., Raghavan, S.: The regenerator location problem. Networks 55(3), 205–220 (2010)
4. Fujie, T.: The Maximum-leaf Spanning Tree Problem: formulations and facets. Networks 43(4), 212–223 (2004)
5. Fujie, T.: An exact algorithm for the Maximum-leaf Spanning Tree Problem. European Journal of Operational Research 104, 250–261 (2003)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness (1979)
7. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica 20(4), 374–387 (1998)
8. Kruskal, J.: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Proceedings of the American Mathematical Society 7, 48–50 (1956)
9. Lucena, A., Maculan, N., Simonetti, L.: Reformulation and solution algorithms for the maximum leaf spanning tree problem. Computational Management Science 7, 289–311 (2010)

10. Magnanti, T.L., Wolsey, L.: Optimal Trees. In: Ball, O., et al. (eds.) Handbooks in OR and MS, vol. 7, pp. 503–615. North-Holland, Amsterdam (1995)
11. Marathe, M.V., Breu, H., Hunt III, H.B., Ravi, S.S., Rosenkrantz, D.J.: Simple heuristics for unit disc graphs. Networks 25, 59–68 (1995)
12. Padberg, M.W., Rinaldi, G.: A Branch-and-Cut algorithm for resolution of large scale of Symmetric Traveling Salesman Problem. SIAM Review 33, 60–100 (1991)
13. Padberg, M.W., Wolsey, L.: Trees and cuts. Annals of Discrete Mathematics 17, 511–517 (1983)