RTS

Floris van der Hout
Department of Informatics, Utrecht University, the Netherlands

Abstract

Submission for the PACE challenge 2024 – exact track. This solver exactly solves the AMDS (Augmented minimum dominating set) problem, which is a more generalised minimum dominating set problem.

The algorithm starts by preprocessing the instance graphs using a combination of reduction rules proposed by Alber [1] and Xiong and Xiao [3]. Then, the graph is split into smaller components. The treewidth of each component is then approximated; if it is low enough, the component gets solved with a treewidth solver [2]. If the treewidth is too high, the CP-SAT solver from OR-Tools is used to tackle the AMDS problem.

2012 ACM Subject Classification Theory of computation -> Graph Theory

Keywords and phrases PACE, Treewidth, minimum dominating set, reduction rules, SAT, exact solver

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Brief description of the Algorithm

The RTS solver consists of three main components:

- Reduction rules [1][3].
- Treewidth approximation and solver [2].
- CP-SAT solver.

First, the input instance is preprocessed using a set of reduction rules (Rule X.1 to X.3 by Xiong and Xiao [3], and the reduction scheme $L.l$ from Alber [1]. These rules simplify the problem by reducing the number of undetermined vertices. Importantly, reductions are not limited to vertex removals and selections alone. To accommodate this extra context, we define a new problem variant called the Augmented minimum dominating set problem (AMDS). This formulation incorporates the augmented context produced by the reduction and also allows seamless use of different reduction rule sets.

After a predefined timeout, or once the strongest reduction rule has been applied, the solver proceeds by decomposing the instance into connected components. Each component can be solved independently, and the individual solutions are later combined to reconstruct a solution for the original instance.

For each component, the treewidth is approximated using the HTD library, and a corresponding nice tree decomposition is generated. If the approximated treewidth is low enough, the component is solved using a dynamic programming approach [2].

Otherwise, the components are solved using the award-winning CP-SAT solver from OR-Tools, for which we provide a SAT formulation made for the AMDS problem.

2 Augmented minimum dominating set

AUGMENTED MINIMUM DOMINATING SET PROBLEM (AMDS). This formulation will integrate the contextual elements of the reduction rule sets developed by Alber [1] and by Xiong and Xiao [3]. This integration will enable the removal and omission of vertices, allowing both reduction rule sets to be applied to the same instance.

© Floris van der Hout;
licensed under Creative Commons License CC-BY 4.0

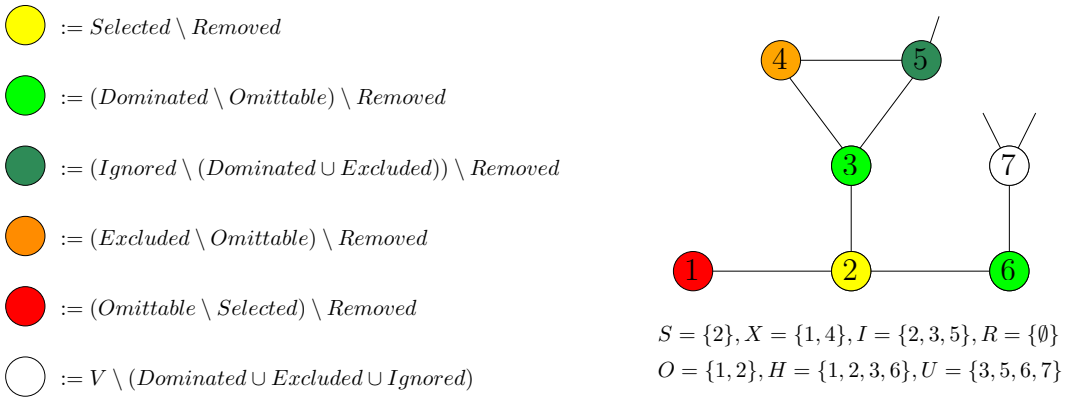

Figure 1 Visualisation of the Augmented minimum dominated set (AMDS) problem, including a legend indicating the context of each node.

