CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Benders Decomposition, Branch-and-Cut and Hybrid Algorithms for the Minimum Connected Dominating Set Problem

Bernard Gendron
Abilio Lucena
Alexandre Salles da Cunha
Luidi Simonetti

December 2012

CIRRELT-2012-76

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

Benders Decomposition, Branch-and-Cut and Hybrid Algorithms for the Minimum Connected Dominating Set Problem

Bernard Gendron[1,*], Abilio Lucena[2], Alexandre Salles da Cunha[3], Luidi Simonetti[4]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[2] Departamento de Administração, Universidade Federal do Rio de Janeiro, Av. Pedro Calmon, no 550 Prédio da Reitoria, 2° andar Cidade Universitária, Rio de Janeiro, Brazil, CEP: 21941-901

[3] Departamento de Ciênca da Computação, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627 Pampulha, Belo Horizonte, MG, Brazil, CEP: 31270-901

[4] Instituto de Computação, Universidade Federal Fluminense Rua Passo da Pátria 156 - Bloco E, 3º andar, São Domingos Niterói, RJ, Brazil, CEP: 24210-240

Abstract. We present exact algorithms for solving the minimum connected dominating set problem in an undirected graph. The algorithms are based on two approaches: a Benders decomposition algorithm and a branch-and-cut method. We also develop a hybrid algorithm that combines these two approaches. Two variants of each of the three resulting algorithms are considered: a stand-alone version and an iterative probing variant. The latter variant is based on a simple property of the problem, which states that if no connected dominating set of a given cardinality exists, then there are no connected dominating set of lower cardinality. We present computational results on a large set of randomly generated instances.

Keywords. Connected dominating set, valid inequalities, branch-and-cut, Benders decomposition.

Acknowledgements. Bernard Gendron is partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants Program. Abilio Lucena is partially funded by CNPq grant 310561/2009-4 and FAPERJ grant E26-110.552/2010. Alexandre Salles da Cunha is partially funded by CNPq grants 302276/2009-2, 477863/2010- 8 and FAPEMIG PRONEX APQ-01201-09. Luidi Simonetti is partially funded by FAPERJ and CNPq grants 483.243/2010-8, 304793/2011-6. All sources of financial support are gratefully acknowledged.

---

* Corresponding author: Bernard.Gendron@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1 Introduction

A dominating set in an undirected connected graph $G = (V, E)$ is a set $D \subseteq V$ such that $\Gamma(D) = V$, where $\Gamma(D) = D \cup \{j \in V | \{i, j\} \in E, i \in D\}$. The minimum dominating set problem consists in finding a dominating set of minimum cardinality. A connected dominating set is a dominating set D such that the subgraph $G(D) = (D, E(D))$ is connected, where $E(D) = \{\{i, j\} \in E | i \in D, j \in D\}$. The minimum connected dominating set problem (MCDSP) consists in identifying a connected dominating set of minimum cardinality.

The MCDSP is closely related to the maximum leaf spanning tree problem (MLSTP), which consists in finding a spanning tree of G with as many leaves as possible (see Lucena et al. (2010) for a review of the literature on the MLSTP). Indeed, given a connected dominating set D, a spanning tree of $G(D)$ can be easily identified. Such a tree can be enlarged into a spanning tree of G, where all vertices in $V \setminus D$ are leaves. Thus, for every connected dominating set D of G, a spanning tree of G with at least $|V| - |D|$ leaves can be efficiently found. In particular, if D is a minimum connected dominating set, a spanning tree of G with the maximum possible number of leaves results from the procedure outlined above.

Domination in graphs is a concept behind a growing number of applications found in the literature. Early applications could be traced back to the location of radar stations (Berge 1973) and a particular network communication problem described in Liu (1968). Nowadays, applications can be found in areas as diverse as the spread of technological innovations (Rogers 2003, Valente 1995), the marketing of new products (Domingos and Richardson 2001, Goldenberg et al. 2001), failures in power systems (Asavathiratham et al. 2001), Web graph problems (Cooper et al. 2005), the spread of communicable diseases (Eubank et al. 2004, Stanley 2006) and helping to alleviate social problems through social networks (Wang et al. 2011).

Applications that specifically involve MCDSP/MLSTP arise in the design of ad-hoc wireless sensor networks, where network topologies may change dynamically (Balasundaram and Butenko 2006). They could also be found in the design of defense strategies against the attack of worms in peer-to-peer networks (Liang and Sencun 2007). Another recent application appears in Chen et al. (2010) and addresses the design of fiber optics networks where regenerators of information may be required at some network vertices (Chen et al. 2010). Regenerators, which are expensive equipments, are necessary to boost information quality, degraded after traveling long distances in cable. Finally, minimum connected dominating sets are also suggested as models to investigate protein-protein interactions (Milenković et al. 2011).

For the MLSTP, polyhedral investigations are carried out in Fujie (2004), while exact algorithms are developed in Fujie (2003) and Lucena et al. (2010). A branch-and-bound algorithm is presented in Fujie (2003), where clever insights lead to the efficient computation of the (relatively weak) corresponding linear programming (LP) relaxation bounds. Two integer programming (IP) formulations are proposed in Lu-

cena et al. (2010). The first one is based on a Steiner reformulation of the problem. The second one considers the problem in a directed graph, seeking for a spanning arborescence with as many leaves as possible. Although the lower bounds implied by the former are stronger, better computational results are obtained with a branch-and-cut method based on the latter. A heuristic method for the MLSTP is also suggested in Lucena et al. (2010). This heuristic method and the Steiner reformulation proposed in Lucena et al. (2010) are similar to those introduced in Chen et al. (2010) for the regenerator location problem, these contributions being independently developed. For the MCDSP, approximation algorithms are developed in Guha and Khuller (1998) and Marathe et al. (1995). Additionally, a preliminary version of a branch-and-cut algorithm, to be further investigated here, is described in Simonetti et al. (2011).

In this paper, we present two exact algorithms for the MCDSP, which rely on a general formulation of the problem based on the following binary variables: $y_i = 1$, if $i \in V$ belongs to a dominating set, 0 otherwise. The model can be stated as follows:

$$z = \min \sum_{i \in V} y_i \tag{1}$$

$$\sum_{j \in \Gamma(\{i\})} y_j \geq 1, \quad i \in V, \tag{2}$$

$$\textit{connected(y)} \tag{3}$$

$$y_i \in \{0, 1\}, \quad i \in V. \tag{4}$$

The objective function, (1), minimizes the number of vertices in any connected dominating set. The *cover inequalities*, (2), define a dominating set. The generic constraint (3) imposes the connectivity of the subgraph induced by y. The two exact algorithms, a Benders decomposition approach and a branch-and-cut method, differ in the way they handle the generic constraint (3). The *Benders decomposition algorithm* iterates between: 1) the solution of a master problem defined by (1), (2) and (4), plus a number of additional inequalities, thus providing a dominating set D induced by y; and 2) the solution of a subproblem, represented by the generic constraint (3), that verifies whether or not D is a connected subgraph and adds cuts to the master problem accordingly. To represent constraint (3), the *branch-and-cut algorithm* uses a classical spanning tree formulation that introduces additional edge-based variables representing whether or not an edge belongs to a spanning tree for the subgraph induced by y. In addition to these two methods, we also investigate a *hybrid algorithm* that applies the Benders decomposition strategy, but builds an initial master problem at each iteration by adding valid inequalities derived from the spanning tree formulation, as in the branch-and-cut method.

We further exploit a basic property of connected dominating sets to devise an *iterative probing strategy* that can be used in combination with any of the three algorithms. This property simply states that if no connected dominating set of a given

cardinality $d > 0$ exists, then there are no connected dominating set of cardinality $d - 1$. We use this property within a simple iterative approach that starts with an initial connected dominating set of cardinality d and attempts to find a connected dominating set of cardinality $d - 1$; if no such set is found, the iterations stop with the current best solution of cardinality d. At every iteration, we use any of our three algorithms to find a connected dominating set of cardinality $d - 1$, thus giving rise to the iterative probing variants of each of the three algorithms. To summarize, we present and compare six exact algorithms for the MCDSP: the *stand-alone* and *iterative probing* variants of the Benders decomposition, the branch-and-cut and the hybrid methods.

The paper is organized as follows. In Section 2, we give the details of the iterative probing strategy. In Section 3, we present the Benders decomposition method, while the branch-and-cut algorithm is the topic of Section 4. Section 5 describes the hybrid algorithm. Computational results are reported in Section 6. We conclude the paper in Section 7.

Notation. We use the following notation throughout the paper: let $G = (V, E)$ be a connected undirected graph and $\mathcal{P}(V) = \{S \subset V | S \neq \emptyset\}$ be the collection of all proper subsets of V; for any $S \in \mathcal{P}(V)$, we denote by $\overline{S}$ its complement $V \setminus S$ and by $\Gamma(S) = S \cup \{j \in V | \{i, j\} \in E, i \in S\}$ its closed neighborhood (when $S = \{i\}$, we write $\Gamma(\{i\}) = \Gamma(i)$); $\mathcal{D} = \{S \in \mathcal{P}(V) | \Gamma(S) = V\}$ and $\overline{\mathcal{D}} = \mathcal{P}(V) \setminus \mathcal{D}$ are the collections of all dominating sets and all non-dominating sets of G, respectively; for any $S \in \mathcal{P}(V)$, we denote by $G(S) = (S, E(S))$ the subgraph induced by S, where $E(S) = \{\{i, j\} \in E | i \in S, j \in S\}$; $\mathcal{C} \subseteq \mathcal{P}(V)$ and $\overline{\mathcal{C}} = \mathcal{P}(V) \setminus \mathcal{C}$ are the collections of all proper subsets of V that induce a connected subgraph and a disconnected subgraph of G, respectively.

