Exact and heuristic algorithms for the weighted total domination problem

Eduardo Álvarez-Miranda*[1,2] and Markus Sinnl†[3]

[1]*Department of Industrial Engineering, Faculty of Engineering, Universidad de Talca, Campus Curicó, Chile*
[2]*Instituto Sistemas Complejos de Ingeniería ISCI, Chile*
[3]*Institute of Production and Logistics Management, Johannes Kepler University Linz, Linz, Austria*

Abstract

Dominating set problems are among the most important class of combinatorial problems in graph optimization, from a theoretical as well as from a practical point of view. In this paper, we address the recently introduced (minimum) weighted total domination problem. In this problem, we are given an undirected graph with a vertex weight function and an edge weight function. The goal is to find a total dominating set D in this graph with minimal weight. A total dominating set D is a subset of the vertices such that every vertex in the graph, including vertices in D, is adjacent to a vertex in D. The weight is measured as the sum of all vertex weights of vertices in D, plus the edge weights in the subgraph induced by D, plus for each vertex not in D the minimum weight of an edge from it to a vertex in D.

In this paper, we present two new Mixed-Integer Programming models for the problem, and design solution frameworks based on them. These solution frameworks also include valid inequalities, starting heuristics and primal heuristics. In addition, we also develop a genetic algorithm, which is based on a greedy randomized adaptive search procedure version of our starting heuristic.

We carry out a computational study to assess the performance of our approaches when compared to the previous work for the same problem. The study reveals that our exact solution algorithms are up to 500 times faster compared to previous exact approaches and instances with up to 125 vertices can be solved to optimality within a timelimit of 1800 seconds. Moreover, the presented genetic algorithm also works well and often finds the optimal or a near-optimal solution within a short runtime. Additionally, we also analyze the influence of instance-characteristics on the performance of our algorithms.

1. Introduction and motivation

Dominating set problems are among the most important class of combinatorial problems in graph optimization, from a theoretical as well as from a practical point of view. For a given graph $G = G(V, E)$, a subset $D \subset V$ of vertices is referred to as a *dominating set* if the remaining vertices, i.e., $V \setminus D$, are *dominated* by D according to a given topological relation (e.g., they are all adjacent to at least one vertex from D). Dominating set problems (also often called *domination problems* in graphs) have attracted the attention of computer scientists and applied mathematicians since the early 50s, and their close relation to covering and independent set problems has lead to the development of a whole research area (see, e.g., [25] and [3] for early references on domination problems).

*ealvarez@utalca.cl
†markus.sinnl@jku.at

There are many applications where set domination and related concepts play a central role, including school bus routing [32], communication networks [35], radio station location [7], social networks analysis [33], biological networks analysis [24], and also chess-problems like the five-queens problem [31]; see e.g., the book [12] for a comprehensive overview of domination problems. Variants of dominating set problems include e.g., the *connected dominating set problem* [6], the *(weighted) independent dominating set problem* [10, 27], among others (see, e.g., [17] for further variants of the dominating set problems).

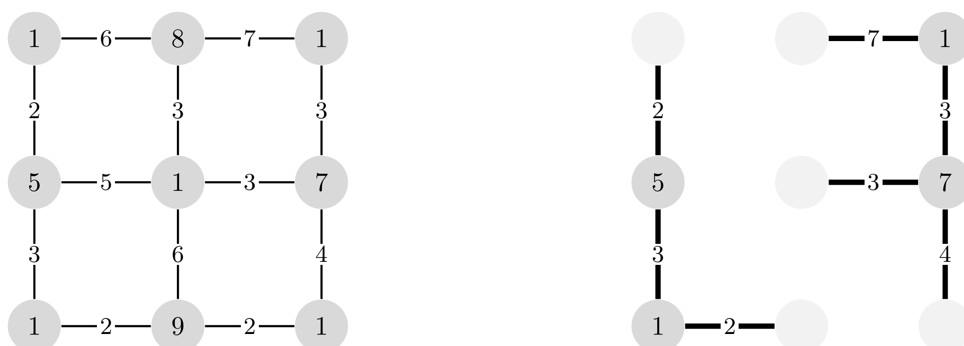
In this paper, we address the recently introduced *(minimum) weighted total domination problem (WTDP)* which is defined as follows.

Definition 1. *Let* $\mathbf{w} : V \to \mathbb{R}^{|V|}_{\geq 0}$ *be a vertex weight function, and let* $\mathbf{c} : E \to \mathbb{R}^{|E|}_{\geq 0}$ *be an edge weight function. The weighted total domination problem is the problem of finding a set* $D \subset V$*, such that every vertex in* V *(including the vertices from* D*) has at least one neighbor in* D *and the function*

$$w(D) = \sum_{i \in D} w_i + \sum_{e \in E(D)} c_e + \sum_{i \in V \setminus D} \min\{c_e \mid e : \{i,j\} \in E \textit{ and } j \in N(i) \cap D\}$$

is minimized, where $E(D) \subseteq E$ *corresponds to the set of edges* inside D*, and* $N(i) \subset V$ *corresponds to the set of neighboring vertices of vertex* $i \in V$*.*

For referring to the different components of the objective function, we denote $\sum_{i \in D} w_i$ as the *vertex selection costs*, $\sum_{e \in E(D)} c_e$ as the *internal edge costs* and $\sum_{i \in V \setminus D} \min\{c_e \mid e : \{i,j\} \in E$ and $j \in N(i) \cap D\}$ as the *external edge costs*. Figure 1 gives an exemplary instance of the WTDP, together with its optimal solution.



(a) Instance

(b) Optimal solution

Figure 1: Instance $I = (G = (V, E, \mathbf{c}, \mathbf{w})$ and optimal solution with weight $14 + 6 + 18 = 38$ (*vertex selection costs+internal edge costs+external edge costs*). We note that a solution consisting of all the vertices with weight one would not be feasible, as it is not a total dominating set, but only a dominating set.

We note that in the WTDP, we are not just concerned with the concept of domination, but with the stronger concept of *total domination*, which imposes that for each vertex $v \in D$, there is also a neighbor of v in D (i.e., the vertices of D also need to be dominated by D). The WTDP was introduced in [22] an is an extension of the (unweighted) total domination problem (TDP), resp., the vertex-weighted total domination problem. In the TDP, the objective function has $w_i = 1$ for all $i \in V$, and $c_e = 0$ for all $e \in E$. The optimal solution of the TDP for a given graph is called its *total domination number*. The TDP was introduced in the 1980s (see [4]) and is NP-hard in general graphs (see [20], for further details). The TDP has a rich history of research focusing on theoretical results, e.g., computational complexity and bounds for the domination number for certain graph classes, we refer the reader to the survey [15] and the book [16] for more details. Applications of total domination include the design of communication networks and the forming of committees [12, 14].

Contribution and Outline The WTDP was recently introduced in [22], where three Mixed-Integer Programming (MIP) formulations to solve the problem were presented and evaluated in a computational study. In this paper, we present two new MIP models for the problem, and design solution frameworks based on them. These solution frameworks also include valid inequalities, starting heuristics and primal heuristics. A genetic algorithm (GA) is also developed, which is based on a greedy randomized adaptive search procedure (GRASP) version of our starting heuristic. We carry out a computational study to assess the performance of our new approaches in comparison to the previous work by [22]. The study reveals that our algorithms are up to 500 times faster and instances with up to 125 vertices can be solved to optimality within a timelimit of 1800 seconds. Moreover, the presented heuristics (i.e., the GA, and just using the GRASP on its own) also work well and often find the optimal, or a near-optimal solution within a short runtime. Furthermore, we also analyze the influence of instance-characteristics on the performance of our algorithms.

The paper is organized as follows. In the reminder of this section, we give a short overview of the models introduced in [22]. In Section 2 we present our two new MIP models, together with valid inequalities. In Section 3 we discuss implementation details of the branch-and-cut algorithms we designed based on our new models, including a description of the starting and primal heuristics. In Section 4, we describe our genetic algorithm. Section 5 contains our computational study, and concluding remarks are provided in Section 6.

1.1 Revisiting the models of [22]

In the following, we give a brief overview of the three formulations for the WTDP presented in [22] (we denote them as (MA1), (MA2), (MA3)). We re-implemented these models and included them in our computational study, see Section 5.

Firstly, consider the following set of variables and constraints which are common to all formulations of [22] and that will also be part of our formulations. Let $\mathbf{x} \in \{0,1\}^{|V|}$ be a vector of binary variables, such that $x_i = 1$ if vertex $i \in V$ is taking as part of the (total) dominating set, and $x_i = 0$ otherwise. Constraints

$$\sum_{j \in N(i)} x_j \geq 1, \ \forall i \in V, \tag{TDOM}$$

ensure that the variables with $x_i = 1$ form a total dominating set. We observe that these constraints are already enough to define the set of feasible solutions. The remaining constraints in the presented models are used to correctly measure the objective function. Let $\mathbf{y} \in \{0,1\}^{|E|}$ be a vector of binary variables associated with the edges E. These variables will be used in all formulations except of (MA3), and they are used differently depending on the considered formulation. In (MA1), (MA2), they are used to measure the contribution of any edge $e = \{i,j\}$ on the objective function, for both the internal edge costs and the external edge costs. In contrast, in the new formulations presented in Section 2, these variables are only used for the internal edge costs, and the external edge costs are modeled in different ways.

Formulation (MA1) Let $\mathbf{z} \in \{0,1\}^{|E|}$ be a vector of binary variables, such that $z_{e=\{i,j\}} = \min\{x_i, x_j\}$ for every edge $e : \{i,j\} \in E$. For a given vertex $i \in V$, let $\delta(i) \subset E$ be the set of edges incident to i.

Formulation (MA1) is given by:

$$
\begin{aligned}
\text{(MA1)} \qquad & w^* = \min \sum_{i \in V} w_i x_i + \sum_{e \in E} c_e y_e && \text{(WTD1.1)} \\
\text{s.t.} \qquad & \text{(TDOM)} \\
& x_i + x_j \geq y_e, \ \forall e : \{i,j\} \in E && \text{(WTD1.3)} \\
& x_i \geq z_e \text{ and } x_j \geq z_e, \ \forall e : \{i,j\} \in E && \text{(WTD1.4)} \\
& z_e \geq x_i + x_j - 1, \ \forall e : \{i,j\} \in E && \text{(WTD1.5)} \\
& y_e \geq z_e, \ \forall e \in E && \text{(WTD1.6)} \\
& x_i + \sum_{e \in \delta(i)} y_e \geq 1, \ \forall i \in V && \text{(WTD1.7)} \\
& \mathbf{x} \in \{0,1\}^{|V|}, \ \mathbf{y} \in \{0,1\}^{|E|} \text{ and } \mathbf{z} \in \{0,1\}^{E}
\end{aligned}
$$

Formulation (MA2) Let M be a large constant (e.g., the maximum vertex-degree of a given instance). Compared to formulation (MA1), (MA2) gets rid of the $\mathbf{z}$-variables with the help of big-M-type constraints. Formulation (MA2) is given by:

$$
\begin{aligned}
\text{(MA2)} \qquad & w^* = \min \sum_{i \in D} w_i x_i + \sum_{e \in E(D)} c_e y_e \\
\text{s.t.} \qquad & \text{(TDOM), (WTD1.3) and (WTD1.7)} && \text{(WTD2.2)} \\
& y_e \leq x_i + x_j - 1, \ \forall e : \{i,j\} \in E && \text{(WTD2.3)} \\
& \sum_{e \in \delta(i)} y_e \leq 1 + M x_i, \ \forall i \in V && \text{(WTD2.4)} \\
& \mathbf{x} \in \{0,1\}^{|V|} \text{ and } \mathbf{y} \in \{0,1\}^{|E|}
\end{aligned}
$$

Formulation (MA3) Finally, formulation (MA3) also gets rid of the binary $\mathbf{y}$-variables, with the help of integer variables $\mathbf{q} \in \{0, 1, \dots, |V|L\}^{|V|}$, where L is a large constant, e.g., the maximum edge weight of a given instance. These variables measure for each $i \in V$ twice contribution to the objective of all the edge-weights of edges adjacent to i (again, for both the internal and external edge costs). Formulation (MA3) is given by:

$$
\begin{aligned}
\text{(MA3)} \qquad & w^* = \sum_{i \in V} \left(w_i x_i + \frac{1}{2} \cdot q_i \right) && \text{(WTD3.1)} \\
\text{s.t.} \qquad & \text{(TDOM)} \\
& q_i \geq 2 \left(c_e x_i - L x_i - \sum_{e' : \{i,j'\} \in E|_{c_{e'} \leq c_e}} L x_{j'} \right), \ \forall e : \{i,j\} \in E, \ \forall i \in V && \text{(WTD3.2)} \\
& q_i \geq \sum_{e : \{i,j\} \in E} c_e \left(x_i + x_j - 1 \right), \ \forall i \in V && \text{(WTD3.3)} \\
& \mathbf{x} \in \{0,1\}^{|V|} \text{ and } \mathbf{q} \in \{0, 1, \dots, |V|L\}^{|V|}
\end{aligned}
$$

2. Two new Mixed-Integer Programming formulations for the WTDP

In this section we present two alternative formulations that, as we show in Section 5, allow the design of algorithmic strategies that outperform the results presented in [22].