Given an instance $\mathcal{I}_{\mathcal{AMDS}} = (G, S, X, I, H, R)$ of AMDS, let $G = (V, E)$ be a simple, undirected graph without loops. The AMDS formulation includes five additional context sets:

- S **(selected)**: the set of vertices that must be included in the minimum dominating set D.
- X **(excluded)**: the set of vertices not permitted to be part of the minimum dominating set D, as more optimal alternatives exist.
- I **(Ignored)**: the set of vertices that do not need to be explicitly dominated, as they are guaranteed to be dominated indirectly—there exists at least one other vertex whose domination necessarily results in the domination of the ignored vertex as well.
- H **(dominated)**: the set of vertices that are already dominated (i.e., have been "hit").
- R **(removed)**: the set of vertices removed from the original instance, hence $V \cap R = \emptyset$, and these vertices cannot be part of the minimum dominating set D.

Some context sets won't be stored explicitly as they can be inferred from the others:

- U **(undetermined)**: $U = V \setminus (S \cup X)$; the set of vertices whose inclusion or exclusion from the minimum dominating set D is still undetermined.
- O **(omittable)**: $O = (S \cup (X \cap H)) \setminus R$; The set of vertices that have not yet been removed but are eligible for removal. This is useful as the rule set X does not remove vertices, and can identify ones that could be removed. For example, a vertex that is excluded and already dominated no longer needs to be dominated and is also not allowed to dominate others, making it safely removable.

An instance $\mathcal{I}_{\mathcal{AMDS}}$ can also be visualised as a graph, as shown in Figure 1. Vertices may belong to multiple sets simultaneously. However, some information is deemed more important than others. For instance, if a vertex belongs to S, it also belongs to $R \cup O$. Nevertheless, its membership in the set S is deemed more important for the reader, which is highlighted in the visualisation. The whole hierarchy can be inferred from the legend in Figure 1.

3 Reduction rules

As part of the preprocessing, we employ a combination of reduction rules proposed by Alber [1] and the reduction rule set X introduced by Xiong and Xiao [3]. The reduction rule set X

offers an excellent foundation for preprocessing the graph; however, it's important to note that during the execution, removing vertices is not permitted—only omitting them is allowed.

3.1 Reduction rule set X

Rule X.1 (Single Dominator). *Given an instance $\mathcal{I}_{\mathcal{AMDS}} = (G, S, X, I, H, R)$ of AMDS, if an undominated vertex v has only one dominator u, we add u to S.*

Rule X.2 (Subset Coverage). *For an instance $\mathcal{I}_{\mathcal{AMDS}} = (G, S, X, I, H, R)$ of AMDS, if there exists two distinct undetermined vertices u, v such that the coverage of u is a subset of $N[v]$, i.e., $C(u) = N[u] \setminus (N[S] \cup I) \subseteq N[v]$, we add u to X.*

Rule X.3 (Ignorable Vertices). *For an instance $\mathcal{I}_{\mathcal{AMDS}} = (G, S, X, I, H, R)$ of AMDS, if there exist two distinct undominated vertices u, v such that $D(u) = N[u] \setminus X \subseteq N[v]$, we add v to I.*

3.2 Reduction scheme $L.l$

The reduction rule $L.l$ is based on analysing the local structure of the graph. Consider a fixed set of l vertices, $V_l := v_1, \ldots, v_l \subseteq V$. The joint neighbourhood $N(V_l)$ can be partitioned into three subsets.

$$N_{exit}(V_l) = \{u \in N(V_l) \mid N(u) \setminus N[V_l] \neq \emptyset\}, \tag{1}$$

$$N_{guard}(V_l) = \{u \in N(V_l) \setminus N_{exit}(V_l) \mid N(u) \cap N_{exit}(V_l) \neq \emptyset\}, \tag{2}$$

$$N_{prison}(V_l) = N(V_l) \setminus (N_{exit}(V_l) \cup N_{guard}(V_l)). \tag{3}$$

The rule L.l aims to reduce the number of undetermined vertices by identifying subsets $\mathcal{W} \subseteq 2^{V_l}$ that dominate all vertices in N_{prison} more effectively than any other subsets of the same size, thus preserving the domination number.