2 Iterative Probing Strategy

The iterative probing strategy is based on the following property:

Proposition 1 *If there exists a connected dominating set of cardinality $d < |V|$, then there exists a connected dominating set of cardinality $d + 1$.*

This property is trivial to show. Assume there exists a connected dominating set D of cardinality $d < |V|$; by adding any vertex in $V \setminus D$, we then obtain a connected dominating set of cardinality $d+1$. As a direct consequence of this property, we have:

Corollary 2 *If there are no connected dominating set of cardinality $d + 1 > 1$, then there are no connected dominating set of cardinality d.*

Given a connected dominating set of cardinality $d + 1 > 1$, the iterative probing strategy simply looks for a connected dominating set of cardinality d. If there is no such set, the algorithm stops. Otherwise, a connected dominating set of cardinality

d is obtained; at the next iteration, the algorithm looks for a connected dominating set of cardinality $d-1$. More formally, the iterative probing strategy can be stated as follows:

1. Find a connected dominating set D; let $d+1 = |D|$. If $d = 0$, then stop: D is the optimal solution of value $d+1$.
2. *Probing*: Try to find a connected dominating set of cardinality d.
3. If no connected dominating set has been found, then stop: D is the optimal solution of value $d+1$.
4. Let D be the connected dominating set just found; let $d+1 = |D|$ and return to step 2.

Under the assumption that the decision problem in step 2 is solved exactly, this strategy provides an optimal solution to the MCDSP, by virtue of Corollary 2. At the first step of this strategy, the case $d = 0$ is easy to verify, prior to the solution of any problem instance, by checking the condition $|\Gamma(i)| = |V|$ for some $i \in V$; whenever this is the case, the minimum connected dominating set contains only one element and the problem is trivially solved.

To solve the decision problem in step 2, we add to the general formulation (1)-(4) the following *d-cut equation*:

$$\sum_{i \in V} y_i = d. \tag{5}$$

The corresponding decision problem, called the d-CDSP, can be solved by any of the three methods described in the next sections: the Benders decomposition algorithm (see Section 3), the branch-and-cut algorithm (see Section 4) or the hybrid algorithm (see Section 5).

3 Benders Decomposition Algorithm

The general strategy in Benders decomposition is to alternate between solving a master problem and a so-called Benders subproblem. In the master problem, some constraints of the original problem are relaxed, which induces a lower bound on the optimal objective function value. In our case, the generic constraint *connected*(y) is relaxed and the master problem is defined by (1), (2), (4) and additional constraints, called *Benders cuts*. Given a solution to the master problem, the Benders subproblem attempts to identify a solution that satisfies all the constraints. Here, the solution y to the master problem induces a dominating set D and a subgraph $G(D) = (D, E(D))$ for which we verify the connectivity, thus attempting to enforce the constraint *connected*(y). Determining the connectivity of $G(D)$ can be performed

in $O(|E(D)|)$ by a graph traversal algorithm. If $G(D)$ is connected, we obtain an upper bound on the optimal objective function value and the algorithm stops, since the lower and upper bounds are equal. Otherwise, the solution y is not feasible and so-called *Benders feasibility cuts* are generated and added to the master problem to be solved at the next iteration.

As mentioned above, the existence of a dominating set of cardinality 1 can be easily verified by checking the condition $|\Gamma(i)| = |V|$ for some $i \in V$. Provided such a set does not exist, the cover inequalities (2) can be strengthened as follows:

$$\sum_{j \in \Gamma(i) \setminus \{i\}} y_j \geq 1, \quad i \in V. \tag{6}$$

This simple modification allows to avoid trivial master problems at the initial stages of the algorithm.

3.1 Benders Cuts

In Benders decomposition, the classical way of cutting a feasible solution of value z^u is simply to impose the constraint that the objective function value should be strictly less than z^u. In our case, since the objective function value is integer, this constraint can be written as:

$$\sum_{i \in V} y_i \leq z^u - 1. \tag{7}$$

This optimality cut is added to the master problem when solving the MCDSP. The value z^u is determined by the heuristic methods described in Appendix A.

Let D be the dominating set found when solving the Benders master problem; when the subgraph induced by D is not connected, at least one vertex in $\overline{D} \equiv V \setminus D$ must be included in any minimum connected dominating set, yielding the feasibility cut:

$$\sum_{i \in \overline{D}} y_i \geq 1. \tag{8}$$

Since there is a finite number of such cuts and, eventually, all disconnected dominating sets would be discarded by adding these cuts, the algorithm converges to an optimal solution to the MCDSP. This type of feasibility cuts arise naturally in the so-called logic-based and combinatorial Benders decomposition frameworks (Codato and Fischetti 2006, Hooker and Ottosson 2003).

Because the resulting algorithm is convergent, it suggests a mathematical programming formulation for the MCDSP, obtained by adding to (1), (4) and (6) the following constraints, called *cut inequalities*:

$$\sum_{i \in \overline{S}} y_i \geq 1, \quad S \in \overline{\mathcal{C}}, \tag{9}$$

where $\overline{\mathcal{C}}$, as indicated above, is the collection of all proper subsets of V that induce disconnected subgraphs. This formulation is similar to one of the models proposed by Fujie (2004) for the MLSTP.

The cut inequalities are in general weak, since many disconnected sets require much more than one extra node to become connected. In order to characterize stronger versions of the cut inequalities, we introduce the notion of a *minimally disconnected set*, which is a set of vertices $T \subset V$ that induces a disconnected subgraph, but such that there exists one vertex $j \in \overline{T}$ for which $G(T \cup \{j\})$ is connected. Any cut inequality associated to a disconnected, but not minimally disconnected, set S is dominated by at least one cut inequality associated to a minimally disconnected set T. Indeed, from any set $S \in \overline{\mathcal{C}}$, it is easy to derive a minimally disconnected set T that generates a tighter cut inequality. To this purpose, it suffices to solve a shortest chain problem with unit lengths in the graph obtained by shrinking all connected components of S. One of the shrunk nodes (connected components) is arbitrarily declared as the source, while all other shrunk nodes are the sinks. Solving the shortest chain problem with unit lengths in this graph (which can be performed in time $O(|E|)$) provides the minimum number of vertices, say m_S, that must be added to S in order to obtain a connected subgraph of G. By removing any of the added vertices, we define a minimally disconnected set T. Since $\sum_{i\in\overline{S}} y_i \geq \sum_{i\in\overline{T}} y_i \geq 1$, the cut inequality associated to the minimally disconnected set T dominates that associated to the disconnected set S, i.e., $\sum_{i\in\overline{T}} y_i \geq 1$ is a lifting of the cut inequality corresponding to S. This implies a second formulation for the MCDSP, where the cut inequalities are restricted to minimally disconnected sets. Let $\overline{\mathcal{C}}_1 = \overline{\mathcal{C}} \cap \{S \in \mathcal{P}(V) | m_S = 1\}$, the collection of minimally disconnected sets; the model is then defined by adding to (1), (4) and (6), the following *lifted cut inequalities*:

$$\sum_{i\in\overline{S}} y_i \geq 1, \quad S \in \overline{\mathcal{C}}_1. \tag{10}$$

In the framework of combinatorial Benders decomposition (Codato and Fischetti 2006), our notion of minimally disconnected set corresponds to the concept of *minimal infeasible subsystem*.

Thus, given a disconnected dominating set D obtained when solving the Benders master problem, one can derive a minimally disconnected set through the shortest chain algorithm outlined above and generate a corresponding lifted cut inequality. This algorithm would follow the framework of combinatorial Benders decomposition (Codato and Fischetti 2006). For the MCDSP, we can improve this algorithm by using the following result, which implies that the lifted cut inequalities can be further tightened:

Proposition 3 *Let $S \in \overline{\mathcal{C}}$ and m_S be the minimum number of vertices that must be added to S to obtain a connected subgraph of G. The following inequality is valid for*

the MCDSP:

$$\sum_{i \in \overline{S}} y_i \geq m_S. \tag{11}$$

Proof: There are two possibilities for S:
1) S is a dominating set, in which case m_S is the smallest number of vertices required to enlarge S to obtain a connected, and therefore a connected dominating, set; thus, the result follows.
2) S is not a dominating set, in which case m_S is the smallest number of vertices required to enlarge S to obtain a connected set; m_S is therefore a lower bound on the number of vertices required to enlarge S to obtain a connected and dominating set for G. The result thus follows. □

This proposition implies a third formulation for the MCDSP, obtained by adding to (1), (4) and (6), the following *strengthened cut inequalities*:

$$\sum_{i \in \overline{S}} y_i \geq m_S, \quad S \in \overline{\mathcal{C}}. \tag{12}$$

Given a disconnected dominating set D obtained when solving the Benders master problem, one can compute m_D through the shortest chain algorithm outlined above and generate a corresponding strengthened cut inequality. Clearly, this inequality dominates the associated lifted cut inequality. Indeed, let $U \subseteq D$ be the set of vertices removed from D to obtain a minimally disconnected set $D \setminus U$ to generate the lifted cut inequality. By definition of U and m_D, we have $m_D = 1 + |U|$. Thus, $\sum_{i \in \overline{D}} y_i \geq m_D \geq 1 + \sum_{i \in U} y_i$, which implies $\sum_{\overline{(D \setminus U)}} y_i \geq 1$, the lifted cut inequality.