2.1 Formulation (F1) and valid inequalities

Let $\mathbf{x} \in \{0,1\}^{|V|}$ be defined as before, and $\mathbf{y} \in \{0,1\}^{|E|}$ be such that $y_{e:\{i,j\}} = 1$ if $x_i = 1$ and $x_j = 1$, and $y_{e:\{i,j\}} = 0$, otherwise, for every edge $e : \{i,j\} \in E$. Let $A = \{(i,j) \cup (j,i) \mid \forall e : \{i,j\} \in E\}$ be the set of bi-directed arcs associated with E, and let $c_{ij} = c_{ji} = c_e$ for all $e \in E$. In contrast to (MA1), we associate $\mathbf{z}$ with these directed arcs, instead of the undirected edges. Let $z_{ij} = 1$ if vertex $j \in V$ is adjacent to the dominating set vertex i through arc $(i,j) \in A$, and $z_{ij} = 0$ otherwise. These variables are used to measure the external edge costs. By using such strategy, the resulting formulation resembles to the formulations of the well-known uncapacitated facility location problem (UFL); we can interpret the set of vertices $j \in D$ as open facilities, we want the vertices $i \in V \setminus D$ be assigned to the facility with the cheapest assignment cost (see, e.g., [8, 19] for recent references on the UFL). Let $\delta^-(i)$ and $\delta^+(i)$ correspond to the set of *incoming* and *outgoing* arcs from and to vertex $i \in V$, respectively.

Using this notation, the WTDP can be formulated as follows:

$$
\begin{aligned}
\text{(F1)} \qquad w^* = \min & \sum_{i \in V} w_i x_i + \sum_{e \in E} c_e y_e + \sum_{(i,j) \in A} c_{ij} z_{ij} \\
\text{s.t.} \quad & \text{(TDOM)} \\
& x_i + \sum_{(j,i) \in \delta^-(i)} z_{ji} = 1, \ \forall i \in V && \text{(XZLINK1)} \\
& z_{ij} \leq x_i, \ \forall (i,j) \in A && \text{(XZLINK2)} \\
& y_e \geq x_i + x_j - 1, \ \forall e : \{i,j\} \in E && \text{(YZLINK)} \\
& \mathbf{x} \in \{0,1\}^{|V|}, \ \mathbf{y} \in \{0,1\}^{|E|} \text{ and } \mathbf{z} \in \{0,1\}^{|A|}.
\end{aligned}
$$

Constraints (XZLINK1) ensure for each $i \in V$, that either $i \in D$, or that it is covered by a $j \in D$. Constraints (XZLINK2) link the $\mathbf{z}$-variables and $\mathbf{x}$-variables. Together with the $\sum_{(i,j) \in A} c_{ij} z_{ij}$-part of the objective function, they ensure that the contribution of vertices $i \in V \setminus D$ is measured correctly (i.e., these are the external edge costs). Finally, constraints (YZLINK) and the $\sum_{e \in E} c_e y_e$-part in the objective function make sure that the contribution of edges $e : \{i,j\}$, where both $i, j \in D$, is measured correctly (i.e., these are the internal edge costs). We note that both variable-sets $\mathbf{y}$ and $\mathbf{z}$ can be relaxed to be continuous, as for binary $\mathbf{x}$, these variables are automatically binary.

Valid inequalities Next, we present three families of valid inequalities for (F1). Separation of these inequalities is discussed in Section 3.1.

Theorem 1. *Inequalities*

$$
y_{e:\{i,j\}} + z_{ij} \leq x_i, \quad \forall (i,j) \in A \qquad \text{(XZLINK2L)}
$$

are valid for (F1).

Proof. These inequalities are a lifted version of inequalities (XZLINK2). Validity follows from the fact, that in any feasible solution, either the edge $e : \{i,j\}$ or the arc (i,j) can be contained, and in both cases, this implies that $i \in D$. □

Theorem 2. *Inequalities*

$$
\sum_{e \in \delta(i)} y_e \geq x_i, \ \forall i \in V \qquad \text{(TDOMY)}
$$

are valid for (F1).

Proof. By definition of total domination, for each $i \in V$, at least one adjacent vertex $j \in N(i)$ must be in D. Thus, if $x_i = 1$, which means $i \in D$, at least one of the y_e-variables for $e \in \delta(i)$ must be one. □

For the next family of valid inequalities, we observe that constraints (YZLINK) together with inequalities $y_{e:\{i,j\}} \leq x_i$ (which are valid, but redundant in our case due to the minimization objective) and the binary constraints on $(\mathbf{x}, \mathbf{y})$ give the *boolean quadric polytope (BQP)* (see, e.g., [26]). Thus all inequalities valid for the BQP are also valid for our formulation. We note that there are many graph problems which can be either directly formulated using the BQP, or using the BQP and additional constraints (see, e.g., [1, 2, 23]). There is a huge number of families of valid inequalities known for the BQP, however, most of them are not useful within a branch-and-cut algorithm as there are no efficient separation procedures known for them (see, e.g., [21]). We thus just used the following inequalities known as *clique inequalities* in our algorithm.

Theorem 3. *Let $C \subset V$, such that $E(C) \subset E$ form a* clique. *The* clique inequalities

$$\sum_{e \in E(C)} y_e \geq \sum_{i \in C} x_i - 1 \tag{CLIQUE}$$

are valid for (F1).

Section 3.1 details how these valid inequalities are incorporated into our solution framework.

2.2 Formulation (F2) and valid inequalities

In formulation (F2), we use continuous variables $q_i \geq 0$, $i \in V$ to measure the external edge costs. This is done by exploiting a Benders decomposition scheme that allows projecting out the $\mathbf{z}$-variables, similar as it is done for the UFL (see, e.g., [8]). By doing so, we obtain a polynomial set of optimality cuts, which are detailed next (note that by adding a "dummy-arc"-variable z_{ii} with weight zero to formulation (F1), replacing x_i in (XZLINK1) with z_{ii} and adding a constraint $z_{ii} \leq x_i$ the connection to UFL becomes directly evident; in the following, we also provide a combinatorial argument for their correctness without the need for Benders decomposition). For ease of exposition, for a given vertex i, let $N'(i) = \{j_1, \ldots, j_k, \ldots, j_{|N(i)|}\}$ be the ordered set of adjacent vertices such that $c_{j_1 i} \leq \ldots \leq c_{j_k i} \leq \ldots \leq c_{j_{|N(i)|} i}$. Then the cuts for a given $i \in V$ are given by

$$q_i \geq c_{ki} - \sum_{k'=1}^{k-1} (c_{ki} - c_{k'i}) x_{k'} - c_{ki} x_i, \ \forall k \in \{1, \ldots, |N'(i)|\}. \tag{EXTCOSTS-i}$$

When $x_i = 0$, i.e., $i \in V \setminus D$, (EXTCOSTS-i) is similar to the Benders optimality cuts for the UFL and, therefore, these inequalities measure the external edge cost for vertex i. When $x_i = 1$, i.e., $i \in D$ (and thus i incurs in no external edge cost), the right hand side of the cuts is at most zero, due to $-c_{ki}x_i$ and, therefore, they are also correct. By replacing (XZLINK1) and (XZLINK2) with (EXTCOSTS-i), the WTDP can be formulated as

$$\begin{aligned}
\text{(F2)} \qquad & w^* = \min \sum_{i \in V} (w_i x_i + q_i) + \sum_{e \in E} c_e y_e \\
& \text{s.t.} \qquad \text{(TDOM)}, \text{(EXTCOSTS-}i\text{)}, \text{(YZLINK)} \\
& \qquad\qquad \mathbf{x} \in \{0,1\}^{|V|},\ \mathbf{y} \in \{0,1\}^{|E|} \text{ and } q_i \geq 0, \forall i \in V.
\end{aligned}$$

We note that $\mathbf{y}$-variables could also be projected out, however, the resulting optimality cuts would not have the same effective structure as (EXTCOSTS-i). Namely, as each $y_{e=\{i,j\}}$ links two vertices $i, j \in V$, the corresponding Benders subproblem for a fixed $\mathbf{x}$ would not decompose for each vertex.

Valid inequalities Inequalities (EXTCOSTS-i) can be lifted by using the $\mathbf{y}$-variables.

Theorem 4. *Let $i \in V$ and $k \in \{1, \ldots, |N'(i)|\}$. Then inequalities*

$$q_i \geq c_{ki} - \sum_{k'=1}^{k-1} (c_{ki} - c_{k'i}) x_{k'} - c_{ki} x_i + \sum_{k'=1}^{k-1} (c_{ki} - c_{k'i}) y_{e=\{k'i\}}, \tag{EXTCOSTS-i-L}$$

are valid for (F2).

Proof. When the $\mathbf{y}$-variables are zero, the inequalities are similar to (EXTCOSTS-i) and thus clearly valid. Now suppose some $y_{e=\{l,i\}}$ for some $1 \leq l \leq k-1$ is one. By definition of the variables, this means that both x_i and x_l are one, and thus on the right-hand-side (rhs) of the cut, we have $c_{ki}-(c_{ki}-c_{li})-c_{ki} = -(c_{ki}-c_{li}) < 0$. Thus, $(c_{ki} - c_{li})$ (which is the coefficient of y_e) can be added to the rhs, which then will be zero and the inequality still remains valid. The same reasoning also applies, if more y_e-variables are one. $\square$

Finally, we observe that inequalities (TDOMY) and (CLIQUE) presented for (F1) are also valid for (F2), as they are in the $(\mathbf{x}, \mathbf{y})$-space.

3. Implementation details of the branch-and-cut algorithms

In this section, we give implementation details of the branch-and-cut algorithms we designed based on (F1) and (F2).

3.1 Initialization and separation of cuts

We first describe how the valid inequalities are incorporated in our frameworks. We note that to design a successful branch-and-cut scheme, it is often crucial to carefully select which cuts to add, e.g., even if in theory the cuts improve the lower bound, they may lead to slow linear programming (LP)-relaxation solution times due to their density or numerical stability, which is detrimental to the node-throughput and thus to the overall performance of the branch-and-cut. We refer to [5, 36, 28] for recent works on theoretical and computational studies on the challenges of cutting-plane selection. In Section 5.2 we also provide computational results obtained when just adding individual families of valid inequalities to the formulations.

The lifted inequalities (XZLINK2L) are added at the initialization, by simply replacing their non-lifted counterpart (XZLINK2). The objective-cuts (EXTCOSTS-i), resp., their lifted version (EXTCOSTS-i-L) in (F2) are added for the five smallest values of c_{ki} for each $i \in V$ at initialization, and the remaining ones are then separated on-the-fly by enumeration. Inequalities (TDOMY) are also separated by enumeration.

