


# PACE Solver Description: UzL Solver for Dominating Set and Hitting Set

Max Bannach  

European Space Agency, AI and Data Science Section, Noordwijk, The Netherlands

Florian Chudigiewitsch 

Institute for Theoretical Computer Science, University of Lübeck, Germany

Marcel Wienöbst<sup>1</sup> 

Institute for Theoretical Computer Science, University of Lübeck, Germany

---

## Abstract

This document contains a short description of our solver for the dominating set and hitting set problems that we submitted to the exact tracks of the PACE Challenge 2025. The solver is based on a straightforward MaxSAT formulation supplemented by hitting-set-based reduction rules. It utilizes a clique solver if the reduced instance is a (small) input for the vertex cover problem and tries to match certain lower bounds by expressing the reduced instance as a SAT problem.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** exact algorithms, dominating set, hitting set

**Digital Object Identifier** 10.4230/LIPIcs.TBD.2025.0

**Category** PACE Solver Description

**Supplementary Material** Code repository:

**URL:** <https://github.com/mwien/PACE2025>

## 1 Introduction

Dominating set and hitting set are classical NP-hard problems [5]. In the *dominating set problem* we seek, for a given undirected graph  $G = (V, E)$ , a minimum-size set of nodes  $S$  such that for every node  $u \in V$ , the node  $u$  or one of its neighbors is in  $S$ . In the *hitting set problem*, the task is to find in a given hypergraph  $H = (V, E)$  a minimum-size set of nodes  $S$  such that every edge  $\{u_1, u_2, \dots\} \in E$  has a non-empty intersection with  $S$ . Clearly, any dominating set instance can be expressed as a hitting set instance with  $E = \{N[u] \mid u \in V\}$ , where  $N[u]$  is the closed neighborhood of  $u$ .

The solver we describe can be used to tackle both problems. However, internally, any dominating set input is directly translated to the corresponding hitting set instance as described above. Thus, from now on, we solely focus on the hitting set problem.

## 2 Reduction Rules for Hitting Set

We use four reduction rules for hitting set exhaustively:

1. If an edge consists of a single node  $u$ , then add  $u$  to  $S$  and remove all edges containing  $u$ .
2. If a node is in exactly one edge, and this edge has size at least two, then discard the node and remove it from the edge.
3. If an edge is a subset of another edge, keep only the smaller edge.

---

<sup>1</sup> Corresponding author.



4. If there are two nodes  $u$  and  $v$  such that  $u \in e$  for all edges  $e$  containing  $v$ , then delete  $v$ .

The correctness of these rules is immediate: In Rule 1 the node  $u$  needs to be in any solution; in Rule 3 the discarded edge is hit by any solution; and for Rules 2 and 4, we can guarantee that there is always a solution without the discarded node by an exchange argument.

### 3 MaxSAT Formulation of Hitting Set

The hitting set problem can naturally be expressed as a Partial MaxSAT problem [3]. This problem allows for both *soft* clauses and *hard* clauses, where the hard clauses must all be satisfied and the goal is to satisfy as many soft clauses as possible. Let  $H = (V, E)$  be a hypergraph, then we formulate the hitting set problem as

$$\begin{array}{ll} \text{Soft Clauses} & \bigwedge_{u \in V} (\neg u), \\ \text{Hard Clauses} & \bigwedge_{\{u_1, u_2, \dots\} \in E} (u_1 \vee u_2 \vee \dots). \end{array}$$

We solve this formulation using the EvalMaxSAT solver [1]. For the dominating set track, we additionally preprocess the formula using maxpre2 [6, 8]. We do not perform this preprocessing for the hitting set instances because it did not yield performance gains and was sometimes counterproductive. Moreover, we tweaked internal parameters of EvalMaxSAT to the problems at hand and, notably, bound the time spent in the conflict minimization phase depending on the conflict size.

### 4 Using Maximum Clique and SAT Solvers

In the case that the reduced instance is a *vertex cover* instance (e.g.,  $|e| \leq 2$  for all  $e \in E$ ) and has at most 350 nodes, we solve the problem in a different way. Inspired by the results of the PACE 2019 [4], we use a maximum clique solver, namely mcqd [7], on the complement graph of  $H$ . Every vertex not in the maximum clique must be in a minimum-size vertex cover. Note that we do not use this strategy for instances with more than 350 nodes, where the size of the usually very dense complement graph could be prohibitively large. Here, we use the MaxSAT formulation stated above.

Before we start the MaxSAT solver, we try to generate a strong lower bound which, eventually, we can match using a SAT formulation. In detail, we greedily search for a perfect matching in the hypergraph, that is, we look for  $|V|/2$  hyperedges of size two such that every node is in exactly one of these edges. Clearly, if we find such a matching, any solution  $S$  must have size at least  $|V|/2$ . In this case, we check using the SAT solver kissat [2] whether this lower bound can be matched by a solution that selects exactly one variable per matching edge (we introduce one variable for every edge in the matching whose truth value describes which of the two nodes in the edge is in the solution; then we simply add a clause for every remaining hyperedge to encode that it gets hit as well).

### 5 Sketch of Correctness

For a brief sketch of a correctness proof, observe that the reduction rules are sound and the MaxSAT formulation for the hitting set problem is immediate (and well-known). Solving vertex cover with a clique-solver in the complement graph is a textbook trick. If the SAT solver finds an assignment of size  $|V|/2$  in the instances discussed above, then clearly this solution is a hitting set and furthermore an optimal solution as it matches the lower bound.

As external solvers, we rely on well-established and open-source software: EvalMaxSAT [1], kissat [2], maxpre2 [8, 6], and mcqd [7].

---

## References

---

- 1 Florent Avellaneda. A short description of the solver evalmaxsat. *MaxSAT Evaluation*, 8:364, 2020.
- 2 Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL, Gimsatul, IsaSAT and Kissat entering the SAT Competition 2024. In Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proceedings of the SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions*, pages 8–10, 2024.
- 3 Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.
- 4 M. Ayaz Dzulfikar, Johannes K. Fichte, and Markus Hecher. The PACE 2019 Parameterized Algorithms and Computational Experiments Challenge: The Fourth Iteration. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*, pages 25:1–25:23, 2019.
- 5 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 6 Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Clause redundancy and preprocessing in maximum satisfiability. In *Proceedings of the 11th International Joint Conference on Automated Reasoning (IJCAR ’22)*, volume 13385 of *Lecture Notes in Computer Science*, pages 75–94. Springer, August 2022.
- 7 Janez Konc and Dušanka Janežic. An improved branch and bound algorithm for the maximum clique problem. *proteins*, 4(5):590–596, 2007.
- 8 Tuukka Korhonen, Jeremias Berg, Paul Saikko, and Matti Järvisalo. Clause redundancy and preprocessing in maximum satisfiability. In Serge Gaspers and Toby Walsh, editors, *Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing, (SAT ’17)*, volume 10491 of *Lecture Notes in Computer Science*, pages 449–456. Springer, 2017.