Thus, for the MCDSP, we are able to generate stronger feasibility cuts than the ones derived in the combinatorial Benders framework. As seen below, the strengthened cut inequalities are instrumental for the algorithm to obtain efficient computational results. Also, since these cuts can be derived independently of the particular Benders decomposition algorithm that we developed, they can be used in other methods; we will use them in our branch-and-cut algorithm presented in Section 4.

3.2 Outline of the Algorithm

In this section, we give an outline of the stand-alone variant of the Benders decomposition algorithm; the iterative probing variant of the same algorithm is described in Section 3.3. The algorithm solves the model defined by (1), (4), (6) and (12). The optimality cut (7) is also added.

To initialize the algorithm, we perform the heuristic method described in Appendix A, followed by the application of the strengthening procedure to be described in Section 3.3. We thus obtain a connected dominating set D of value $z^u = |D|$, which is used in the optimality cut (7). The Benders decomposition algorithm can be outlined as follows:

1. Find a connected dominating set D of value z^u; if $z^u = 1$, then stop: D is the optimal solution.
2. Solve the Benders master problem.
3. If no feasible solution has been found, then stop: D is the optimal solution of value z^u.
4. Otherwise, let D be the dominating set just found; if D is a connected dominating set, then stop: D is the optimal solution.
5. If D is not connected, generate a strengthened feasibility cut and return to step 2.

3.3 Iterative Probing Variant

At every step of the iterative probing strategy, we can perform the Benders decomposition algorithm outlined above by simply adding to the master problem formulation the d-cut equation (5). We can, however, further improve the performance of the iterative probing variant of Benders decomposition by generating, instead of the cut on the objective function (7), which is implied by the d-cut equation (5), another form of optimality cut that mirrors the feasibility cut. If the subgraph induced by D is connected, then, in order to find an improving connected dominating set, we must exclude D from further consideration, which can be done by requiring at least one vertex in D to be excluded from an optimal solution, yielding the following cut:

$$\sum_{i \in D} y_i \leq |D| - 1. \tag{13}$$

This constraint is implied by the d-cut equation (5), since $d = |D| - 1$. Hence, this weaker version of the optimality cut is never added to the master problem. However, it is possible to efficiently derive from it a strengthened cut as follows. If, for each vertex $i \in D$, removing i from D yields a disconnected subgraph, the right-hand side can then be replaced by $|D| - 2$, since in this case, we know there cannot be a feasible solution obtained by removing any single vertex in D. Likewise, if, for each vertex $i \in D$, removing i from D yields a non-dominating subgraph, the right-hand side can also be replaced by $|D| - 2$, since in this case also, there cannot be an optimal solution obtained by removing any single vertex in D.

The *strengthening procedure* thus scans each vertex $i \in D$ to verify if $D \setminus \{i\}$ is a connected dominating set; therefore, it can be performed in time $O(|D||E|)$. After scanning the vertices in D, if we determine that removing i from D yields a disconnected or a non-dominating subgraph for each vertex $i \in D$, we then generate the *strengthened optimality cut*:

$$\sum_{i \in D} y_i \leq |D| - 2. \tag{14}$$

If, for some vertex $i \in D$, $D \setminus \{i\}$ is a connected dominating set, we have identified a feasible solution of value $d = |D| - 1$. We immediately stop checking the condition for the other vertices in D; instead, we restart the strengthening procedure with $D \setminus \{i\}$ in place of D. As a result, the strengthening procedure will always terminate with a strengthened optimality cut (associated to D or to a subset of D defining a connected dominating set), potentially generating a series of successively improving feasible solutions along the way.

It is worth noting that it suffices to maintain at most one strengthened optimality cut over the whole course of the algorithm. Indeed, let us assume that the Benders master problem at the current iteration includes the d-cut equation (5), with right-hand side $d = |D| - 1$, as well as the strengthened optimality cut (14). Assuming the problem is feasible, we then obtain a dominating set D' of cardinality d. If D' is disconnected, a feasibility cut is generated, but no optimality cut. If D' is connected, we update the d-cut equation $\sum_{i \in V} y_i = d - 1 = |D'| - 1 = |D| - 2$, which dominates the strengthened optimality cut $\sum_{i \in D} y_i \leq |D| - 2$. The latter can therefore be removed and possibly replaced by another one of the form $\sum_{i \in D''} y_i \leq |D''| - 2$, where $D'' \subseteq D'$.

Note that the strengthened optimality cut (14) dominates the Benders feasibility cut associated to each disconnected set $D \setminus \{i\}$. Each such set is minimally disconnected, since D is connected; the Benders feasibility cut has the form $\sum_{j \in \overline{(D \setminus \{i\})}} y_j \geq 1$. This inequality is implied by the d-cut equation $\sum_{j \in V} y_j = |D| - 1$ and the strengthened optimality cut, since $\sum_{j \in \overline{(D \setminus \{i\})}} y_j = \sum_{j \in V} y_j - \sum_{j \in D \setminus \{i\}} y_j = \sum_{j \in V} y_j - \sum_{j \in D} y_j + y_i \geq (|D| - 1) - (|D| - 2) + y_i \geq 1$.

The iterative probing variant of the Benders decomposition algorithm follows the stand-alone variant, with a few exceptions. First, the optimality cut (7) is replaced by the d-cut equation (5) for which we gradually decrease the right-hand side; a strengthened optimality cut of the form (14) is also added to the Benders master problem. The heuristic method described in Appendix A is performed to provide a connected dominating set D of value $|D| = d + 1$, which is used to initialize the d-cut equation (5). In addition, we apply the strengthening procedure to this initial connected dominating set D. Since D is a feasible solution, but not necessarily a minimum dominating set, the strengthening procedure might generate an improved feasible solution, as well as an initial strengthened optimality cut of the form (14), which is added to the Benders master problem. The initial Benders master problem thus contains the d-cut equation (5) and the strengthened optimality cut (14) associated to the best connected dominating set D found so far.

The iterative probing variant of the Benders decomposition algorithm can be outlined as follows:

1. Find a connected dominating set D; if $|D| = 1$, then stop: D is the optimal solution.

2. Apply the strengthening procedure; add the resulting strengthened optimality

cut to the Benders master problem; if a new connected dominating set D has been found, let $d = |D| - 1$ and update the right-hand side of the d-cut equation.

3. Solve the Benders master problem.
4. If no feasible solution has been found, then stop: D is the optimal solution.
5. Let D be the dominating set just found; if D is a connected dominating set:
   (a) Let $d = |D| - 1$ and update the right-hand side of the d-cut equation.
   (b) Apply the strengthening procedure; add the new strengthened optimality cut to the Benders master problem; if a new connected dominating set D has been found, let $d = |D| - 1$ and update the right-hand side of the d-cut equation.
6. If D is not connected, generate a strengthened feasibility cut and return to step 3.

4 Branch-and-Cut Algorithm

The branch-and-cut algorithm is based on a particular representation of the generic constraint *connected*(y) as linear inequalities, namely, a representation that defines a spanning tree of the subgraph of G induced by y. Denote by $G(D)$ such a subgraph, $D \subset V$ being its vertex set. Additionally, define $x_e = 1$, if edge $e \in E$ belongs to a spanning tree of $G(D)$, 0 otherwise. Finally, consider a polyhedral region $\mathcal{P}_0$ defined by inequalities (2) and the following constraints, that characterize a spanning tree of $G(D)$:

$$\sum_{e \in E} x_e = \sum_{i \in V} y_i - 1, \tag{15}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{i \in S \setminus \{j\}} y_i, \quad S \in \mathcal{P}(V),\ j \in S, \tag{16}$$

$$0 \leq x_e \leq 1, \quad e \in E, \tag{17}$$

$$0 \leq y_i \leq 1, \quad i \in V. \tag{18}$$

A formulation for the MCDSP (Simonetti et al. 2011) is thus given by:

$$z = \min \{\sum_{i \in V} y_i | (x, y) \in \mathcal{P}_0 \cap (\mathbb{R}_+^{|E|}, \{0,1\}^{|V|})\}. \tag{19}$$

One classical approach for strengthening the LP relaxation of an IP model consists in identifying structures in the formulation and then appending valid inequalities for each structure into the original model. Our formulation embeds two basic structures. Denoting by $\mathcal{T}$ the polyhedral region defined by (15)-(18), the first one, i.e., the tree

polytope, is defined as the convex hull of $\{(x, y) \in \mathcal{T} \cap (\mathbb{R}_+^{|E|}, \{0,1\}^{|V|})\}$. Likewise, denoting by $\mathcal{S}$ the polyhedral region defined by (2) and (18), the second structure, i.e., the set covering polytope, is defined as the convex hull of $\{y \in \mathcal{S} \cap \{0,1\}^{|V|}\}$. Facet-defining inequalities for the set covering polytope are notoriously difficult to separate (Borndörfer, R. 1998). In the sequel, we show how $\mathcal{P}_0$ can be strengthened by other means, using problem specific arguments.