Clique inequalities (CLIQUE) are separated heuristically. We observe that inequalities (YZLINK) are a special case of (CLIQUE) for $|C| = 2$. For each edge $e = \{i, j\} \in E$, we try to construct a violated inequality (CLIQUE) by greedily constructing a clique containing e. Thus, initially, let $C = \{i, j\}$. Let $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ be the LP-values at the current branch-and-cut node. We sort all vertices $k \in \cap_{i\in C} N(i)$ (i.e., all candidate vertices to grow the clique C) in descending order according to $|N(k)| \cdot (\tilde{x}_k + \epsilon)$, for $\epsilon = 0.0001$. Note that by adding any vertex k to C, the (potential) violation of the constructed clique inequality changes by $\tilde{x}_k - \sum_{i\in C} \tilde{y}_{e=\{i,k\}}$. Thus, we iterate through the sorted list of candidate vertices to increase C, and whenever this value is greater than ϵ for a given k, we add it to C, and repeat the procedure for this C. This is done, until no more vertex can be added to C. We then add the clique inequality for this C if it is violated. To speed-up separation, if an edge e is already contained in a clique inequality added during the current round of separation at a branch-and-cut node, we do not consider it in constructing additional clique inequalities.

In order to avoid overloading the LP-relaxation with cuts and to allow for a fast node-throughput in the branch-and-cut, we only separate inequalities at the root node and limit separation to ten rounds. Naturally, to ensure correctness when using (F2) violation of objective-cuts (EXTCOSTS-i), resp., their lifted version (EXTCOSTS-i-L), is also checked whenever an integer solution is obtained during the branch-and-cut. As inequalities (TDOMY) and in particular (CLIQUE) can become quite dense, especially if the instance graph has many edges, we use the option `UseCutFilter` provided by CPLEX (the chosen MIP-solver), when adding these cuts. With this option, CPLEX checks the cut with the same criteria (e.g., density) as it checks its own general purpose cuts, and adds it only if it determines that it is beneficial.

3.2 Starting and primal heuristic and local search

We implemented both a starting heuristic and a primal heuristic; the former gets called at the initialization while the latter gets called during the execution of the corresponding branch-and-cut algorithms. Both of these heuristics construct feasible solutions, which we then try to improve by applying a local search procedure The starting heuristic starts out with the solution $D^H = V$ consisting of the set of all vertices (which clearly is a feasible solution). We then greedily remove vertices from D^H as long as the solution remains feasible. Algorithm 1 details our starting heuristic.

At each iteration, we use a score $score_i$ for choosing the vertex to remove; this score gives for each vertex in D^H the improvement in objective solution value it would bring if it is removed. When removing a vertex, say i, its vertex weight v_i and the internal edge costs w_{ij} for $j \in D^H$ are not applicable anymore. On the other hand, we need to consider the new external edge cost for covering i and, moreover, we have to consider that all the vertices $j' \in V \setminus D^H$ that are covered by i up to that iteration now need to be covered by another vertex in D^H (thus for covering these vertices we will get new, similar or higher, external edge costs). We note that removing a vertex only causes local changes in the solution structure, thus we do not need to calculate $score_i$ for each vertex in D^H from scratch in every iteration. In particular, when node i gets removed, the score needs to be re-calculated only for neighboring nodes $j \in N(i)$ and for the corresponding neighbors $j' \in N(j)$ (removing i may change the external edge costs associated with such a j' as both i and j' share j as neighbor). We observe that verifying if D^H is still be a total dominating set after removing i (line 9 of Algorithm 1) can be done efficiently by storing the number $N_j^H = |N(j) \cap D^H|$ for each $j \in V$, i.e., the number of neighbors of j contained in D^H. At the begging of the algorithm execution, it holds $N_j^H = |N(j)|$, and whenever a vertex i' gets removed in the course of the algorithm, N_j^H gets decreased by one for each $j \in N(i)$. Therefore, $D^H \setminus \{i\}$ is still a total dominating set, if and only if $N_j^H > 1$ for each $j \in N(i)$.

The primal heuristic is guided by the $(\tilde{\mathbf{x}})$-values of the LP-relaxation at the current branch-and-cut node. First, we sort the vertices $i \in V$ in descending order according to $\tilde{x}_i$. Afterwards, ties are broken first by degree of the vertices (again in descending order), and if there remain ties, they are broken by vertex-index. Let *sorted* be the list of sorted vertices, $D^H = \emptyset$ (the solution to be constructed) and $covered = \emptyset$ (the list of vertices covered by D^H). To construct a heuristic solution, we iterate through *sorted* and whenever $|N(i) \cap (V \setminus covered)| > 0$ for the currently considered vertex i, i.e., i covers a vertex not yet covered by the current partial solution D^H, we add i to D^H and update *covered* by $covered \cup N(i)$. We stop when $covered = V$, i.e., D^H is a total dominating set and thus a feasible solution.

The local search procedure is shown in Algorithm 2. It uses two local search operators, namely adding a vertex i to the current solution D^H and removing a vertex i from the current solution D^H. The procedures `testAddVertex(`D^H`,`i`)` and `testRemoveVertex(`D^H`,`i`)` revert the change in objective function caused by adding/removing a vertex i. This can be done efficiently, as the changes caused by these moves are of a local nature, as described above (e.g., the test for the change caused by removing i is exactly the calculation of the score-function in Algorithm 1). We first try the add-move, and when this move cannot improve the current solution anymore, we try the remove-move. If it is successful, we go back to trying the add-move, if not, the local search terminates. We iterate through the vertices by their indexes, and if a move is possible, we apply it, and then restart (i.e., we use a *first improvement* strategy).

3.3 Branching priorities

For both (F1) and (F2), once the $\mathbf{x}$-variables are fixed to binary, the values of all the other variables (i.e., $(\mathbf{y}, \mathbf{z})$, resp., $(\mathbf{y}, \mathbf{q})$) automatically follow. We thus give branching priorities $100 \cdot |N(i)|$ to the $\mathbf{x}$-variables in the MIP-solver, CPLEX in our case (while the branching priorities of the other variables were left at their default value, i.e., zero).

4. A genetic algorithm

Genetic algorithms (GAs) are among the most prominent metaheuristic approaches for solving (combinatorial) optimization problems; we refer the reader to the book [18] for an overview on essential elements of this class of procedures. GAs have been developed for tackling set dominating problems. For instance, a hybrid GA has been developed in [13] for the *minimum dominating set problem*, where the GA methodology is

```
input  : instance (G = (V, E), (c, w)) of the WTDP
output: total dominating set D^H
D^H ← V
score_i ← −∞                // store score function for faster evaluation, −∞ indicates re-calculation needed
improvingMoveExists ← false
do
    improvingMoveExists ← false
    vertexToRemove ← null
    bestScore = 0
    for i ∈ D^H do
        if D^H \ {i} is not a total dominating set then
            continue
        if score_i = −∞ then                          // re-calculation of score needed
            score_i ← w_i                              // vertex costs saved
            for j ∈ N(i) ∩ D^H do
                score_i ← score_i + w_ij               // internal edge costs saved
            w* ← min_{j∈D^H} w_ij            // external edge cost for covering i updated
            score_i ← score_i − w*
            for j ∈ N(i) ∩ (V \ D^H) do       // external edge costs for vertices currently covered by i updated
                if i ∈ arg min_{j'∈D^H} w_jj' then
                    w*_j ← min_{j∈(D^H\{i})} w_ij   // external edge cost for covering j updated
                    score_i ← score_i − w*_j
        if score_i > bestScore then
            bestScore = score_i
            vertexToRemove ← i
            improvingMoveExists ← true
    if improvingMoveExists then
        D^H ← D^H \ {vertexToRemove}
        for ∀j ∈ N(vertexToRemove) do // score for the neigbors of vertexToRemove, and their neigbors need to be updated
            for ∀j' ∈ N(j) do
                score_j' ← −∞
while improvingMoveExists
```

Algorithm 1: Starting heuristic

combined with local search and intensification schemes; likewise, in [9] a parallelized GA is presented for the same problem. Further examples on GA-based approaches for related problems can be found in [34] for the *dominating tree problem*, and in [29] for the *minimum weight minimum connected dominating set problem*.

In their general setting, GAs explore the solution space by keeping a set of feasible solutions, denoted as *population*. Starting from an initial population, the algorithm iteratively creates a new population (i.e., new solutions) by typically using the following three (randomized) bio-inspired operators: *selection*, *mutation* and *crossover*. The *selection* operator selects a subset of the current population (according to a *fitness* value of each solution), from which (usually) pairs of solutions are taken and a *crossover* operator is applied to combine these pairs to create a new solutions. To these new solutions a *mutation* operator is applied, which randomly modifies the solution in order to keep the population diverse.

Algorithm 3 gives an outline of the genetic algorithm we developed for the WTDP. The initial population is constructed by using a generalized randomized adaptive search procedure (GRASP) version of our starting heuristic. GRASP is a general technique to generate (diverse) heuristic solutions by randomizing

input : total dominating set D^H
output: total dominating set D^H (with potentially better objective function value)

```
improvingMoveExists ← false
do
    improvingMoveExists ← false
    for i ∉ D^H do
        if testAddVertex(D^H,i) > 0 then
            D^H = D^H ∪ {i}
            improvingMoveExists ← true
            break
    if improvingMoveExists == false then
        for i ∈ D^H do
            if testRemoveVertex(D^H,i) > 0 then
                D^H = D^H \ {i}
                improvingMoveExists ← true
                break
while improvingMoveExists
```

Algorithm 2: Local search

the construction phase. For further details on GRASP, the reader is referred to the recent textbook [30]. In order to turn our starting heuristic into a GRASP, we add randomization to the choosing of the vertex with the best score (i.e., lines 22-24): If a vertex i has $score_i > bestScore$, we generate a random integer in $[0, 99]$ and only apply lines 22-24 if this integer is larger than a given value $cutoff$.

As *crossover* operator, we also use modifications of Algorithm 1, resp., the GRASP. In particular, for crossover between two solution D^1, D^2, we use the GRASP, and set $D^H \leftarrow D^1 \cup D^2$ as initial solution in line 1. For *mutation*, we generate a random integer m in a given range $[m_l, m_u]$ and then randomly remove m vertices from the current solution D^H. After removing these vertices, D^H may be infeasible, in order to make it feasible, we apply the same heuristic as our primal heuristic (with just the degree of vertices as sorting criterion, as of course we have no LP-values). After mutation, we also apply the local search procedure described in Algorithm 2. The newly obtained solutions are merged with the current population, and then the *populationSize* best are selected as the next generation, for a given value of *populationSize*. As a fitness value for selection, we use the objective function values of the solutions. In order to keep the population diverse, we keep at most one solution for each fitness value and size $|D^H|$ in the population (this is done by checking if the current population already contains a solution with the fitness value and size of the currently created solution, and if yes, the solution is discarded). To create the population, we run the GRASP *initialPopulationSize* times, for a given value of parameter *initialPopulationSize* and then select the *populationSize* best solutions.

We used the following parameter values in our implementation, these values were determined using some preliminary computational experiments: $initialPopulationSize = 100$, $populationSize = 40$, $cutoff = 30$, $[m_l, m_u] = [1, 4]$, and $nIterations = 20$.