Given that a reduction has been found for the subsets $\mathcal{W} \subseteq 2^{V_l}$, to preserve the domination number, there must exist a subset $W \in \mathcal{W}$ such that $W \subseteq D$ in the optimal dominating set D.

This constraint can be represented as a Boolean formula, as outlined in Definition 1. These Boolean constraints can then be modelled as a gadget (see Definition 2), which can be integrated into the graph, allowing for the exhaustive application of the reduction rule.

Definition 1. *Let $\mathcal{W} \subseteq 2^V$ be a collection of subsets of V. The constraints with $\mathcal{W}$ is a boolean formula F_W in disjunctive normal form: $F_{\mathcal{W}} = \bigvee_{W \in \mathcal{W}} \bigwedge_{w \in W} w$.*

The constraint should be as compact as possible to minimise the size of the gadget. We define "compactification" using the following concept: a collection of subsets $\mathcal{W} \subseteq 2^V$ is considered compact if no two elements within $\mathcal{W}$ are subsets of one another.

Lemma 1. *Let $\mathcal{W} \subseteq 2^V$. There exists a minimal compact subset $\widehat{\mathcal{W}} \subseteq \mathcal{W}$ such that $F_{\mathcal{W}}$ is logically equivalent to $F_{\widehat{\mathcal{W}}}$ and $\widehat{\mathcal{W}}$ can be found in polynomial time.*

Definition 2. *Let $G = (V, E)$ be a graph, and let $\mathcal{W} = \{W_1, \ldots, W_s\} \subseteq 2^V$ be a collection of subsets of V. Define $l = |\bigcup_{i=1}^{s} W_i|$ and $p = \prod_{i=1}^{s} |W_i|$. $F_{\mathcal{W}}$-gadget is constructed as follows: Introduce a set of p selector vertices $S = \{u_{(x_1,\ldots,x_s)} \mid x_i \in \{1, ..., |W_i|\}$. If $p < l$, also introduce $l - p$ blocker vertices $B = \{b_1, \ldots, b_{l-p}\}$. By connecting the newly introduced edges to those in G in a specific way, one can enforce constraints within the graph. The following edges are added between the newly introduced vertices and the original vertices of G. For each $i \in \{1, \ldots, s\}$, let $W_i = \{w_{i1}, ..., w_{i|W_i|}\} \in \mathcal{W}$. For each $j \in \{1, \ldots, |W_i|\}$, the following edges are added.*

- *An edge between w_{ij} and every selector vertices in $\{u_{(x_1,\ldots,x_s)} \in S \mid x_i = j\}$.*
- *An edge between w_{ij} and each blocker vertex in B.*

We denote the resulting augmented graph by $G \oplus F_{\mathcal{W}}$.

The core idea of the gadget is that the formula $F_{\mathcal{W}}$ is satisfied when all selector vertices are dominated. To ensure that only vertices from the original graph are included in the dominating set, blocker vertices are introduced to prevent any selector vertex from being part of the optimal dominating set D.