4.1 Valid Inequalities

The cover inequalities (2) can be lifted to

$$\sum_{j \in \Gamma(i)} y_j - \sum_{e \in E(\Gamma(i))} x_e \geq 1, \quad i \in V. \tag{20}$$

To show that (20) is valid for the MCDSP, consider a connected dominating set D and a vertex $i \in V$. Since $|D \cap \Gamma(i)| \geq 1$ and since the edges in $E(D)$ selected to span the set form a tree, we have that the number of selected edges in $E(\Gamma(i))$ must be at most $|\Gamma(i)| - 1$ (otherwise, there would be at least one cycle in the solution). Note that this strengthened version of (2) implies the cover constraints (6) used in the Benders decomposition approach.

From another standpoint, constraints (20) can be viewed as a strengthened version of the *generalized subtour elimination constraints*, or GSECs, (16). 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 $\Gamma(i)$ 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 (20).

Replacing inequalities (2) with those in (20) lead to substantial improvements in LP relaxation bounds. For some test instances, particularly those defined over very sparse graphs, LP bounds increased by as much as 90%.

Let us now introduce a lifting for another type of GSEC that follows from the same type of arguments used above to obtain stronger cover inequalities. Assume that, among the vertices in a particular set $S \in \mathcal{P}(V)$, at least one vertex must be included in a connected dominating set. In particular, this is true whenever $\overline{S}$ is non-dominating or disconnected, i.e., $\overline{S} \in \overline{\mathcal{D}}$ or $\overline{S} \in \overline{\mathcal{C}}$. Thus, GSECs (16) can be replaced by the stronger version:

$$\sum_{e \in E(S)} x_e \leq \sum_{j \in S} y_j - 1, \quad \overline{S} \in \overline{\mathcal{D}} \cup \overline{\mathcal{C}}. \tag{21}$$

For $S \in \overline{\mathcal{C}}$, we also make use of the strengthened cut inequalities (12), derived as Benders feasibility cuts in Section 3.1.

Another valid inequality for the MCDSP can be derived by observing that, whenever S and its complement $\overline{S}$ are both non-dominating, at least one edge in $E(S, \overline{S}) =$

$\{\{i,j\} \in E | i \in S, j \in \overline{S}\}$ (i.e., the edges in the cut implied by S) must be chosen. This is true since the vertices in a connected dominating set cannot be exclusively confined to S or to $\overline{S}$. More formally, we have:

$$\sum_{e \in E(S,\overline{S})} x_e \geq 1, \quad S \in \overline{\mathcal{D}}, \overline{S} \in \overline{\mathcal{D}}. \tag{22}$$

A particular case of inequalities (22) arises when $\Gamma(i) \cap \Gamma(j) = \emptyset$ for a given pair of vertices $i, j \in V$. Under these conditions, it is immediate to verify that any cutset $E(S, \overline{S})$ separating $\Gamma(i)$ and $\Gamma(j)$ implies a valid inequality of the type (22).

Separating (22) for the particular case highlighted above is not difficult. To illustrate it, assume that an LP relaxation of formulation (19) is available together with its corresponding support graph. After gluing together the vertices respectively found in $\Gamma(i)$ and $\Gamma(j)$, super vertices i and j would result in the support graph. In doing so, when applicable, multiple edges between two given end nodes are replaced by a single edge. Additionally, the weight of that edge is set equal to the sum of the weights of the edges that imply it. It thus follows that determining the maximum flow between super vertices i and j identifies a minimum capacity cut between $\Gamma(i)$ and $\Gamma(j)$ and therefore solves the separation problem.

In order to evaluate the benefits of such separation scheme, we keep a list with all pairs of vertices i and j such that $\Gamma(i) \cap \Gamma(j) = \emptyset$. During each LP relaxation at the root node of the enumeration tree, we identify, as described above, the minimum cut separating $\Gamma(i)$ and $\Gamma(j)$, for each pair i, j in the list. The impact of adding cuts identified this way into the LP relaxation at the root node is, however, quite small. Therefore, we actually do not use such a separation procedure in the final version of our branch-and-cut algorithm.

Finally, inequalities (22) can be generalized whenever V is partitioned into subsets $(S_1, \ldots, S_k)$, for $k \geq 2$, where S_l, for any $l \in \{1, \ldots, k\}$, satisfies the following conditions: (1) $S_l \in \overline{\mathcal{D}}$ and (2) $S_l \cap S \neq \emptyset$, for any $S \in \mathcal{D}$. Under these conditions, the following inequality is valid for $\mathcal{P}_0$:

$$\sum_{e \in E(S_1,\ldots,S_k)} x_e \geq k - 1, \tag{23}$$

where $E(S_1, \ldots, S_k)$ is the set of edges with endpoints in different partition sets. These inequalities relate to the Steiner partition facets introduced in Chopra and Rao (1994a,b) and are not easy to separate exactly. However, as described below, we have devised a simple separation heuristic for them.

Preliminary computational results for the formulations described in this section appeared in Simonetti et al. (2011). In that reference, the LP relaxation bounds are obtained by optimizing the objective function over polytope $\mathcal{P}_1 = \mathcal{P}_0^+ \cap \{(22)\}$, where $\mathcal{P}_0^+$ corresponds to $\mathcal{P}_0$ with the stronger cover inequalities (20) being used instead of (2). In Section 6, these results are compared with stronger bounds obtained by optimizing over polytope $\mathcal{P}_1^+ = \mathcal{P}_1 \cap \{(12), (21) \text{ and } (23)\}$.

4.2 Outline of the Algorithm

The branch-and-cut algorithm is initialized by generating an upper bound with the heuristic described in Appendix A. The strengthening procedure of Section 3.3 is then applied in an attempt to improve the feasible solution thus obtained and to generate a strengthened optimality cut (14). In addition, the strengthening procedure is called every time the branch-and-cut algorithm generates a feasible solution. A new strengthened optimality cut (14) is then added to the model. Furthermore, this optimality cut is also stored in a cut pool to ensure that the cut is available for use at the nodes to be explored after backtracking.

Valid dual bounds are obtained by first solving the LP relaxation of

$$\min \{\sum_{i \in V} y_i | (x, y) \in \mathcal{P} \cap (\mathbb{R}_+^{|E|}, \{0,1\}^{|V|})\}, \tag{24}$$

$\mathcal{P}$ being the polyhedral region defined by (15), (20) and

$$x_e \leq y_i,\ x_e \leq y_j, \quad e = \{i, j\} \in E, \tag{25}$$

where (25) naturally follow from the GSECs corresponding to $S = \{i, j\}$, $i, j \in V$, $i \neq j$. Let $(\overline{x}, \overline{y})$ be an optimal solution to this model and $\overline{G} = (\overline{V}, \overline{E})$ be the support subgraph it implies for G, i.e., $\overline{V} = \{i \in V | \overline{y}_i > 0\}$ and $\overline{E} = \{e \in E | \overline{x}_e > 0\}$. If $(\overline{x}, \overline{y})$ is integer and there is no GSEC (16) violated by it, the solution is optimal for the MCDSP. Otherwise, one should attempt to reinforce the relaxation by appending to it valid inequalities that are violated by $(\overline{x}, \overline{y})$.

The exact separation of GSECs can be efficiently carried out in $O(n^4)$ time complexity Padberg and Wolsey (1983), via a maximum flow-minimum cut algorithm. However, for the solution algorithms investigated in this paper, GSECs are not separated exactly, since in terms of overall branch-and-cut running times, it proves more advantageous to only separate GSECs heuristically, through a procedure described below. In spite of that, for comparison purposes, corresponding values of $\min\{\sum_{i \in V} y_i | (x, y) \in \mathcal{P}_0^+\}$, $\min\{\sum_{i \in V} y_i | (x, y) \in \mathcal{P}_1\}$ and $\min\{\sum_{i \in V} y_i | (x, y) \in \mathcal{P}_1^+\}$ obtained under the exact separation of GSECs, are reported in Section 6.1.

Our heuristic separation of GSECs is carried out as follows. First, the edges in $\overline{E}$ are sorted in non-increasing order of their $\overline{x}_e$ values. Then, a maximum cardinality forest of $\overline{G}$ is computed through Kruskal algorithm Kruskal (1956). Preference for entering the solution is given to edges with higher $\overline{x}_e$ values. In accordance with Kruskal algorithm, each edge entering the solution merges two connected components into a larger one. In this process, for every new connected component being formed, their vertices are checked for GSEC violation. The procedure stops after a maximum cardinality forest is obtained.

Although driven for separating GSECs, the heuristic outlined above is also used to separate additional families of valid inequalities. This is carried out right after an edge inclusion operation is performed by the heuristic. Accordingly, let S be the

vertex set for the connected component thus obtained, where $\overline{S}$ is its complement in V. Sets S and $\overline{S}$ are then checked for violation of strengthened GSECs (21), strengthened feasibility cuts (12), and cutsets (22). More precisely, if $\overline{S} \in \overline{\mathcal{D}}$, at least one vertex in S must be part of any connected dominating set and therefore S implies a valid lifted GSEC inequality (21) that should be checked for violation. Moreover, if $S \in \overline{\mathcal{C}}$, the value of m_S should be computed and the inequality (12) implied by S should be checked for violation. Finally, if S and $\overline{S}$ belong to $\overline{\mathcal{D}}$, violation of (22) should be checked. Furthermore, if $\overline{S} \in \overline{\mathcal{C}}$ also applies, the cut inequality is lifted into a k-partition inequality (23). If no violated inequality is found by the heuristic, branching on variables is implemented.