5. Computational results

The branch-and-cut framework was implemented in C++ using CPLEX 12.9 as MIP solver and the genetic algorithm was also implemented in C++. The computational study was carried out on an Intel Xeon E5 v4 CPU with 2.5 GHz and 6GB memory using a single thread. All CPLEX parameters were left at default values (except branching priorities, see Section 3.3), and we set the timelimit for a run to 1800 seconds (similar to the timelimit in [22]).

```
input  : instance I = (G = (V, E), (c, w)) of the WTDP, parameters
         initialPopulationSize, populationSize, cutoff, [m_l, m_u], nIterations
output: total dominating set D^H
population ← ∅
for i = 1, ..., initialPopulationSize do
    newD ← GRASP(I, cutoff)
    if there is no solution with the same objective value and size as newD in population then
        population ← population ∪ newD
population ← select(population, populationSize)
for i = 1, ..., nIterations do
    for all pairs D^1, D^2 from population do
        newD ← crossover(D^1, D^2, cutoff)
        newD ← mutation(newD, [m_l, m_u])
        newD ← localSearch(newD)
        if there is no solution with the same objective value and size as newD in population then
            population ← population ∪ newD
    population ← select(population, populationSize)
D^H ← select(population, 1)
```

Algorithm 3: Genetic algorithm

5.1 Instance description

Instances similar to the instances of [22] In [22], the authors created instances to test their formulations. Unfortunately, the instances are not available online, thus we generated our own, following the same procedure as described in [22]. These instances are generated according to the Erdös-Rényi model, where one fixes the number of nodes, $|V|$ and a probability $p \in [01]$ that allows to control the edge density of the resulting graph. Edge and vertex weights, $\mathbf{c}$ and $\mathbf{w}$, respectively, are random integers between one and five. As in [22], we considered $|V| \in \{20, 50, 100\}$ and $p \in \{0.2, 0.5, 0.8\}$, which leads to instance ranging from 20 nodes and 31 edges to 100 nodes and 3943 edges. For each pair (n, p) we generated five instances (instead of one as done in [22]), using the `gnp_random_graph(n,p)`-method from the `networkx`-package [11] to obtain the Erdös-Rényi graphs. This set has $3 \cdot 3 \cdot 5 = 45$ instances. We denote this set of instances as `MA`, individual instances are addressed as `MA`$-|V| - p - id$, where $id \in \{1, \ldots, 5\}$.

New instances To analyze the influence of different weight structures, we generated an additional set of instances, denoted as `NEW`, as in the `MA` both are in a similar (small) range. We again used the Erdös-Rényi model, and considered $|V| \in \{75, 100, 125\}$ and $p \in \{0.2, 0.5, 0.8\}$. We used the following range-combinations for $(\mathbf{c}, \mathbf{w})$: $([1, 50], [1, 10])$, $([1, 25], [1, 25])$,$([1, 10], [1, 50])$. For each combination of $(|V|, p)$ and $(\mathbf{c}, \mathbf{w})$ we created five instances. Thus, this set has $3 \cdot 3 \cdot 3 \cdot 5 = 135$ instances. The instance set is denoted as `NEW`, individual instances are addressed as `NEW`$-|V| - p - c_u - id$, where $id \in \{1, \ldots, 5\}$ and c_u is the upper bound of the considered range for $\mathbf{c}$ (i.e., $c_u \in \{10, 25, 50\}$).

Both sets of instances we created are available online at `https://msinnl.github.io/pages/instancescodes.html`.

5.2 Assessing the effect of the valid inequalities

We now analyze the effect of the families of valid inequalities presented in Section 2. In order to test this, we added them individually to the corresponding model of the LP-relaxation of (F1) and (F2), and then also added all of them together. There is an exponential number of cliques; therefore, in order to get an impression of the effect of clique inequalities (CLIQUE), we heuristically calculate edge clique covers (i.e., set of cliques, such that every edge occurs in one of the cliques) using our separation heuristic described in Section 3.1 (using the vertex-degree as sorting criteria), and add the corresponding inequalities induced by the cliques in this cover.

In figure 2, we give a plot of the obtained LP gaps (over both instance sets), calculated as $100 \cdot (w^B - w^{LP})/w^B$, where w^B is the best solution value we obtained using our approaches, and w^{LP} is the value of the considered LP relaxation. Note that the figure does not give a plot for (F1)+(XZLINK2L),(F2),(F2)+(EXTCOSTS-i-L),(F2)+(TDOMY), (F2)+(CLIQUE). This is, because using the liftings (XZLINK2L), resp., (EXTCOSTS-i-L) on their own had no effect on the value of the LP relaxation. Moreover, the values obtained by (F2),(F2)+(TDOMY), (F2)+(CLIQUE) are just the same as their counterpart with (F1), as in (F2), the $\mathbf{z}$-variables are projected out in a Benders way, and (TDOMY) and (CLIQUE) operate in the $(\mathbf{x}, \mathbf{y})$-space. On the other hand, there is a slight difference between (F1)+all and (F2)+all, with (F1)+all giving slightly lower gaps. An explanation for this is, that once inequalities (TDOMY) and (CLIQUE) are present in the model, the liftings (XZLINK2L), resp., (EXTCOSTS-i-L) also start to have an effect on the bounds (as both (TDOMY) and (CLIQUE) "push" the $\mathbf{y}$-variables, which are the variables added in the lifting).

Overall, we see that both (TDOMY) and (CLIQUE) on their own result in a considerable improvement of the LP gap, e.g., without any inequalities only around 20% of the instances have an LP gap of 20% or less, while adding (TDOMY) or (CLIQUE) increases the number of instances with such a gap to about 30%. Adding all families together gives again a considerable improvement, now for about 50% of instances, the LP gap is 20% or less. When adding all inequalities, the largest LP gap is around 50%, while without adding any inequalities, the largest gap is over 70%.

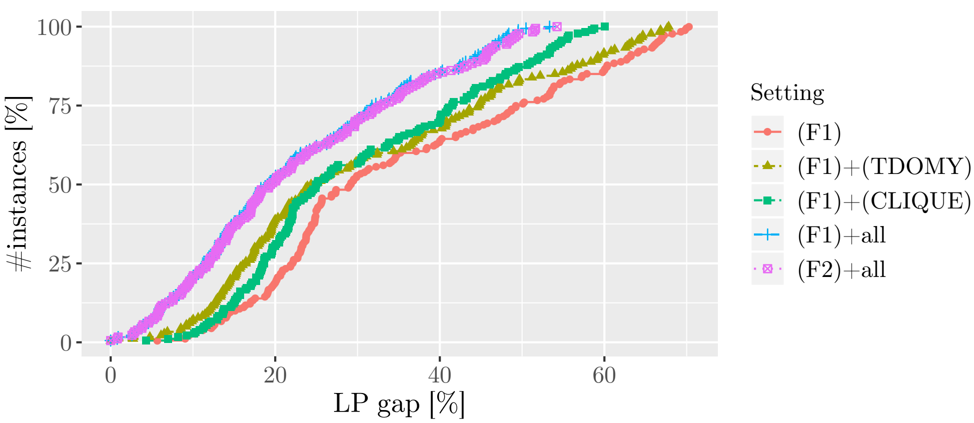


Figure 2: LP-gap plot for adding families of valid inequalities to (F1) and (F2).

5.3 Detailed results

In this section, we provide detailed results obtained by the branch-and-cut frameworks based on the formulations (F1) and (F2) described in Section 3, and also the genetic algorithm. A comparison with the models of [22] is also done.

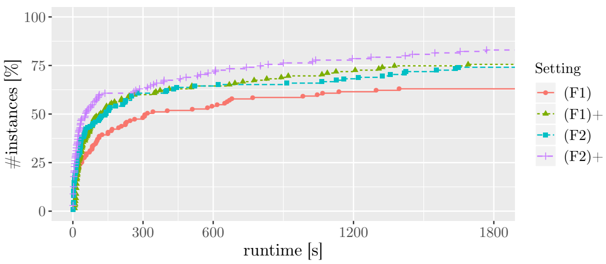
In Table 1 we report the following results for instance set `MA`. In columns (F1)+ and (F2)+, we report the results attained by our branch-and-cut algorithms. In columns (F1) and (F2) we report the results attained when solving (F1) and (F2) directly with CPLEX, without any of the branch-and-cut ingredients presented in Section 3 (note that in case of (F2), constraints EXTCOSTS-i are separated as they are needed to ensure correctness). In columns (MA1), (MA2), and (MA3) we report the results attained when solving directly by CPLEX the formulations provided in [22]. In this table, we report for each approach the runtime (column $t[s]$), the objective value of the best obtained solution (column w^B) and the optimality gap (column $g[\%]$, calculated as $100 \cdot (w^B - LB)/w^B$, where LB is the obtained lower bound).

The results reported in Table 1 show, that the approaches proposed in this paper (i.e., (F1), (F2), (F1)+ and (F2)+), are considerably more effective than those proposed in [22]: All of our approaches, with the exception of (F1), managed to solve all the instances within the timelimit, while none of the approaches (MA1), (MA2), and (MA3) managed to solve the instances with 100 nodes to optimality (and (MA1), (MA2) also fail for some of the instances with 50 nodes). Moreover, for the instances where our approaches as well as the approaches of [22] can solve the problem, our approaches are up to 500 times faster; see, e.g, instances `MA-50-0.5-4`, where (F2)+ takes 2 seconds, while (MA3) takes 1201 seconds (and (MA1), (MA2) reach the timelimit). Likewise,the gaps attained by our strategies (when the time limit is reached), are considerable smaller than those attained by the [22] approaches (for instance, while the maximum gap attained by (F1) is 9.06%, the maximum gap attained by (MA3) is 72.53%). When comparing among our approaches, we see that for all but two instances (`MA-100-0.8-3` and `MA-100-0.8-4`), (F2)+ is the fastest, and for these two instances (F2) is the fastest. Not surprisingly, the instances become harder with larger number of vertices, and, also higher density (p) seems to make the instances harder.

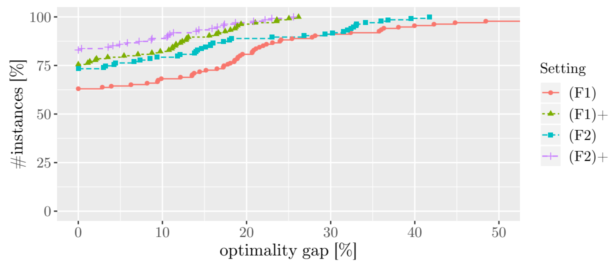
Table 1: Comparison with previous approaches (MA1), (MA2), (MA3) from literature on instance set `MA`