Definition 3. *For two sets $\emptyset \neq W, W' \subseteq V$, we say that W is better than W' if $|W| \leq |W'|$ and $N[W] \supseteq N[W']$. If W is better than W', we write $W \leq W'$. If $W' = \emptyset$ and $W \neq \emptyset$, then always $W \leq W'$.*

Rule L.l *Consider l pairwise distinct vertices $V_l := \{v_1, ..., v_l\} \subseteq V$ and suppose $N_{prison}(V_l) \neq \emptyset$.*

- *Compute the set $\mathcal{W} := \{W \subseteq V_l | N_{prison}(V_l) \subseteq N[W]\}$ of all vertex subsets of V_l that dominate all prisoner vertices $N_{prison}(V_l)$, and the set of all alternatives to dominate $N_{prison}(V_l)$ with less than l vertices: $\mathcal{W}_{altern} := \{W \subseteq N[N_{prison}(V_l)] \mid N_{prison}(V_l) \subseteq N[W]$ and $|W| < l\}$.*
- *Compute the compactifications of $\widehat{\mathcal{W}}$ of $\mathcal{W}$ and $\widehat{\mathcal{W}}_{altern}$ of $\mathcal{W}_{altern}$.*
- *If $(\forall W \in \widehat{\mathcal{W}}_{altern} \exists W' \in \widehat{\mathcal{W}} : W' \leq W)$, then*
  - *remove the vertices $\{V \in N_{guard}(V_l) \cup N_{prison}(V_l) \mid N[v] \subseteq \bigcap_{W \in \widehat{\mathcal{W}}} N[W]\}$,*
  - *put an $F_{\widehat{\mathcal{W}}}$ to G for the constraint associated with $\widehat{\mathcal{W}}$.*

Alber's reduction rule scheme is more powerful but also takes more execution time because it analyses the local structure of the graph. $L_{practical}.l$ performs better when omitted vertices can be subsequently removed; therefore, after the application of the reduction rule set X, the omitted vertices can be eliminated.

As the value of l increases, the strength of the rule also increases, but it becomes more computationally expensive. This emphasises the importance of initially applying cheaper reductions to minimise the number of vertices that need to be considered.

A timer sets a limit on preprocessing time; if exceeded, it moves to the next step.

Algorithm 1 gives an overview of the sequence of reduction rules.

4 Dynamic programming approach bounded by treewidth for AMDS

Given a graph G with a corresponding *nice* tree decomposition (X', T') of width at most k. One can compute the domination number (and minimum dominating set) of a graph G in $O^*(4^k)$. This can be accomplished with a dynamic programming algorithm [2].

Algorithm 1 Reduce($\mathcal{I}_{AMDS}$)

procedure REDUCE($\mathcal{I}_{AMDS}$)
  $\mathcal{I}_{AMDS} \leftarrow X(\mathcal{I}_{AMDS})$
  bulk-remove all omitted vertices ($v \in O$).
  for $l = 2$ **to** desired maximum depth **do**
    $\mathcal{I}_{AMDS} \leftarrow L.l(\mathcal{I}_{AMDS},\ l)$
  end for
  Bulk-remove all omitted vertices ($v \in O$).
  return $\mathcal{I}_{AMDS}$.
end procedure

The original recursive formulations are intended for instances that do not include context sets. To adapt these formulations for use with instances of $\mathcal{I}_{AMDS}$, which do involve context sets, we need to modify two specific formulations: the "introduce vertex" and the "forget vertex." These modifications are essential for ensuring the correctness while effectively incorporating the context sets.

A bit of context: for a note $t \in T'$ and a colouring s, (which assigns each vertex in the bag X'_t a colour of black (0), white (1) or grey ($\hat{0}$): the value $c[t, s]$ represents the size of the minimum compatible set. This is the minimum dominating set of the induced subgraph of node t (more on this in the book [2]), under the constraint that the colouring is fixed. Here, $c(v)$ denotes the colour assigned to a vertex v.

Assuming $O = \emptyset$, it is necessary to incorporate the three context sets: S, X, and H into the original formulation to ensure the solution remains optimal for $\mathcal{I}_{AMDS}$.