4.3 Iterative Probing Variant

At each step of the iterative probing strategy, the branch-and-cut algorithm is performed to solve the decision problem obtained by appending the d-cut equation (5) to the model. When a feasible solution of value d is obtained, the branch-and-cut algorithm is stopped. A new step of the iterative probing strategy is then performed. First, the strengthening procedure is applied in an attempt to improve the current feasible solution of value d. Then, the model is initialized with the updated d-cut equation and all the cuts generated so far, except GSECs (16). Finally, the branch-and-cut algorithm for solving the new decision problem is performed.

Valid inequalities used to strengthen the formulation are the same as those used in the stand-alone version of the branch-and-cut algorithm. Accordingly, at every iterative probing solution round, inequalities (12), (16), and (21) to (23) are separated as described above.

5 Hybrid Algorithm

Our computational results, to be presented in Section 6, show that the branch-and-cut algorithm outperforms the Benders decomposition method on sparse instances, while the Benders decomposition algorithm is much faster on dense instances. This observation motivated the development of a hybrid algorithm that attempts to combine the best features of the two approaches. Indeed, the relaxations built by the branch-and-cut algorithm for sparse instances are extremely good, while the Benders decomposition algorithm improves the feasible solutions quickly for dense instances.

The hybrid algorithm therefore implements a Benders decomposition algorithm, but builds stronger restricted master problems by performing the separation of valid inequalities (12), (16), and (21) to (23), when solving the root node of every Benders master problem. More precisely, from one Benders iteration to the next, all the cuts generated so far are kept in the Benders master problem, while additional cuts can be generated at the root node. Thus, at each iteration of the hybrid algorithm, the Benders master problem is solved by a cut-and-branch algorithm, where the

master problem at the root node is initialized with all the cuts generated at previous iterations. In the stand-alone version of the hybrid algorithm, the main loop of the Benders decomposition algorithm is performed until an optimal solution is found (see Section 3.2).

The iterative probing version of the hybrid algorithm proceeds in a similar way. The separation of valid inequalities is conducted at the root node of every Benders master problem and the generated cuts are kept for subsequent master problems. The main loop of the iterative probing variant of the Benders decomposition method is then performed (see Section 3.3).

6 Computational Experiments

In this section, we empirically evaluate the six exact solution algorithms presented in this paper: the stand-alone (SABC) and iterative probing (IPBC) branch-and-cut algorithms, the stand-alone (SABE) and iterative probing (IPBE) Benders decomposition algorithms and the stand-alone (SAHY) and iterative probing (IPHY) hybrid algorithms. All algorithms were implemented in C and computational experiments were carried out on a 2.0 GHz Intel XEON E5405 machine with 8 Gbytes of RAM memory. Search tree management for SABC and IPBC was enforced via the callback routines of the mixed-IP solver `XPRESS`, release 19.00. The enumeration strategy implemented for these two algorithms was best-first search. For the Benders decomposition algorithms, the MIP module of `XPRESS` was used, under default settings, to solve the Benders master problems.

The MLSTP/MCDSP instances used in our experiments were introduced in Lucena et al. (2010). They are associated with graphs $G = (V, E)$ with densities ranging from 5% to 70% and number of vertices $n \in \{30, 50, 70, 100, 120, 150, 200\}$. At most one test instance exists for every possible combination of n and d and therefore any given instance is clearly identified in our tables as `n_d`. For each of these instances, every solution algorithm was allowed to run for at most 3600 CPU seconds. Whenever that limit was reached and optimality had not been proven, the instance was left unsolved by the corresponding algorithm.

6.1 Linear Programming Lower Bounds

Table 1 presents a number of different LP relaxation bounds for the MCDSP and the MLSTP. Entries in the first column identify the test instances. This is followed, in the next three columns, by MCDSP LP relaxation bounds respectively implied by polytopes $\mathcal{P}_0^+$, $\mathcal{P}_1$ and $\mathcal{P}_1^+$. For the computation of these three bounds, we first call the separation heuristic for GSECs outlined above. Therefore, not only GSECs, but also inequalities (12) and (21)-(23) are separated. If no violated cut is found by the heuristic, the exact separation of GSECs, through minimum cut algorithms, is carried out next. During the application of the GSEC exact separation procedure in

Padberg and Wolsey (1983), each vertex set (and its complement) that results from a minimum cut computation is checked (when applicable) for the violation of (12) and (21)-(23). Therefore, the LP relaxation bounds quoted for $\mathcal{P}_1$ and $\mathcal{P}_1^+$ represent a lower bound on the values that would otherwise be obtained if exact separation were used. Entries in the following two columns give LP relaxation bounds for the two different MLSTP reformulations investigated in Lucena et al. (2010), namely, the directed graph (DGR) reformulation and the Steiner tree reformulation (STR). MLSTP bounds, in this case, are expressed in terms of their corresponding MCDSP bounds. Finally, optimal MCDSP integer solution values are presented in the last table column. For any given instance, whenever a particular LP relaxation bound could not be computed within the 3600 CPU seconds time limit imposed, character "-" appears in the corresponding entry.

These results show that the bounds for $\mathcal{P}_1^+$ are always better than or equal to corresponding DGR bounds and are, most of the time, weaker than their STR counterparts. However, STR bounds are typically very expensive to compute, with the time limit being exceeded for that formulation for 13 out of the 41 tested instances. Bounds for $\mathcal{P}_1$ significantly improve on those obtained for $\mathcal{P}_0^+$. However, only marginal gains are obtained while going from $\mathcal{P}_1$ to $\mathcal{P}_1^+$. This fact, as we will see below, partially explains why, with a very few exceptions, neither SABC nor IPBC significantly improve on the results obtained in Simonetti et al. (2011).

6.2 Comparison of Algorithms

In Table 2, we compare the CPU times needed by each of the six proposed algorithms and by the branch-and-cut methods in Simonetti et al. (2011) and Lucena et al. (2010). Whenever an algorithm attains the imposed time limit of 3600 seconds, a character "-" appears in the columns associated to that algorithm. Detailed computational results for the branch-and-cut, Benders decomposition and hybrid algorithms are presented, respectively, in Tables 3, 4 and 5 in Appendix B.

These results show that SABC and IPBC manage to solve, respectively, 33 and 31 out of the 41 tested instances (`120_d20` and `150_d30` are solved by SABC, but not by IPBC). When we focus on those instances that both algorithms manage to solve, it appears that IPBC outperforms SABC when densities $d \geq 50\%$ apply. Conversely, SABC outperforms IPBC when $d \leq 30\%$ holds. However, this somewhat general trend does not always hold; for example, for instance `120_d5`, where CPU times for SABC and IPBC are respectively 705.85 and 105.86 seconds. For that instance, since initial upper bounds are already very close to optimal solution values, just a few probing iterations are required for IPBC to find a proven optimal solution.

We now focus on comparing SABC and a preliminary version of the same algorithm investigated in Simonetti et al. (2011). Although the algorithm in Simonetti et al. (2011) already attempts to heuristically separate lifted GSECs (21), SABC goes further in that direction and additionally attempts to separate strengthened feasibil-

Table 1: Linear programming lower bounds

Instance	$\mathcal{P}_0^+$	$\mathcal{P}_1$	$\mathcal{P}_1^+$	DGR	STR	OPT
30_d10	8.60	14.40	14.50	14.12	14.34	15
30_d20	5.09	6.18	6.18	5.68	6.49	7
30_d30	2.92	3.60	3.60	3.05	3.50	4
30_d50	1.95	2.38	2.38	1.86	2.11	3
30_d70	1.36	1.83	1.83	1.30	2.00	2
50_d5	15.55	31.00	31.00	31.00	31.00	31
50_d10	9.17	10.68	10.73	10.37	11.15	12
50_d20	4.76	5.24	5.25	4.88	5.52	7
50_d30	3.28	3.69	3.69	3.26	3.93	5
50_d50	1.98	2.44	2.44	1.82	2.20	3
50_d70	1.45	1.84	1.84	1.31	2.00	2
70_d5	17.10	26.31	26.31	25.29	26.44	27
70_d10	9.82	11.23	11.24	10.90	11.40	13
70_d20	4.92	5.37	5.37	5.12	5.63	7
70_d30	3.27	3.62	3.62	3.20	3.86	5
70_d50	2.05	2.44	2.44	1.95	2.05	3
70_d70	1.43	1.91	1.91	1.35	2.00	2
100_d5	18.00	21.63	21.63	20.79	22.04	24
100_d10	10.05	10.98	10.98	10.62	11.07	13
100_d20	5.24	5.52	5.52	5.15	5.62	8
100_d30	3.37	3.74	3.74	3.33	-	6
100_d50	2.10	2.51	2.51	1.97	-	4
100_d70	1.45	1.94	1.94	1.36	2.05	3
120_d5	19.12	22.74	22.74	22.48	22.87	25
120_d10	9.79	10.66	10.66	10.33	10.87	13
120_d20	5.14	5.35	5.35	5.07	-	8
120_d30	3.40	3.76	3.76	3.31	-	6
120_d50	1.99	2.49	2.49	1.37	2.15	4
120_d70	1.44	1.92	1.92	-	-	3
150_d5	19.60	21.72	21.73	21.35	21.94	26
150_d10	10.27	10.69	10.65	10.56	10.84	14
150_d20	5.05	5.37	5.37	4.95	-	9
150_d30	3.42	3.81	3.81	3.33	-	6
150_d50	1.98	2.47	2.47	1.90	-	4
150_d70	1.44	1.99	1.99	1.37	-	3
200_d5	20.35	22.52	22.52	22.17	22.69	27
200_d10	10.16	10.53	10.56	10.39	-	16
200_d20	4.95	5.26	5.26	4.87	-	9
200_d30	3.35	3.77	3.77	3.23	-	7
200_d50	2.01	2.53	2.53	1.93	-	4
200_d70	1.44	2.00	2.00	1.37	2.03	3

Character "-" indicates that the LP relaxation could not be evaluated within 3600 seconds

ity cuts (12) and cutset inequalities (22)-(23). In spite of that, the two algorithms appear to perform similarly and the additional inequalities separated by SABC do not seem to pay off computationally.