instance			(F1)			(F1)+			(F2)			(F2)+			(MA1)			(MA2)			(MA3)		
$\|V\|$	p	id	$t[s]$	w^*	g[%]	$t[s]$	w^*	g[%]	$t[s]$	w^*	g[%]	$t[s]$	w^*	g[%]	$t[s]$	w^*	g[%]	$t[s]$	w^*	g[%]	$t[s]$	w^*	g[%]
20	0.2	1	**1**	63	0.00	**1**	63	0.00	**1**	63	0.00	**1**	63	0.00	2	63	0.00	3	63	0.00	2	63	0.00
20	0.2	2	**1**	58	0.00	**1**	58	0.00	**1**	58	0.00	**1**	58	0.00	3	58	0.00	3	58	0.00	**1**	58	0.00
20	0.2	3	3	58	0.00	**1**	58	0.00	**1**	58	0.00	**1**	58	0.00	3	58	0.00	3	58	0.00	**1**	58	0.00
20	0.2	4	**1**	51	0.00	**1**	51	0.00	**1**	51	0.00	**1**	51	0.00	4	51	0.00	3	51	0.00	2	51	0.00
20	0.2	5	**1**	55	0.00	**1**	55	0.00	**1**	55	0.00	**1**	55	0.00	3	55	0.00	4	55	0.00	**1**	55	0.00
20	0.5	1	**1**	44	0.00	**1**	44	0.00	**1**	44	0.00	**1**	44	0.00	5	44	0.00	4	44	0.00	**1**	44	0.00
20	0.5	2	**1**	47	0.00	**1**	47	0.00	**1**	47	0.00	**1**	47	0.00	4	47	0.00	5	47	0.00	**1**	47	0.00
20	0.5	3	**1**	46	0.00	**1**	46	0.00	**1**	46	0.00	**1**	46	0.00	6	46	0.00	4	46	0.00	**1**	46	0.00
20	0.5	4	**1**	40	0.00	**1**	40	0.00	**1**	40	0.00	**1**	40	0.00	4	40	0.00	3	40	0.00	**1**	40	0.00
20	0.5	5	**1**	41	0.00	**1**	41	0.00	**1**	41	0.00	**1**	41	0.00	4	41	0.00	3	41	0.00	**1**	41	0.00
20	0.8	1	**1**	37	0.00	**1**	37	0.00	**1**	37	0.00	**1**	37	0.00	3	37	0.00	4	37	0.00	2	37	0.00
20	0.8	2	3	35	0.00	**1**	35	0.00	**1**	35	0.00	**1**	35	0.00	4	35	0.00	4	35	0.00	**1**	35	0.00
20	0.8	3	**1**	40	0.00	**1**	40	0.00	**1**	40	0.00	**1**	40	0.00	5	40	0.00	4	40	0.00	**1**	40	0.00
20	0.8	4	**1**	34	0.00	**1**	34	0.00	**1**	34	0.00	**1**	34	0.00	4	34	0.00	5	34	0.00	**1**	34	0.00
20	0.8	5	**1**	34	0.00	**1**	34	0.00	**1**	34	0.00	**1**	34	0.00	5	34	0.00	3	34	0.00	**1**	34	0.00
50	0.2	1	2	111	0.00	**1**	111	0.00	4	111	0.00	**1**	111	0.00	991	111	0.00	216	111	0.00	229	111	0.00
50	0.2	2	3	106	0.00	**1**	106	0.00	3	106	0.00	**1**	106	0.00	1380	106	0.00	438	106	0.00	309	106	0.00
50	0.2	3	8	111	0.00	**1**	111	0.00	4	111	0.00	**1**	111	0.00	TL	114	10.40	755	111	0.00	552	111	0.00
50	0.2	4	4	101	0.00	**1**	101	0.00	4	101	0.00	**1**	101	0.00	796	101	0.00	322	101	0.00	409	101	0.00
50	0.2	5	13	108	0.00	**1**	108	0.00	6	108	0.00	**1**	108	0.00	TL	108	12.19	1565	108	0.00	1129	108	0.00
50	0.5	1	4	82	0.00	3	82	0.00	3	82	0.00	**2**	82	0.00	TL	82	19.75	TL	82	9.11	631	82	0.00
50	0.5	2	5	85	0.00	**2**	85	0.00	3	85	0.00	**2**	85	0.00	TL	88	19.39	1579	85	0.00	794	85	0.00
50	0.5	3	22	84	0.00	3	84	0.00	4	84	0.00	**2**	84	0.00	TL	87	18.31	TL	85	9.37	1082	84	0.00
50	0.5	4	16	82	0.00	3	82	0.00	4	82	0.00	**2**	82	0.00	TL	82	17.55	TL	82	14.90	1201	82	0.00
50	0.5	5	15	82	0.00	4	82	0.00	4	82	0.00	**2**	82	0.00	TL	83	20.30	TL	82	12.69	1062	82	0.00
50	0.8	1	6	77	0.00	7	77	0.00	5	77	0.00	**4**	77	0.00	TL	77	8.39	1063	77	0.00	876	77	0.00
50	0.8	2	3	72	0.00	3	72	0.00	3	72	0.00	**2**	72	0.00	452	72	0.00	307	72	0.00	299	72	0.00
50	0.8	3	3	74	0.00	3	74	0.00	4	74	0.00	**2**	74	0.00	1100	74	0.00	587	74	0.00	446	74	0.00
50	0.8	4	6	76	0.00	5	76	0.00	4	76	0.00	**3**	76	0.00	TL	76	10.23	1242	76	0.00	736	76	0.00
50	0.8	5	13	79	0.00	15	79	0.00	**7**	79	0.00	**7**	79	0.00	TL	79	16.13	1720	79	0.00	1310	79	0.00
100	0.2	1	898	175	0.00	92	175	0.00	566	175	0.00	**50**	175	0.00	TL	183	38.79	TL	178	34.82	TL	175	56.48
100	0.2	2	251	174	0.00	14	174	0.00	276	174	0.00	**12**	174	0.00	TL	174	34.17	TL	175	34.60	TL	188	61.21
100	0.2	3	TL	178	6.48	239	177	0.00	1434	177	0.00	**121**	177	0.00	TL	195	43.27	TL	189	41.47	TL	183	60.95
100	0.2	4	TL	169	2.17	81	169	0.00	562	169	0.00	**37**	169	0.00	TL	172	37.79	TL	175	36.77	TL	172	59.06
100	0.2	5	TL	171	5.39	97	167	0.00	1473	167	0.00	**47**	167	0.00	TL	170	39.18	TL	172	38.20	TL	173	60.06
100	0.5	1	TL	147	2.69	304	147	0.00	292	147	0.00	**108**	147	0.00	TL	166	51.92	TL	160	49.55	TL	149	62.37
100	0.5	2	707	144	0.00	158	144	0.00	152	144	0.00	**51**	144	0.00	TL	154	44.81	TL	148	42.81	TL	150	66.31
100	0.5	3	995	147	0.00	401	147	0.00	186	147	0.00	**128**	147	0.00	TL	157	48.58	TL	149	43.22	TL	160	62.88
100	0.5	4	TL	149	9.06	725	146	0.00	289	146	0.00	**214**	146	0.00	TL	156	50.99	TL	150	46.05	TL	160	63.98
100	0.5	5	TL	139	3.18	466	139	0.00	242	139	0.00	**155**	139	0.00	TL	148	49.27	TL	145	44.48	TL	152	71.38
100	0.8	1	1655	136	0.00	346	136	0.00	172	136	0.00	**97**	136	0.00	TL	150	55.44	TL	141	51.52	TL	136	62.64
100	0.8	2	759	140	0.00	894	140	0.00	283	140	0.00	**249**	140	0.00	TL	146	49.55	TL	141	44.35	TL	147	65.31
100	0.8	3	1212	141	0.00	1032	141	0.00	**236**	141	0.00	325	141	0.00	TL	144	53.13	TL	149	50.87	TL	153	71.51
100	0.8	4	TL	141	7.45	1652	141	0.00	**334**	141	0.00	495	141	0.00	TL	148	52.31	TL	147	50.83	TL	142	71.17
100	0.8	5	990	134	0.00	509	134	0.00	231	134	0.00	**160**	134	0.00	TL	152	55.60	TL	148	51.19	TL	156	72.53

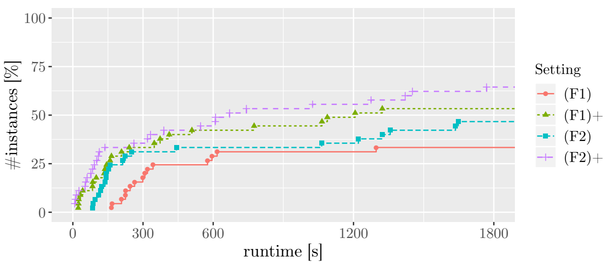
In the following, we focus on the instance set `NEW` and our solution algorithms to get more insights on the performance of them, in particular, their behavior with respect to different weight structures. In Figure 3, we present plots of runtimes to optimality, and optimality gaps (for the unsolved instances of the respective approaches) for (F1) (F2), (F1)+ and (F2)+. Figures 3a and 3b give runtimes optimality gaps, respectively, for the complete set of instances `NEW`, while Figures 3c-3g show results for the different weight structure, i.e., Figures 3c and 3d are for the instances with $c_u = 10$, Figures 3e and 3f are for the instances with $c_u = 25$ and Figure 3g is for the instances with $c_u = 50$ (since all instances are solved to optimality we do not provide an optimality gap plot). Note the different scales on the x-axis of Figure 3g compared to the other runtime-plots.



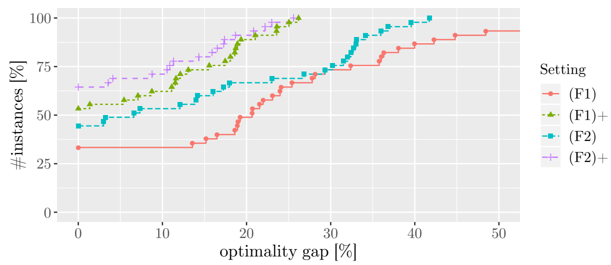
(a) Runtime plot for all instances of the set



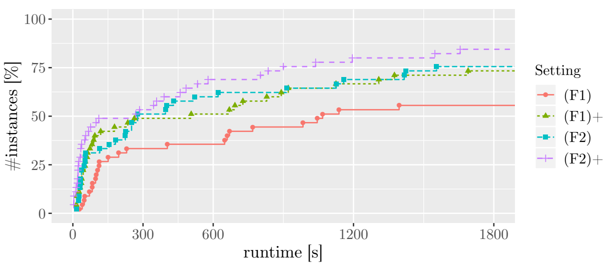
(b) Optimality gap plot for all instances of the set



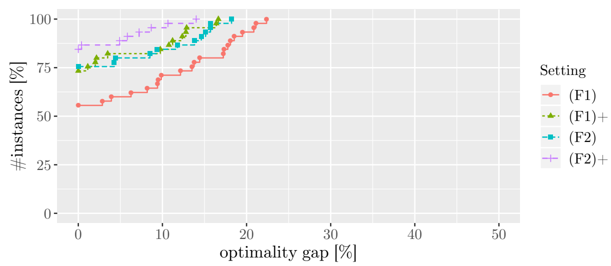
(c) Runtime plot for instances with $c_u = 10$



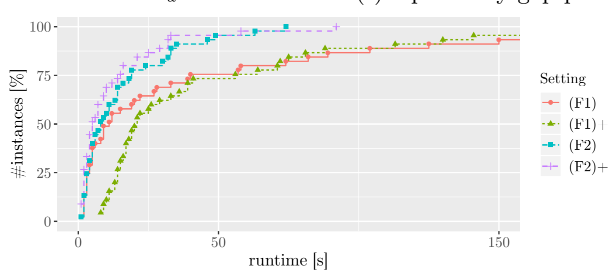
(d) Optimality gap plot for instances with $c_u = 10$



(e) Runtime plot for instances with $c_u = 25$



(f) Optimality gap plot for instances with $c_u = 25$



(g) Runtime plot for instances with $c_u = 50$ (all solved to optimality with all approaches, but x-axis cut off at 150 seconds for better readability.)

Figure 3: Plots of runtimes and optimality gap of our MIP-approaches on instance set `NEW` and different subgroups of these instances

In the plots shown in Figure 3 we see a strong connection between the weight structure and the computational difficulty of the instances. All instances with $c_u = 50$ can be solved by all approaches within the timelimit; furthermore, (F2) and (F2)+ only need at most 100 seconds. However, for both $c_u = 25$ and $c_u = 10$, the situation is strikingly different, in particular, for $c_u = 10$; we can observe that only (F1)+ and (F2)+ manage to solve over 50% of the instances within the timelimit. This behavior might be explained by the fact that, for these instances, edges play a more important role (due to their larger weight range), and thus the problem becomes more similar to a BQP problems and, as it has been shown previously in the literature, problems where a BQP structure plays an important role a often very hard to solve (see, e.g., [1]). Such hypothesis could be validated by the fact that for these instances, the (F1)+ approach, which includes the valid inequalities (in particular BQP-like inequalities CLIQUE) is the approach working second best (not only when looking at the runtime, but also when looking at the optimality gap for the unsolved instances). Moreover, for instances with $c_u = 50$ (where edge weights are less influential), the second best approach is (F2), which does not contain any valid inequalities. Despite these differences, when considering at all instances, we see that (F2)+ works best, managing to solve around 80% of the instances within the timelimit, followed by (F2) and (F1)+, which both manage to solve around 75% of the instances.