For an *introduce vertex* node $t \in T'$ with child $t' \in T'$, we have $X_t = X_{t'} \cup \{v\}$, so vertex v is introduced at bag t. The original formulation assumes $v \notin H$, which always holds in the absence of context sets. However, in an instance of $\mathcal{I}_{AMDS}$, we must distinguish between the cases where $v \in H$ and $v \notin H$. If $v \notin H$, we can apply the original *introduce vertex* formulation without change. Conversely, if $v \in H$, we need to adjust the formulation to recognise that v has already been dominated before this point. Therefore, if $v \in H$, the *introduce vertex* recursive formula changes as follows:

$$c[t, s] = \begin{cases} c[t', s \mid X_{t'}] & \text{when} \quad s(v) = 0, \\ c[t', s \mid X_{t'}] & \text{when} \quad s(v) = \hat{0}, \\ 1 + c[t', s \mid X_{t'}] & \text{when} \quad s(v) = 1. \end{cases}$$

This modification is valid because $s(v) = 0$ indicates that v is not in the compatible set. In the original formulation, this would lead to an infeasible partial solution, since v is introduced without any incident edges and thus cannot be dominated unless it dominates itself. However, if $v \in H$, means it is already dominated prior, then assigning $s(v) = 0$ becomes valid as no additional domination is needed for v, as the requirement has already been satisfied.

For an *Forget* node $t \in T'$ with child $t' \in T'$ we have $X'_t = X'_{t'} \setminus \{w\}$. so vertex w is removed at bag t. The original formulation assumes that $w \notin X$, which always holds in the absence of context sets. If $w \notin X$, we can apply the original *forget* formulation without modification.

However, if $w \in X$, we need to adjust the formulation to account for the fact that x is not allowed to be in the compatible set. Therefore, if $w \in X$ the *forget* recursive formula changes as follows:

$$c[t, s] = c[t', s_{w \to 0}]$$

If the algorithm finishes successfully, the solution to the entire instance is obtained by taking the *minimum compatible set* stored at the root whose colouring contains no grey vertices and combining it with the set S of selected vertices.

5 ILP/SAT formulation AMDS

Suppose we have an AMDS instance $\mathcal{I}_{\mathcal{AMDS}} = (G, S, X, I, H, R)$. Let $U = \{u_1, ..., u_n\}$ be the inferred set of undetermined vertices, where $|U| = n$. We associate a binary variable x_i to each undetermined vertex such that:

$$x_i = \begin{cases} 1, & \text{if } u_i \in D. \\ 0, & \text{if } u_i \notin D. \end{cases}$$

The AUGMENTED DOMINATING SET PROBLEM can be formulated using the equations below as ILP/SAT constraints.

$$min \sum_{i=1}^{n} u_i + |S| \tag{4}$$

$$\sum_{u \in (N[v] \setminus (X \cup S))} x_u \geq 1, \quad \forall v \in (V \setminus (H \cup I)) \tag{5}$$

$$u_i \in [0, 1], \quad \forall u_i \in U \tag{6}$$

Verifying the correctness of this formulation is simple, as we only need to determine whether the undetermined vertices are part of the minimum dominating set D, since the inclusion or exclusion of all other vertices is already known. All vertices already dominated or ignored, i.e., those in $H \cup I$, do not require a constraint stating that they must have at least one neighbour in the minimum dominating set D since they are already dominated. For the undominated vertices, only the undetermined vertices in their closed neighbourhood can potentially dominate them. This of these all combined ensures that the AMDS is optimal.

References

1 Jochen Alber, Britta Dorn, and Rolf Niedermeier. A general data reduction scheme for domination in graphs. SOFSEM 2006, pages 137–147, 2006. doi:/10.1007/11611257_11.

2 Marek Cygan, Fedor V Fomin, Daniel Marx, Saket Saurabh, Lukasz Kowalik, Daniel Lokshtanov, and Marcin Pilipczuk. *Parameterized Algorithms*. Springer, 1st edition, July 2015.

3 Ziliang Xiong and Mingyu Xiao. Exactly solving minimum dominating set and its generalization. IJCAI-24, pages 7056–7064, 2024. Main Track. doi:10.24963/ijcai.2024/780.