Before comparing SABC to DGR, we remark that the DGR algorithm in Lucena et al. (2010) was tested on a 3.00Ghz Intel XEON X5472 based machine with 16 GBytes of RAM memory. Nevertheless, for comparison purposes, the differences in the machines used here and in Lucena et al. (2010) do not favor our computational results. Therefore, for any test instance that could not be solved in less than 3600 time seconds by DGR, with the machine considered in Lucena et al. (2010), a character "-" also appears in its corresponding CPU time entry. From the computational results shown in Table 2, the two algorithms provide optimality certificates for essentially the same set of instances. Indeed, there are only three instances solved by DGR that SABC could not solve. However, DGR tends to be faster than SABC, sometimes much faster, for low density instances, while the reverse is true for high density ones. In some cases, one algorithm turns out to be one order of magnitude faster than the other (see, for example, results for instances `120_d5` and `200_d50`).

We now discuss how the two Benders decomposition algorithms compare between each other. Optimality certificates are produced by SABE and IPBE, respectively, for 37 and 38 out of the 41 tested instances. This translates into the highest success rates attained for the algorithms investigated in this study. Only `50_d5`, `70_d5` and `200_d10` cannot be solved to optimality by either algorithm, while only SABE fails to solve `200_d5`. As one may observe from the results in Table 4, and as it is normally the case for Benders decomposition algorithms, SABE and IPBE usually perform well if just a few feasibility cuts are required to attain optimality. For the instances in our test bed, IPBE is usually faster than SABE. Additionally, four new optimality certificates are attained by IPBE for instances where $n = 200$ and $d \leq 30$. Although `200_d10` is not solved by neither IPBE nor by SABE within the time limit imposed, these algorithms respectively require 29450 and 24550 CPU seconds to find the proven optimal solution ($z = 16$). Instance `200_d5`, which is not solved by SABE within the time limit, is actually solved by that algorithm after 4460 seconds. These two instances cannot be solved by branch-and-cut algorithms, including those found in the literature, even after relaxing the time limit constraints.

In this study, with five exceptions, namely instances `30_d10`, `50_d5`, `50_d10`, `70_d5` and `100_d5`, Benders decomposition algorithms outperform branch-and-cut algorithms. Although the Benders decomposition algorithms are, in many cases, up to four orders of magnitude faster than branch-and-cut algorithms, they require far more time to solve three of these five instances and do not manage to solve the other two. Instances `30_d10`, `50_d5`, `50_d10` and `70_d5`, however, are solved quite easily, within two seconds, by the branch-and-cut algorithms. On the other hand, larger previously unsolved MCDSP instances can now be solved to optimality by our Benders decomposition algorithms.

We now discuss the computational results obtained by the hybrid algorithms.

SAHY and IPHY solve similar sets of instances with, respectively, 35 and 36 instances out of the 41 tested instances (instance v70_d5 was solved by IPHY, but not by SAHY). Considering the instances solved to optimality, once again, the iterative probing strategy produces much better results, in terms of average CPU times. For the instances that are not solved within the time limit by at least one of the two approaches, IPHY always finds a better or at least equal upper bound (see the results for instances v70_d5, v150_d5 and v200_d5 in Table 5, for which the upper bounds are better for IPHY).

Although the Benders algorithms solve more instances to optimality than the hybrid algorithms, the latter usually perform better for the instances with 100 vertices or less for which the Benders methods need a large number of iterations. When the number of vertices is larger, the hybrid algorithms struggle, in a similar way, but not to the same extent, as the branch-and-cut algorithms do. This is explained by the fact that the cutting-plane phase at the root node is, by itself, computationally heavy. For such large instances, the benefits implied by the stronger lower bounds provided by the master problems under the hybrid framework do not pay off, given the additional amount of CPU time needed to solve the separation problems and to deal with the increase in model size.

It is noteworthy that, sometimes, when the initial upper bounds provided by the heuristic method are already the optimal values, the Benders algorithms need less iterations than the hybrid methods, despite the fact that each master problem in SAHY and IPHY separates constraints that reinforce connectivity. For example, the results for instance v70d_30 in Tables 4 and 5 show that SABE and IPBE need just one iteration to solve this instance, while SAHY and IPHY require two iterations.

During our computational experiments, we also evaluated another variant of the hybrid algorithms discussed here. This variant performs the same cutting-plane procedure, but only at the root node of the first Benders master problem. Subsequent Benders master problems are initialized with the cuts generated when solving the first Benders master problem, but no additional inequalities are generated. Compared to SAHY and IPHY, the stand-alone and iterative probing implementations of this variant obtain slightly better results for the instances with more than 100 vertices, but are dominated by SAHY and IPHY for the instances with 100 vertices or less for which Benders algorithms typically perform poorly, namely instances v50d_5, v70d_5 and 100d_5.

To summarize, our computational experiments show that our stand-alone branch-and-cut algorithm is competitive with other branch-and-cut algorithms in the literature, while better overall results are obtained by the iterative probing variant of Benders decomposition, although very sparse instances are solved much faster by the branch-and-cut method. The iterative probing version of the hybrid algorithm provides a robust method: although it is rarely the fastest algorithm for any given instance, it is rarely the worst. In fact, for instances with up to 120 vertices, it is the only algorithm among the six we suggested that proves optimality for all the

instances. For instances with 150 and 200 vertices, the iterative probing version of the Benders algorithm gives the best results among all algorithms, including those proposed in the literature.

7 Conclusions

In this paper, we presented exact algorithms for solving the minimum connected dominating set problem (MCDSP). Two fundamental approaches were described: Benders decomposition and branch-and-cut algorithms. Hybrid algorithms were also developed to take advantage of the best features of both methods. Two variants of the resulting three approaches were designed: a stand-alone version and an iterative probing variant. The latter variant is based on a simple property of the MCDSP, which states that if no connected dominating set of a given cardinality $d > 0$ exists, then there are no connected dominating set of cardinality $d - 1$. Overall, six exact algorithms were developed and tested: the stand-alone and iterative probing variants of the Benders decomposition, the branch-and-cut and the hybrid methods. Our computational experiments showed that the iterative probing variant of Benders and hybrid algorithms performed well on our set of tested instances: for instances with 120 vertices or less, the iterative probing hybrid method is the only algorithm among the six suggested ones that proved optimality for all the instances, while for instances with more than 120 vertices, the iterative probing Benders approach provided the best results among the six algorithms.

Future work include the development of specialized separation algorithms for strengthened GSECs (21), cut constraints (22) and k-partition inequalities (23), which, in the current version of the branch-and-cut algorithm, are generated during the separation procedures for GSECs (16). We also want to investigate the possibility of strengthening the formulation of the Benders master problems by incorporating valid inequalities from the set covering polytope. Finally, the Benders algorithm, especially in its iterative probing variant, provided very good results on our tested instances, being extremely effective on dense instances. It would be interesting to explore the behavior of a similar approach on other graph optimization problems.

Acknowledgments

Bernard Gendron is partially funded by the Natural Sciences and Engineering Council of Canada (NSERC) Discovery Grants Program. Abilio Lucena is partially funded by CNPq grant 310561/2009-4 and FAPERJ grant E26-110.552/2010. Alexandre Salles da Cunha is partially funded by CNPq grants 302276/2009-2, 477863/2010-8 and FAPEMIG PRONEX APQ-01201-09. Luidi Simonetti is partially funded by FAPERJ and CNPq grants 483.243/2010-8, 304793/2011-6. All sources of financial support are gratefully acknowledged.

References

Asavathiratham, C., S. Roy, B. Lesieutre, G. Verghese. 2001. The influence model. *IEEE Control Systems* **21(6)** 52–64.

Balasundaram, B., S. Butenko. 2006. Graph domination, coloring and cliques in telecommunications. *Handbook of Optim. in Telecomm*. Springer, 865–890.

Berge, C. 1973. *Graphs and Hypergraphs*. North Holland.

Borndörfer, R. 1998. Aspects of set packing, partitioning and covering. Ph.D. thesis, Konrad-Zuse-Zentrum fur Informationstechnik, Berlin.

Chen, S., I. Ljubić, S. Raghavan. 2010. The regenerator location problem. *Networks* **55(3)** 205–220.

Chopra, S., M.R. Rao. 1994a. The Steiner tree problem I: Formulations, compositions and extensions of facets. *Math. Programming* **64(1-3)** 209–229.

Chopra, S., M.R. Rao. 1994b. The Steiner tree problem II: Properties and classes of facets. *Math. Programming* **64(1-3)** 231–246.

Codato, G., M. Fischetti. 2006. Combinatorial Benders' cuts for mixed-integer linear programming. *Oper. Res.* **54(4)** 756–766.