The results reported in Figure 3 are complemented by Tables 2-4, where we give detailed results of our approaches, including the genetic algorithm (indicated by GA). Moreover, we also report the results obtained when running only the GRAP-part of the GA (i.e., lines 1-5 in Algorithm 3). There is one table for each value of $|V|$ in order to allow an analysis from another point of view. In these tables, we present the runtime (column $t[s]$; TL indicates timelimit reached, and ML memorylimit), and the objective value of the best obtained solution (column w^B); for the MIP-approaches also the optimality gap (column $g[\%]$), and the number of branch-and-cut nodes (column $\#nBN$); and for the GA and GRASP also the primal gap compared to the best solution found by the MIP-approaches (column $pg[\%]$, calculated as $100 \cdot (w^H - w^{MIP})/w^{MIP}$, where w^{MIP} is the value of the best solution found by the MIP-approaches and w^H the value of the best solution found by the GRASP, resp., GA).

In the tables, we can see that all our approaches manage to solve all instances with $|V| = 75$ to optimality. For instances with $|V| = 100$, (F2)+ solves all but two instances, and for instances with $|V| = 125$, (F2)+ solves 24 out of 45 instances to optimality within the timelimit. In general, for nearly all instances (F2)+ works best, i.e., either it has the smallest runtime, or, for unsolved instances, it has the smallest optimality gap. For the instances, where (F2)+ is not the best performing approach, (F2) gives the best results. With respect to this, we can see that (F2) only performs better for instances with $p = 0.8$ (i.e., denser instances). A possible explanation for this could be, that for denser instances, the LPs with added valid inequalities becomes denser, in particular the clique inequalities, as there will also be a lot more cliques for denser graphs. Thus, while adding the valid inequalities improves the bound, the drawback of longer LP solution times and thus the slower node-throughput in the branch-and-cut becomes burdensome. This is also reflected in the number of branch-and-cut nodes enumerated, (F2) often enumerates around ten times as much nodes as (F2)+, while the runtime of both approaches is quite similar (and over all instances, adding the valid inequalities pays off, as only for the dense instances with $p = 0.8$ the described drawback is having an effect). With respect to (F1) and (F1)+, the situation is similar, i.e., (F1) has a considerably higher node-throughput, but in general (F1)+ performs better. Moreover, when comparing (F1)+ and (F2)+, it can be seen that (F1)+ usually needs less branch-and-cut nodes to prove optimality (when it manages to do so), but is slower than (F2)+, as the "slimmer" formulation of (F2)+ allows for a faster node-throughput (while still being "strong enough" for proving optimality). The largest optimality gap is 25.59% and is obtained for instance `125-0.8-10-2`.

From the results reported in the tables, we can also conclude that both heuristics perform quite well. The GRASP takes at most nine seconds (for some of the instances with $|V| = 125$), and the largest primal gap around 20.21% (instance `NEW-125-0.2-10-3`), while most of the primal gaps are smaller than 10% and for slightly less than half of the instances, it is zero. The largest primal gaps are obtained for instances with $p = 0.2$. Likewise, the GA takes at most 86 seconds (for instance `NEW-125-0.8-50-5`) and only for 30 out of 135 instances, there is a positive primal gap (the largest is 5.26% for instance `NEW-75-0.2-10-4`, and for most instances with positive primal gap, the gap is under 1%). Interestingly, for none of the unsolved instances, the GA could find an improved solution compared to the best solution found by the MIP approaches.

Table 2: Comparison of our approaches on instance set `NEW` with $|V|$=75.

instance			(F1)				(F1)+				(F2)				(F2)+				GRASP			GA		
p	c_u	id	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	pg[%]	$t[s]$	w^B	pg[%]
0.2	10	1	576	686	0.00	105831	32	686	0.00	1832	251	686	0.00	123502	**15**	686	0.00	1921	1	769	12.10	5	686	0.00
0.2	10	2	617	770	0.00	135320	25	770	0.00	1167	224	770	0.00	121349	**8**	770	0.00	1147	1	871	13.12	6	794	3.12
0.2	10	3	319	661	0.00	48274	23	661	0.00	665	85	661	0.00	46205	**8**	661	0.00	796	1	765	15.73	6	661	0.00
0.2	10	4	595	703	0.00	105611	42	703	0.00	2456	443	703	0.00	216827	**26**	703	0.00	2938	1	762	8.39	7	740	5.26
0.2	10	5	168	758	0.00	23620	24	758	0.00	1092	160	758	0.00	82778	**11**	758	0.00	1340	1	857	13.06	6	779	2.77
0.2	25	1	51	498	0.00	6617	16	498	0.00	263	37	498	0.00	14486	**3**	498	0.00	285	1	556	11.65	6	504	1.20
0.2	25	2	49	546	0.00	8616	16	546	0.00	170	37	546	0.00	18598	**3**	546	0.00	179	1	607	11.17	6	546	0.00
0.2	25	3	40	518	0.00	6505	15	518	0.00	157	27	518	0.00	10784	**4**	518	0.00	224	1	603	16.41	5	518	0.00
0.2	25	4	649	498	0.00	115335	36	498	0.00	3276	226	498	0.00	116161	**19**	498	0.00	3772	1	521	4.62	6	498	0.00
0.2	25	5	97	513	0.00	15696	15	513	0.00	245	54	513	0.00	23682	**4**	513	0.00	288	1	526	2.53	6	513	0.00
0.2	50	1	2	339	0.00	124	10	339	0.00	21	2	339	0.00	162	**1**	339	0.00	28	1	340	0.29	6	339	0.00
0.2	50	2	2	382	0.00	133	13	382	0.00	43	2	382	0.00	237	**1**	382	0.00	40	1	414	8.38	5	382	0.00
0.2	50	3	**1**	335	0.00	64	14	335	0.00	6	**1**	335	0.00	54	**1**	335	0.00	10	1	341	1.79	5	341	1.79
0.2	50	4	3	333	0.00	317	14	333	0.00	59	3	333	0.00	400	**2**	333	0.00	59	1	338	1.50	6	333	0.00
0.2	50	5	3	347	0.00	374	15	347	0.00	82	3	347	0.00	423	**2**	347	0.00	94	1	353	1.73	6	347	0.00
0.5	10	1	1297	581	0.00	99392	159	581	0.00	4820	213	581	0.00	62202	**109**	581	0.00	8222	1	590	1.55	13	581	0.00
0.5	10	2	299	602	0.00	30769	134	602	0.00	3840	152	602	0.00	41014	**84**	602	0.00	6719	1	641	6.48	11	602	0.00
0.5	10	3	226	545	0.00	19174	100	545	0.00	2739	141	545	0.00	35146	**61**	545	0.00	4960	1	545	0.00	10	545	0.00
0.5	10	4	264	540	0.00	25262	84	540	0.00	1960	109	540	0.00	28090	**55**	540	0.00	3797	1	580	7.41	10	540	0.00
0.5	10	5	165	519	0.00	13004	85	519	0.00	1803	119	519	0.00	28200	**52**	519	0.00	3291	1	551	6.17	10	519	0.00
0.5	25	1	106	387	0.00	7161	52	387	0.00	1269	31	387	0.00	7626	**23**	387	0.00	1598	1	402	3.88	10	387	0.00
0.5	25	2	71	384	0.00	5083	44	384	0.00	1194	24	384	0.00	5674	**20**	384	0.00	1458	1	413	7.55	10	384	0.00
0.5	25	3	83	362	0.00	4769	26	362	0.00	343	31	362	0.00	5898	**13**	362	0.00	442	1	380	4.97	10	362	0.00
0.5	25	4	100	366	0.00	7073	63	366	0.00	1833	52	366	0.00	14482	**31**	366	0.00	2551	1	371	1.37	9	371	1.37
0.5	25	5	84	331	0.00	4938	31	331	0.00	389	24	331	0.00	4688	**14**	331	0.00	470	1	331	0.00	10	331	0.00
0.5	50	1	5	240	0.00	348	16	240	0.00	77	4	240	0.00	424	**3**	240	0.00	97	1	244	1.67	9	240	0.00
0.5	50	2	**2**	238	0.00	43	11	238	0.00	28	**2**	238	0.00	68	**2**	238	0.00	30	1	245	2.94	9	238	0.00
0.5	50	3	2	215	0.00	0	8	215	0.00	0	2	215	0.00	47	**1**	215	0.00	0	1	215	0.00	9	215	0.00
0.5	50	4	8	235	0.00	553	14	235	0.00	141	5	235	0.00	703	**4**	235	0.00	134	1	235	0.00	9	235	0.00
0.5	50	5	4	206	0.00	198	11	206	0.00	47	3	206	0.00	198	**2**	206	0.00	47	1	206	0.00	8	206	0.00
0.8	10	1	343	571	0.00	24196	241	571	0.00	3009	**137**	571	0.00	23527	**137**	571	0.00	4798	2	613	7.36	16	571	0.00
0.8	10	2	208	520	0.00	12433	144	520	0.00	1481	**86**	520	0.00	14721	102	520	0.00	2583	2	520	0.00	15	520	0.00
0.8	10	3	245	543	0.00	13720	165	543	0.00	2105	122	543	0.00	19460	**92**	543	0.00	3475	2	543	0.00	15	543	0.00
0.8	10	4	308	571	0.00	17925	208	571	0.00	2722	142	571	0.00	27914	**113**	571	0.00	4340	2	571	0.00	15	571	0.00
0.8	10	5	225	509	0.00	10311	137	509	0.00	1374	94	509	0.00	15107	**77**	509	0.00	2412	2	509	0.00	17	509	0.00
0.8	25	1	196	357	0.00	6516	119	357	0.00	1527	53	357	0.00	7532	**51**	357	0.00	1957	2	360	0.84	15	357	0.00
0.8	25	2	151	338	0.00	4736	89	338	0.00	1119	**34**	338	0.00	4454	**34**	338	0.00	1201	2	356	5.33	15	338	0.00
0.8	25	3	28	323	0.00	1495	44	323	0.00	439	**15**	323	0.00	1568	22	323	0.00	549	2	323	0.00	13	323	0.00
0.8	25	4	113	345	0.00	4240	73	345	0.00	670	47	345	0.00	6870	**37**	345	0.00	947	2	345	0.00	13	345	0.00
0.8	25	5	112	311	0.00	3977	53	311	0.00	570	32	311	0.00	3900	**25**	311	0.00	747	2	311	0.00	15	311	0.00
0.8	50	1	**2**	182	0.00	29	8	182	0.00	13	4	182	0.00	136	**2**	182	0.00	16	2	182	0.00	14	182	0.00
0.8	50	2	3	188	0.00	61	9	188	0.00	33	3	188	0.00	86	**2**	188	0.00	30	2	188	0.00	11	188	0.00
0.8	50	3	3	191	0.00	64	9	191	0.00	16	**2**	191	0.00	35	**2**	191	0.00	19	2	191	0.00	11	191	0.00
0.8	50	4	5	196	0.00	127	13	196	0.00	67	**4**	196	0.00	222	**4**	196	0.00	62	2	196	0.00	12	196	0.00
0.8	50	5	**5**	192	0.00	176	15	192	0.00	81	6	192	0.00	402	7	192	0.00	77	2	192	0.00	15	192	0.00

Table 3: Comparison of our approaches on instance set `NEW` with $|V|$=100.

instance			(F1)				(F1)+				(F2)				(F2)+				GRASP			GA		
p	c_u	id	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	pg[%]	$t[s]$	w^B	pg[%]
0.2	10	1	TL	931	23.09	150968	373	873	0.00	17695	TL	914	16.03	488396	**319**	873	0.00	28266	1	930	6.53	12	873	0.00
0.2	10	2	TL	991	20.67	195361	348	944	0.00	16097	TL	966	13.96	537600	**261**	944	0.00	21216	1	983	4.13	13	944	0.00
0.2	10	3	TL	933	21.97	175869	509	878	0.00	24488	TL	937	17.98	488300	**389**	878	0.00	32623	1	905	3.08	11	878	0.00
0.2	10	4	TL	837	19.00	160900	775	837	0.00	36727	TL	850	14.18	533800	**546**	837	0.00	44398	1	879	5.02	11	837	0.00
0.2	10	5	TL	913	23.99	166974	412	840	0.00	19910	TL	847	12.08	455300	**332**	840	0.00	31130	1	907	7.98	12	870	3.57
0.2	25	1	660	591	0.00	52158	82	591	0.00	3688	401	591	0.00	97432	**55**	591	0.00	5421	1	591	0.00	12	591	0.00
0.2	25	2	TL	655	3.93	154809	94	653	0.00	4420	1126	653	0.00	279737	**67**	653	0.00	7342	1	687	5.21	11	655	0.31
0.2	25	3	769	612	0.00	64664	61	612	0.00	2355	251	612	0.00	56158	**34**	612	0.00	3142	1	648	5.88	12	616	0.65
0.2	25	4	TL	558	2.87	122658	38	552	0.00	917	224	552	0.00	53608	**22**	552	0.00	1716	1	602	9.06	11	552	0.00
0.2	25	5	TL	606	6.27	172508	506	606	0.00	31561	TL	609	4.40	401219	**345**	606	0.00	40428	1	646	6.60	12	607	0.17
0.2	50	1	11	418	0.00	929	19	418	0.00	193	8	418	0.00	1247	**5**	418	0.00	249	1	422	0.96	12	420	0.48
0.2	50	2	9	447	0.00	774	18	447	0.00	177	8	447	0.00	1154	**7**	447	0.00	261	1	472	5.59	11	456	2.01
0.2	50	3	5	419	0.00	339	17	419	0.00	106	6	419	0.00	590	**4**	419	0.00	124	1	427	1.91	11	419	0.00
0.2	50	4	74	403	0.00	4679	21	403	0.00	329	19	403	0.00	3709	**10**	403	0.00	395	1	418	3.72	12	410	1.74
0.2	50	5	12	375	0.00	800	20	375	0.00	290	10	375	0.00	1663	**5**	375	0.00	328	1	379	1.07	13	379	1.07
0.5	10	1	TL	763	18.61	104422	TL	743	7.09	26400	TL	743	7.32	240277	**1421**	743	0.00	61206	2	749	0.81	26	749	0.81
0.5	10	2	TL	708	28.16	112961	1207	698	0.00	15696	1357	698	0.00	226826	**670**	698	0.00	25318	3	705	1.00	25	700	0.29
0.5	10	3	TL	728	16.48	103135	1323	699	0.00	19211	TL	699	3.00	291119	**742**	699	0.00	29592	3	730	4.43	24	718	2.72
0.5	10	4	TL	726	13.57	107121	1088	726	0.00	13761	1324	726	0.00	218790	**609**	726	0.00	22324	2	775	6.75	26	726	0.00
0.5	10	5	TL	761	24.11	124466	TL	702	1.37	24691	TL	744	17.25	240100	**1275**	702	0.00	51404	2	743	5.84	25	702	0.00
0.5	25	1	670	461	0.00	18182	235	461	0.00	2640	182	461	0.00	22792	**99**	461	0.00	3913	3	461	0.00	25	461	0.00
0.5	25	2	230	437	0.00	6776	178	437	0.00	2454	115	437	0.00	12140	**76**	437	0.00	3372	2	448	2.52	19	437	0.00
0.5	25	3	404	434	0.00	10685	263	434	0.00	3821	155	434	0.00	16969	**111**	434	0.00	4425	3	443	2.07	22	434	0.00
0.5	25	4	TL	494	8.20	63896	921	482	0.00	16949	621	482	0.00	90411	**533**	482	0.00	22037	2	489	1.45	25	482	0.00
0.5	25	5	1395	456	0.00	40173	829	456	0.00	12615	430	456	0.00	59506	**358**	456	0.00	16885	3	470	3.07	23	457	0.22
0.5	50	1	4	260	0.00	27	17	260	0.00	5	5	260	0.00	131	**3**	260	0.00	5	2	260	0.00	22	260	0.00
0.5	50	2	3	271	0.00	27	17	271	0.00	15	3	271	0.00	55	**2**	271	0.00	14	2	271	0.00	21	271	0.00
0.5	50	3	9	283	0.00	282	21	283	0.00	119	7	283	0.00	404	**4**	283	0.00	135	3	283	0.00	21	283	0.00
0.5	50	4	27	291	0.00	914	39	291	0.00	353	11	291	0.00	1070	**10**	291	0.00	355	2	296	1.72	22	291	0.00
0.5	50	5	12	269	0.00	347	29	269	0.00	228	14	269	0.00	1254	**9**	269	0.00	251	2	269	0.00	21	269	0.00
0.8	10	1	TL	730	20.71	55600	TL	730	15.57	11878	TL	730	**3.22**	176962	TL	730	8.77	30496	4	730	0.00	39	730	0.00
0.8	10	2	TL	697	15.18	40114	TL	683	11.61	5773	1064	683	0.00	103180	**1025**	683	0.00	17483	4	688	0.73	37	683	0.00
0.8	10	3	TL	721	19.24	47870	TL	718	11.53	10506	**1636**	718	0.00	154346	1769	718	0.00	31269	4	718	0.00	37	718	0.00
0.8	10	4	TL	726	36.02	52165	TL	709	8.75	9566	TL	712	6.60	153824	**1452**	709	0.00	28487	4	709	0.00	41	709	0.00
0.8	10	5	TL	703	18.92	43898	TL	700	18.08	7205	**1221**	700	0.00	118429	TL	700	3.55	28770	4	710	1.43	39	704	0.57
0.8	25	1	1138	442	0.00	15789	1125	442	0.00	7017	**396**	442	0.00	34791	459	442	0.00	9633	5	452	2.26	40	442	0.00
0.8	25	2	1068	430	0.00	15155	693	430	0.00	4218	**277**	430	0.00	27907	285	430	0.00	5284	4	430	0.00	32	430	0.00
0.8	25	3	984	426	0.00	15999	669	426	0.00	4180	**251**	426	0.00	19709	269	426	0.00	5152	4	426	0.00	36	426	0.00
0.8	25	4	1045	428	0.00	17287	891	428	0.00	5285	**277**	428	0.00	27607	390	428	0.00	7049	4	428	0.00	35	428	0.00
0.8	25	5	TL	447	9.51	26364	1375	432	0.00	8047	**520**	432	0.00	51363	578	432	0.00	10357	4	432	0.00	42	432	0.00
0.8	50	1	33	259	0.00	993	75	259	0.00	396	**16**	259	0.00	1415	21	259	0.00	427	4	259	0.00	32	259	0.00
0.8	50	2	6	246	0.00	41	25	246	0.00	33	**5**	246	0.00	96	6	246	0.00	44	4	246	0.00	9	246	0.00
0.8	50	3	9	238	0.00	106	26	238	0.00	31	**5**	238	0.00	154	**5**	238	0.00	39	4	238	0.00	34	238	0.00
0.8	50	4	28	253	0.00	673	56	253	0.00	210	**14**	253	0.00	757	16	253	0.00	232	4	258	1.98	34	253	0.00
0.8	50	5	39	248	0.00	1042	81	248	0.00	414	**18**	248	0.00	1428	25	248	0.00	451	5	250	0.81	31	248	0.00

Table 4: Comparison of our approaches on instance set NEW with $|V|$=125.

instance			(F1)				(F1)+				(F2)				(F2)+				GRASP			GA		
p	c_u	id	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	g[%]	$\#nN$	$t[s]$	w^B	pg[%]	$t[s]$	w^B	pg[%]
0.2	10	1	TL	1031	29.33	111233	TL	1031	13.08	38900	TL	1122	33.07	289800	TL	1026	**11.31**	66500	2	1112	8.38	24	1026	0.00
0.2	10	2	TL	1038	25.41	103199	TL	1038	5.43	39364	TL	1136	29.30	323400	TL	1038	**4.11**	70700	2	1069	2.99	22	1038	0.00
0.2	10	3	TL	935	21.55	112721	1065	935	0.00	18545	TL	1006	23.01	307900	**610**	935	0.00	23794	2	1124	20.21	23	947	1.28
0.2	10	4	TL	1087	32.38	162826	TL	1050	11.10	35800	TL	1102	30.22	307500	TL	1052	**10.55**	65400	2	1121	6.76	21	1051	0.10
0.2	10	5	TL	1067	38.06	98022	TL	978	12.12	46300	TL	1069	32.41	293644	TL	974	**10.88**	72100	2	1112	14.17	25	975	0.10
0.2	25	1	TL	752	14.43	79324	727	720	0.00	20101	TL	777	15.13	249000	**484**	720	0.00	30681	2	803	11.53	26	720	0.00
0.2	25	2	TL	748	9.42	115536	1690	746	0.00	49679	TL	755	9.34	250400	**1038**	746	0.00	66326	2	768	2.95	24	748	0.27
0.2	25	3	TL	758	17.24	76593	1308	715	0.00	32387	TL	756	13.82	262200	**802**	715	0.00	58391	2	752	5.17	21	717	0.28
0.2	25	4	TL	725	13.52	125557	TL	701	1.13	45548	TL	726	11.79	277800	**1195**	701	0.00	68666	2	726	3.57	22	705	0.57
0.2	25	5	TL	690	12.15	125451	TL	684	3.49	48278	TL	714	14.62	284000	**1548**	684	0.00	94996	2	747	9.21	23	697	1.90
0.2	50	1	22	455	0.00	914	19	455	0.00	94	14	455	0.00	1809	**3**	455	0.00	112	2	457	0.44	21	455	0.00
0.2	50	2	15	477	0.00	552	22	477	0.00	163	11	477	0.00	1216	**4**	477	0.00	153	2	493	3.35	23	477	0.00
0.2	50	3	150	490	0.00	5438	33	490	0.00	379	32	490	0.00	4963	**9**	490	0.00	446	2	501	2.24	21	490	0.00
0.2	50	4	307	467	0.00	10476	36	467	0.00	678	63	467	0.00	11763	**14**	467	0.00	903	2	504	7.92	23	467	0.00
0.2	50	5	680	457	0.00	27859	71	457	0.00	1974	74	457	0.00	12890	**29**	457	0.00	2719	2	468	2.41	24	459	0.44
0.5	10	1	TL	888	35.77	74241	TL	817	19.35	11969	TL	920	32.99	189400	TL	817	**15.90**	32310	4	817	0.00	41	817	0.00
0.5	10	2	TL	838	27.80	71722	TL	815	18.60	11600	TL	902	35.97	165500	TL	815	**14.33**	28242	5	827	1.47	45	815	0.00
0.5	10	3	TL	931	48.44	71111	TL	836	21.04	12000	TL	915	32.01	183200	TL	836	**18.68**	31000	4	880	5.26	45	872	4.31
0.5	10	4	TL	912	36.32	81756	TL	867	23.60	12100	ML	947	34.16	194451	TL	867	**20.84**	28328	4	914	5.42	55	867	0.00
0.5	10	5	TL	949	39.97	76935	TL	867	25.04	12998	TL	995	41.73	188520	TL	867	**22.20**	30407	5	906	4.50	55	867	0.00
0.5	25	1	TL	613	21.13	37273	TL	566	9.75	13891	TL	566	**4.27**	160389	TL	566	4.91	37868	5	566	0.00	48	566	0.00
0.5	25	2	TL	542	9.87	30162	TL	533	2.02	14217	915	533	0.00	78306	**900**	533	0.00	24650	5	561	5.25	48	533	0.00
0.5	25	3	TL	563	13.77	30148	TL	538	2.16	12186	1417	538	0.00	111362	**835**	538	0.00	19259	5	567	5.39	49	538	0.00
0.5	25	4	TL	567	18.54	37033	TL	552	16.40	10610	TL	576	15.71	149600	TL	552	**10.66**	37400	4	565	2.36	53	552	0.00
0.5	25	5	TL	572	19.52	38695	TL	545	12.36	15193	TL	552	**8.51**	148100	TL	545	8.67	40091	5	548	0.55	48	548	0.55
0.5	50	1	40	334	0.00	785	64	334	0.00	473	19	334	0.00	1495	**16**	334	0.00	481	4	336	0.60	40	334	0.00
0.5	50	2	19	330	0.00	500	41	330	0.00	251	13	330	0.00	631	**12**	330	0.00	255	4	330	0.00	38	330	0.00
0.5	50	3	20	315	0.00	247	39	315	0.00	80	9	315	0.00	219	**7**	315	0.00	77	5	315	0.00	49	315	0.00
0.5	50	4	57	316	0.00	834	88	316	0.00	488	33	316	0.00	2657	**21**	316	0.00	504	5	316	0.00	51	316	0.00
0.5	50	5	104	311	0.00	2479	113	311	0.00	1099	33	311	0.00	3111	**32**	311	0.00	1107	4	311	0.00	40	311	0.00
0.8	10	1	TL	855	53.04	33869	TL	793	18.91	4892	TL	855	32.69	123500	TL	793	**17.38**	15086	9	793	0.00	78	793	0.00
0.8	10	2	TL	913	44.80	35253	TL	853	26.18	6445	TL	899	36.83	117000	TL	845	**25.59**	17975	8	854	1.07	72	845	0.00
0.8	10	3	TL	885	42.29	34230	TL	787	18.71	4916	TL	841	26.83	107600	TL	787	**17.29**	16300	9	829	5.34	74	787	0.00
0.8	10	4	TL	853	55.10	34257	TL	777	17.46	4700	TL	830	31.51	109100	TL	777	**16.52**	15100	9	829	6.69	83	777	0.00
0.8	10	5	TL	865	58.98	34289	TL	820	23.57	5188	TL	904	39.57	114502	TL	813	**23.00**	16200	8	827	1.72	77	813	0.00
0.8	25	1	TL	514	18.09	18822	TL	508	12.79	6100	**1555**	508	0.00	100191	TL	508	7.23	16220	9	521	2.56	69	510	0.39
0.8	25	2	TL	504	17.32	13406	TL	498	10.76	6000	**1158**	498	0.00	75456	1656	498	0.00	16200	9	499	0.20	65	498	0.00
0.8	25	3	TL	533	20.87	17604	TL	513	12.87	5558	TL	550	15.73	107800	TL	513	**5.83**	16743	9	523	1.95	77	513	0.00
0.8	25	4	TL	505	17.77	19977	TL	493	11.21	5241	**1424**	493	0.00	92315	TL	493	0.39	17238	8	506	2.64	75	493	0.00
0.8	25	5	TL	515	22.38	16373	TL	504	16.65	7000	TL	528	18.21	114500	TL	504	**14.02**	19469	8	519	2.98	76	504	0.00
0.8	50	1	511	307	0.00	4416	355	307	0.00	1499	**49**	307	0.00	2868	92	307	0.00	1568	8	307	0.00	64	307	0.00
0.8	50	2	58	296	0.00	897	130	296	0.00	360	**24**	296	0.00	1484	32	296	0.00	397	8	296	0.00	57	296	0.00
0.8	50	3	125	294	0.00	1888	141	294	0.00	316	**30**	294	0.00	1655	33	294	0.00	351	8	294	0.00	71	294	0.00
0.8	50	4	82	270	0.00	974	72	270	0.00	105	35	270	0.00	1747	**15**	270	0.00	116	9	270	0.00	86	270	0.00
0.8	50	5	89	278	0.00	1326	206	278	0.00	659	**46**	278	0.00	2611	58	278	0.00	744	9	278	0.00	77	278	0.00