Cooper, C., R. Klasing, M. Zito. 2005. Lower bounds and algorithms for dominating sets in Web graphs. *Internet Math.* **2(3)** 275–300.

Domingos, P., M. Richardson. 2001. Mining the network value of customers. *Proc. of the Seventh Internat. Conf. on Knowledge Discovery and Data Mining*.

Eubank, S., H. Guclu, V.S.A. Kumar, M.V. Marathe, A. Srinivasan, Z. Toroczkai, N. Wang. 2004. Modelling disease outbreaks in realistic urban social networks. *Nature* **429(6988)** 180–184.

Fujie, T. 2003. An exact algorithm for the maximum leaf spanning tree problem. *Comput. Oper. Res.* **30(13)** 1931–1944.

Fujie, T. 2004. The maximum-leaf spanning tree problem: formulations and facets. *Networks* **43(4)** 212–223.

Goldenberg, J., B. Libai, E. Muller. 2001. Using complex systems analysis to advance marketing theory development. *Acad. of Marketing Science Rev.* **9** 1–19.

Guha, S., S. Khuller. 1998. Approximation algorithms for connected dominating sets. *Algorithmica* **20(4)** 374–387.

Hooker, J.N., G. Ottosson. 2003. Logic-based Benders decomposition. *Math. Programming* **96(1)** 33–60.

Kruskal, J.B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *P. Am. Math. Soc.* **7** 48–50.

Liang, X., Z. Sencun. 2007. A feasibility study on defending against ultra-fast topological worms. *Proc. of The Seventh IEEE Intern. Conf. on Peer-to-Peer Comput. (P2P'07)*.

Liu, C. L. 1968. *Introduction to Combinatorial Mathematics*. McGraw-Hill.

Lucena, A., N. Maculan, L. Simonetti. 2010. Reformulation and solution algorithms for the maximum leaf spanning tree problem. *Comput. Management Sci.* **7(3)** 289–311.

Marathe, M.V., H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz. 1995. Simple heuristics for unit disc graphs. *Networks* **25(2)** 59–68.

Milenković, T., V. Memišević, A. Bonato, N. Pržulj. 2011. Dominating biological networks. *PLoS ONE* **6(8)**.

Padberg, M.W., L. Wolsey. 1983. Trees and cuts. *Ann. Discrete Math.* **17** 511–517.

Rogers, E.M. 2003. *Diffusion of Innovations*. Free Press.

Simonetti, L., A.S. da Cunha, Lucena A. 2011. The minimum connected dominating set problem: Formulation, valid inequalities and branch-and-cut algorithm. *Lecture Notes in Computer Science* **6701** 162–169. Network Optim. - Proc. of the 5th Internat. Network Optim. Conf., Hamburg.

Stanley, E. A. 2006. Social networks and mathematical modelling. *Connections* **27(1)** 43–49.

Valente, T. 1995. *Network Models of the Diffusion of Innovations*. Hampton Press.

Wang, F., H. Du, E. Camachoa, K. Xua, W. Lee, Y. Shi, S. Shan. 2011. On positive influence dominating sets in social networks. *Theoret. Comp. Sci.* **412(3)** 265–269.

Appendix A: Heuristic Method

All algorithms introduced in this paper, as well as those they are compared to, are initialized with an MCDSP upper bound provided by the dynamic greedy heuristic introduced in Lucena et al. (2010). That heuristic, in spite of being part of a MLSTP paper, is actually geared into solving the MCDSP. It works with two sets: $\mathcal{D}$, to represent vertices in a connected dominating set and $\mathcal{L}$, to represent those vertices which have at least one neighbor in $\mathcal{D}$. The procedure is initialized by setting $\mathcal{D} = \{v\}$ and $\mathcal{L} = \Gamma_v \setminus \{v\}$ for any $v \in V$. Then, the basic operation performed at each iteration is to try to push vertices from $\mathcal{L}$ into $\mathcal{D}$, until a connected dominating set is found. Assuming that i is moved from $\mathcal{L}$ to $\mathcal{D}$, in the next iteration we have: $\mathcal{L} \leftarrow \mathcal{L} \setminus \{i\} \cup (\Gamma_i \setminus (\mathcal{D} \cup \mathcal{L}))$ and $\mathcal{D} \leftarrow \mathcal{D} \cup \{i\}$. Preference is given to include in $\mathcal{D}$ vertices with as many neighbors as possible, not already included in $\mathcal{D} \cup \mathcal{L}$. The heuristic stops when $V = \mathcal{L} \cup \mathcal{D}$. At that point, $\mathcal{D}$ defines a connected dominating set.

In our implementation, the heuristic is executed n times. In each one, set $\mathcal{D}$ is initialized with a different vertex $v \in V$. Therefore the procedure we actually implemented could be cast as a multi-start version of the greedy heuristic in Lucena et al. (2010).

Appendix B: Detailed Computational Results

First column entries in Table 3 identify the tested instances. Initial upper bounds, under the heading IUB, appear in the second column. These bounds, computed

with the heuristic method described in Appendix A, are used to initialize all exact algorithms considered here. For the next two columns, respectively under the headings UB and $t(s)$, the following computational results are presented for SABC: the best upper bound available on termination and the CPU time spent, in seconds. Whenever the time limit is reached and the instance is left unsolved, character "-" appears in the corresponding $t(s)$ entry. The next three columns apply to IPBC. The first of them shows, under the heading Iter, the number of probing iterations carried out, i.e., the number of times a new best incumbent feasible solution was found and, consequently, the right-hand-side of the d-cut equation (5) had to be reduced. Finally, the last two columns in Table 3, respectively, apply to the branch-and-cut algorithm in Simonetti et al. (2011) and the DGR algorithm in Lucena et al. (2010). They indicate the CPU time, in seconds, taken by these algorithms to find proven optimal solutions, respectively for the MCDSP and the MLSTP. Given that the same machine was used here and in Simonetti et al. (2011), the previously defined CPU time limit directly applies to the algorithm in Simonetti et al. (2011).

Computational results for Benders decomposition algorithms SABE and IPBE appear in Table 4. For that table, with the single exception of the Iter heading, definitions previously introduced for Tables 1 and 3 apply. Entries under that heading now accumulate the number of Benders master problems (BMP) that had to be solved before the incumbent solution was proven to be optimal or else when the time limit was reached. Additionally, they also accumulate the number of times a feasibility cut had to be appended to the master problem. Table 5 presents the computational results for the hybrid algorithms SAHY and IPHY; the format is identical to that of Table 4, as well as the meanings of each column entries.

Table 2: Computational results: CPU time comparisons, in seconds

Instance	Benders		Branch-and-cut		Hybrid		Literature		
	SABE	IPBE	SABC	IPBC	SAHY	IPHY	prev. SABC		DGR
v30_d10	41.84	24.84	0.02	0.02	12.79	8.14	0.01		0.01
v30_d20	0.01	0.00	0.02	0.03	0.00	0.02	0.02		0.10
v30_d30	0.02	0.02	0.05	0.05	0.04	0.02	0.05		0.03
v30_d50	0.00	0.00	0.01	0.01	0.01	0.01	0.04		0.08
v30_d70	0.01	0.00	0.02	0.01	0.01	0.00	0.02		0.01
v50_d5	-	-	0.02	0.01	22.32	9.44	0.02		0.01
v50_d10	12.42	1.66	0.58	0.20	2.67	1.64	0.42		0.36
v50_d20	0.21	0.08	0.74	0.86	0.41	0.20	0.66		1.32
v50_d30	0.19	0.10	0.24	0.23	0.47	0.30	0.25		1.21
v50_d50	0.00	0.00	0.23	0.06	0.01	0.01	0.25		0.51
v50_d70	0.00	0.00	0.25	0.02	0.01	0.02	0.29		0.04
v70_d5	-	-	1.10	0.26	-	290.36	1.42		0.26
v70_d10	1.05	1.38	14.23	5.42	25.11	1.26	34.29		4.73
v70_d20	0.37	0.18	2.01	1.74	1.16	0.58	2.16		16.30
v70_d30	0.54	0.22	1.13	0.93	0.84	0.36	1.00		2.90
v70_d50	0.01	0.02	0.63	0.40	0.02	0.02	0.70		1.33
v70_d70	0.01	0.01	0.99	0.04	0.03	0.02	0.79		1.92
v100_d5	2542.09	1963.44	38.74	65.05	970.48	35.89	342.25		12.50
v100_d10	0.50	0.33	17.15	35.84	2.67	1.70	32.11		9.36
v100_d20	1.88	1.28	205.92	534.00	6.48	2.67	174.93		86.16
v100_d30	3.83	2.48	209.20	275.29	11.16	4.45	193.65		258.15
v100_d50	1.57	0.77	40.28	27.42	3.27	1.60	35.41		132.55
v100_d70	1.57	0.03	12.93	12.46	1.58	0.92	12.03		154.10
v120_d5	3.38	10.44	705.85	105.86	1118.58	27.90	-		2.65
v120_d10	3.74	3.44	-	-	56.25	18.67	-		65.49
v120_d20	5.02	3.78	828.28	-	16.46	8.34	610.89		393.47
v120_d30	5.26	4.43	496.42	1039.94	14.14	7.62	475.54		653.70
v120_d50	4.19	2.54	161.80	153.47	8.73	4.60	168.55		815.64
v120_d70	2.26	0.04	34.39	26.18	2.84	2.21	31.67		356.31
v150_d5	1047.98	259.40	-	-	-	-	-		2954.00
v150_d10	51.11	28.39	-	-	651.85	195.35	-		3247.89
v150_d20	366.37	273.03	-	-	2117.74	902.43	-		-
v150_d30	21.13	11.27	2077.29	-	34.62	24.71	1954.00		2317.35
v150_d50	7.84	5.78	535.38	342.95	17.54	10.81	481.61		2756.36
v150_d70	4.28	0.06	51.18	45.94	5.05	2.95	43.75		1828.86
v200_d5	-	1901.71	-	-	-	-	-		-
v200_d10	-	-	-	-	-	-	-		-
v200_d20	1687.24	1943.93	-	-	-	-	-		-
v200_d30	3208.45	1848.78	-	-	-	-	-		-
v200_d50	24.79	19.31	2247.33	1279.91	41.00	28.55	2249.43		20155.00
v200_d70	10.52	0.13	334.21	261.65	9.00	5.61	271.91		8154.13