6. Conclusions and future work

In this paper, we presented exact and heuristic solution algorithms for the recently introduced *(minimum) weighted total domination problem (WTDP)* (see [22]). The WTDP is a problem from the family of domination problems, which are among the most basic combinatinatorial problems in graph optimization. In the WTDP we are not just concerned with the concept of domination (i.e., finding a vertex-set $D \subset V$ for a given graph $G = (V, E)$, such that each vertex is either in D or adjacent to it), but with the stronger concept of *total domination*, which imposes that for each vertex $v \in D$, there is also a neighbor of v in D (i.e., the vertices of D also need to be dominated by D). In the WTDP, we have a weight function associated with the vertices and edges of the graph. The goal is to find a total dominating set D with minimal weight. The weight counted in the objective is the weight of the vertices selected for D, the weight of the edges between vertices in D, and for each vertex in $V \setminus D$, the smallest weight of an edge between it and a vertex in D.

We introduced two new Mixed-Integer Programming models for the problem, and designed solution frameworks based on them. These solution frameworks include valid inequalities, starting heuristics and primal heuristics. In addition, we also developed a genetic algorithm (GA), which is based on a greedy randomized adaptive search procedure (GRASP) version of our starting heuristic.

In a computational study, we compared our new exact approaches to the previous MIP approached presented in [22] and also analyzed the performance of the GRASP and GA. The study revealed that our exact solution algorithms are up to 500 times faster compared to the exact approaches of [22] and instances with up to 125 vertices can be solved to optimality within a timelimit of 1800 seconds. Moreover, the GRASP and GA also works well and often find the optimal or a near-optimal solution within a short runtime. In the study, we also investigated the influence of different instance-characteristics, e.g., density and weight-structure on the performance of our approaches. Instances, where the edge weights are in a larger range compared to the vertex weights turned out to be the most difficult for our algorithms, while high density also plays a role in making instances difficult.

The attained results confirm that domination problems are computationally challenging and, therefore, require the combined effort of MIP-based and heuristic approaches in order to tackle more difficult instances. Therefore, we believe that the development of further modeling and algorithmic advances for domination problem variants is an interesting venue for future work as these problems are relevant both from the methodological and practical point of view.

Acknowledgments E. Álvarez-Miranda acknowledges the support of the National Commission for Scientific and Technological Research CONICYT, Chile, through the grant FONDECYT N.1180670 and through the Complex Engineering Systems Institute PIA/BASAL AFB180003.

References

References

[1] A. Billionnet. Different formulations for solving the heaviest k-subgraph problem. *INFOR: Information Systems and Operational Research*, 43(3):171–186, 2005.

[2] F. Bonomo, J. Marenco, D. Saban, and N. Stier-Moses. A polyhedral study of the maximum edge subgraph problem. *Discrete Applied Mathematics*, 160(18):2573–2590, 2012.

[3] E. Cockayne and S. Hedetniemi. Towards a theory of domination in graphs. *Networks*, 7(3):247–261, 1977.

[4] E. Cockayne, R. Dawes, and S. Hedetniemi. Total domination in graphs. *Networks*, 10(3):211–219, 1980.

[5] S. Dey and M. Molinaro. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170(1):237–266, 2018.

[6] D. Du and P. Wan. *Connected Dominating Set: Theory and Applications*, volume 77 of *Springer Optimization and Its Applications*. Springer, 1st edition, 2013.

[7] D. Erwin. Dominating broadcasts in graphs. *Bulletin of the Institute of Combinatorics and its Applications*, 42:89–105, 2004.

[8] M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2016.

[9] C. Giap and D. Ha. Parallel genetic algorithm for minimum dominating set problem. In *2014 International Conference on Computing, Management and Telecommunications (ComManTel)*, pages 165–169. IEEE, 2014.

[10] W. Goddard and M. Henning. Independent domination in graphs: A survey and recent results. *Discrete Mathematics*, 313(7):839–854, 2013.

[11] A. Hagberg, P. Swart, and D. Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory, Los Alamos, NM, United States, 2008.

[12] T. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. CRC press, 2013.

[13] A. Hedar and R. Ismail. Hybrid genetic algorithm for minimum dominating set problem. In D. Taniar, O. Gervasi, B. Murgante, E. Pardede, and B. Apduhan, editors, *International conference on computational science and its applications*, volume 6019 of *Lecture Notes in Computer Science*, pages 457–467. Springer, 2010.

[14] M. Henning. Restricted total domination in graphs. *Discrete Mathematics*, 289(1-3):25–44, 2004.

[15] M. Henning. A survey of selected recent results on total domination in graphs. *Discrete Mathematics*, 309(1):32–63, 2009.

[16] M. Henning and A. Yeo. *Total domination in graphs*. Springer, 2013.

[17] L. Kang. *Variations of Dominating Set Problem*, pages 3363–3394. Springer, 2013.

[18] O. Kramer, editor. *Genetic Algorithm Essentials*, volume 679 of *Series on Applied Optimization*. Springer, 1st edition, 2017.

[19] G. Laporte, S. Nickel, and F. S. da Gama. *Location science*, volume 528. Springer, 2015.

[20] R. Laskar, J. Pfaff, S. Hedetniemi, and S. Hedetniemi. On the algorithmic complexity of total domination. *SIAM Journal on Algebraic Discrete Methods*, 5(3):420–425, 1984.

[21] A. N. Letchford and M. M. Sørensen. A new separation algorithm for the boolean quadric and cut polytopes. *Discrete Optimization*, 14:61–71, 2014.

[22] Y. Ma, Q. Cai, and S. Yao. Integer linear programming models for the weighted total domination problem. *Applied Mathematics and Computation*, 358:146–150, 2019.

[23] E. Macambira and C. de Souza. The edge-weighted clique problem: valid inequalities, facets and polyhedral computations. *European Journal of Operational Research*, 123(2):346–371, 2000.

[24] J. Nacher and T. Akutsu. Minimum dominating set-based methods for analyzing biological networks. *Methods*, 102:57–63, 2016.

[25] O. Ore. *Theory of Graphs*, volume 38 of *Colloquium Publications*. American Mathematical Society, 1st edition, 1962.

[26] M. Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Mathematical Programming*, 45(1-3):139–172, 1989.

[27] P. Pinacho, C. Blum, and J. Lozano. The weighted independent domination problem: Integer linear programming models and metaheuristic approaches. *European Journal of Operational Research*, 265(3): 860–871, 2018.

[28] R. Rahmaniani, T. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[29] D. Rengaswamy, S. Datta, and S. Ramalingam. Multiobjective genetic algorithm for minimum weight minimum connected dominating set. In A. Abraham, P. Muhuri, A. Muda, and N. Gandhi, editors, *International Conference on Intelligent Systems Design and Applications*, volume 736 of *Advances in Intelligent Systems and Computing*, pages 558–567. Springer, 2017.

[30] M. Resende and C. Ribeiro, editors. *Optimization by GRASP*. Springer, 1st edition, 2016.

[31] J. Rolfes. Copositive formulations of the dominating set problem and applications master thesis. Master's thesis, Department of Mathematics,Faculty of Science, University of Cologne, 2014.

[32] J. Sarubbi, C. Mesquita, E. Wanner, V. Santos, and C. Silva. A strategy for clustering students minimizing the number of bus stops for solving the school bus routing problem. In *2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1175–1180, 2016.

[33] P. Sun and X. Ma. Dominating communities for hierarchical control of complex networks. *Information Sciences*, 414:247–259, 2017.

[34] S. Sundar. A steady-state genetic algorithm for the dominating tree problem. In G. Dick, W. Browne, P. Whigham, M. Zhang, L. Thu, B. Hisao, I. Yaochu, J. Xiaodong, L. Yuhui, S. Pramod, S. amd Kay, C. Tan, and K. Tang, editors, *Asia-Pacific Conference on Simulated Evolution and Learning*, volume 8886 of *Lecture Notes in Computer Science*, pages 48–57. Springer, 2014.

[35] P. Wan, K. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1597–1604, 2002.

[36] F. Wesselmann and U. Stuhl. Implementing cutting plane management and selection techniques. Technical report, University of Paderborn, Germany, 2012.