Note: "prev. SABC" stands for the branch-and-cut algorithm in Simonetti et al. (2011) and "DGR" for the branch-and-cut algorithm in Lucena et al. (2010), based on formulation DGR; character "-" indicates that the time limit of 3600 seconds was achieved and the instance was not solved

Table 3: Detailed computational results: branch-and-cut algorithms

		Current study					CPU time (s) - Literature	
		SABC		IPBC			prev. SABC	DGR
Instance	IUB	UB	$t(s)$	Iter	UB	$t(s)$		
30_d10	15	15	0.02	1	15	0.02	0.01	0.01
30_d20	7	7	0.02	1	7	0.03	0.02	0.1
30_d30	5	4	0.05	2	4	0.05	0.05	0.03
30_d50	3	3	0.01	1	3	0.01	0.04	0.08
30_d70	2	2	0.02	1	2	0.01	0.02	0.01
50_d5	31	31	0.02	1	31	0.01	0.02	0.01
50_d10	13	12	0.58	2	12	0.20	0.42	0.36
50_d20	7	7	0.74	1	7	0.86	0.66	1.32
50_d30	5	5	0.24	1	5	0.23	0.25	1.21
50_d50	3	3	0.23	1	3	0.06	0.25	0.51
50_d70	2	2	0.25	1	2	0.02	0.29	0.04
70_d5	29	27	1.1	3	27	0.26	1.42	0.26
70_d10	14	13	14.23	2	13	5.42	34.29	4.73
70_d20	7	7	2.01	1	7	1.74	2.16	16.3
70_d30	5	5	1.13	1	5	0.93	1	2.9
70_d50	3	3	0.63	1	3	0.40	0.7	1.33
70_d70	2	2	0.99	1	2	0.04	0.79	1.92
100_d5	25	24	38.74	2	24	65.05	342.25	12.5
100_d10	13	13	17.15	1	13	35.84	32.11	9.36
100_d20	8	8	205.92	1	8	534.00	174.93	86.16
100_d30	6	6	209.2	1	6	275.29	193.65	258.15
100_d50	4	4	40.28	1	4	27.42	35.41	132.55
100_d70	3	3	12.93	1	3	12.46	12.03	154.1
120_d5	26	25	705.85	2	25	105.86	-	2.65
120_d10	15	13	-	1	15	-	-	65.49
120_d20	8	8	828.28	1	8	-	610.89	393.47
120_d30	6	6	496.42	1	6	1039.94	475.54	653.7
120_d50	4	4	161.8	1	4	153.47	168.55	815.64
120_d70	3	3	34.39	1	3	26.18	31.67	356.31
150_d5	27	27	-	1	27	-	-	2954
150_d10	15	15	-	1	15	-	-	3247.89
150_d20	9	9	-	1	9	-	-	-
150_d30	6	6	2077.29	1	6	-	1954	2317.35
150_d50	4	4	535.38	1	4	342.95	481.61	2756.36
150_d70	3	3	51.18	1	3	45.94	43.75	1828.86
200_d5	29	29	-	1	29	-	-	-
200_d10	16	16	-	1	16	-	-	-
200_d20	9	9	-	1	9	-	-	-
200_d30	7	7	-	1	7	-	-	-
200_d50	4	4	2247.33	1	4	1279.91	2249.43	20155
200_d70	3	3	334.21	1	3	261.65	271.91	8154.13

Note: "prev. SABC" stands for the branch-and-cut algorithm in Simonetti et al. (2011) and DGR for the branch-and-cut algorithm in Lucena et al. (2010), based on formulation DGR

Table 4: Detailed computational results: Benders decomposition algorithms

		SABE			IPBE		
Instance	IUB	Iter	UB	$t(s)$	Iter	UB	$t(s)$
30_d10	15	230	15	41.84	328	15	24.84
30_d20	7	2	7	0.01	2	7	0
30_d30	5	5	4	0.02	3	4	0.02
30_d50	3	1	3	0	1	3	0
30_d70	2	1	2	0.01	1	2	0
50_d5	31	439	31	-	724	31	-
50_d10	13	68	12	12.42	37	12	1.66
50_d20	7	1	7	0.21	1	7	0.08
50_d30	5	1	5	0.19	1	5	0.1
50_d50	3	1	3	0	1	3	0
50_d70	2	1	2	0	1	2	0
70_d5	29	379	29	-	2383	29	-
70_d10	14	5	13	1.05	17	13	1.38
70_d20	7	1	7	0.37	1	7	0.18
70_d30	5	1	5	0.54	1	5	0.22
70_d50	3	1	3	0.01	1	3	0.02
70_d70	2	1	2	0.01	1	2	0.01
100_d5	25	367	24	2542.09	723	24	1963.44
100_d10	13	1	13	0.5	1	13	0.33
100_d20	8	1	8	1.88	1	8	1.28
100_d30	6	1	6	3.83	1	6	2.48
100_d50	4	1	4	1.57	1	4	0.77
100_d70	3	1	3	1.57	1	3	0.03
120_d5	26	5	25	3.38	49	25	10.44
120_d10	15	3	13	3.74	6	13	3.44
120_d20	8	1	8	5.02	1	8	3.78
120_d30	6	1	6	5.26	1	6	4.43
120_d50	4	1	4	4.19	1	4	2.54
120_d70	3	1	3	2.26	1	3	0.04
150_d5	27	58	26	1047.98	55	26	259.4
150_d10	15	3	14	51.11	5	14	28.39
150_d20	9	3	9	366.37	3	9	273.03
150_d30	6	1	6	21.13	1	6	11.27
150_d50	4	1	4	7.84	1	4	5.78
150_d70	3	1	3	4.28	1	3	0.06
200_d5	29	13	29	-	49	27	1901.71
200_d10	16	1	16	-	1	16	-
200_d20	9	1	9	1687.24	1	9	1943.93
200_d30	7	2	7	3208.45	2	7	1848.78
200_d50	4	1	4	24.79	1	4	19.31
200_d70	3	1	3	10.52	1	3	0.13

Table 5: Detailed computational results: hybrid algorithms

		SAHY			IPHY		
Instance	IUB	Iter	UB	$t(s)$	Iter	UB	$t(s)$
v30_d10	15	15	15	12.79	37	15	8.14
v30_d20	7	3	7	0.00	3	7	0.02
v30_d30	5	6	4	0.04	3	4	0.02
v30_d50	3	2	3	0.01	2	3	0.01
v30_d70	2	2	2	0.01	2	2	0.00
v50_d5	31	11	31	22.32	12	31	9.44
v50_d10	13	6	12	2.67	13	12	1.64
v50_d20	7	2	7	0.41	2	7	0.20
v50_d30	5	2	5	0.47	2	5	0.30
v50_d50	3	2	3	0.01	2	3	0.01
v50_d70	2	2	2	0.01	2	2	0.02
v70_d5	29	66	29	-	89	27	290.36
v70_d10	14	8	13	25.11	4	13	1.26
v70_d20	7	2	7	1.16	2	7	0.58
v70_d30	5	2	5	0.84	2	5	0.36
v70_d50	3	2	3	0.02	2	3	0.02
v70_d70	2	2	2	0.03	2	2	0.02
v100_d5	25	22	24	970.48	24	24	35.89
v100_d10	13	2	13	2.67	2	13	1.70
v100_d20	8	2	8	6.48	2	8	2.67
v100_d30	6	2	6	11.16	2	6	4.45
v100_d50	4	2	4	3.27	2	4	1.60
v100_d70	3	2	3	1.58	2	3	0.92
v120_d5	26	26	25	1118.58	15	25	27.90
v120_d10	15	6	13	56.25	6	13	18.67
v120_d20	8	2	8	16.46	2	8	8.34
v120_d30	6	2	6	14.14	2	6	7.62
v120_d50	4	2	4	8.73	2	4	4.60
v120_d70	3	2	3	2.84	2	3	2.21
v150_d5	27	19	27	-	38	26	-
v150_d10	15	4	14	651.85	4	14	195.35
v150_d20	9	4	9	2117.74	4	9	902.43
v150_d30	6	2	6	34.62	2	6	24.71
v150_d50	4	2	4	17.54	2	4	10.81
v150_d70	3	2	3	5.05	2	3	2.95
v200_d5	29	2	29	-	7	28	-
v200_d10	16	1	16	-	1	16	-
v200_d20		1	9	-	1	9	-
v200_d30	7	2	7	-	2	7	-
v200_d50	4	2	4	41.00	2	4	28.55
v200_d70	3	2	3	9.00	2	3	5.61