PACE Solver Description: Root

Canhui Luo

Huazhong University of Science and Technology, Wuhan, China

Qingyun Zhang

Huazhong University of Science and Technology, Wuhan, China

Zhouxing Su

Huazhong University of Science and Technology, Wuhan, China

Zhipeng Lü

Huazhong University of Science and Technology, Wuhan, China

Abstract

We present a unified heuristic solver for the PACE 2025 challenge, addressing both the dominating set and hitting set problems by reducing them to the unicost set covering problem. Our solver applies standard reduction rules, a multi-round frequency-based greedy initializer, and a local search guided by adaptive element weights. Additional techniques, such as component-level exact solving and swap restriction, further enhance performance. In the final official evaluation, our proposed solver achieved second place in the heuristic track for the dominating set problem of the PACE 2025 challenge, while securing first place in the heuristic track for the hitting set problem.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Computing methodologies → Search methodologies

Keywords and phrases PACE 2025, Dominating Set, Hitting Set, Heuristic Optimization, Weighted Local Search

Digital Object Identifier 10.4230/LIPIcs...

Supplementary Material *Software (Source Code)*: `https://github.com/lxily/PACE2025.DS-HS`

Acknowledgements We want to thank the organizers of PACE 2025 challenge and all other participants for creating such an engaging challenge.

1 Challenge Problem and Transformation

The PACE 2025 challenge involves two fundamental NP-hard problems: the dominating set problem and the hitting set problem. The dominating set problem seeks to select as few vertices as possible in a given graph such that every other vertex is adjacent to at least one selected vertex. The hitting set problem requires selecting as few elements as possible from a set system such that each set contains at least one selected element. In fact, the former can be regarded as a special case of the latter. We unify these two problems by transforming them into a unicost set covering problem, as follows:

- In the dominating set problem, each vertex v_i is represented as both a set s_i and an element e_i. Each set s_i covers all elements corresponding to its neighboring vertices.
- In the hitting set problem, each original set s_i^o is mapped to an element e_i, and each original element $e_j^o \in s_i^o$ is mapped to a set s_j covering e_i.

Subsequently, the optimization objectives of the two original problems are unified as minimizing the number of selected sets that cover all elements, although such a transformation inevitably loses certain characteristic information of the original problems.

2 Solver Methodology

In this section, we present our heuristic solver for the set covering problem. It integrates instance reduction, initialization, element-weighted local search, and two simple but effective optimization strategies that further enhance the search process.

2.1 Preprocessing via Reduction Rules

Considering that the datasets in the PACE 2025 Challenge may contain millions of sets and elements, applying possible reductions to them is highly beneficial for the subsequent search. We employ three classical reductions without sacrificing optimality [2]:

- **Element Dominance:** If all sets that cover element e_i also cover element e_j (where $i! = j$), then e_j can be safely removed.
- **Set Dominance:** If all elements covered by set s_i are also covered by set s_j (where $i! = j$), then s_i can be safely removed.
- **Mandatory Coverage:** If an element e_i is uniquely covered by one set s_i, then s_i must be selected, and all elements covered by s_i can be removed.

These reduction rules preserve the existence and structure of optimal solutions. By iteratively applying them, the problem size can typically be reduced significantly. We solve the reduced instance and merge the optimized result with the sets fixed during the reduction process to form the final solution.

2.2 Frequency-Guided Initialization

We define the *importance score* $\mathrm{IS}(e_i) = \frac{1}{\mathrm{freq}(e_i)}$ of an element as the reciprocal of its frequency (i.e., the number of sets that can cover it). Then, the score of a set $\mathrm{IS}(s_i) = \sum_{e_j \in \mathrm{uncovered}(s_i)} \mathrm{IS}(e_j)$ is the sum of the scores of all currently uncovered elements it covers. The greedy algorithm iteratively selects the set with the highest score, thereby prioritizing the coverage of hard-to-cover elements. After each selection, scores of related sets are updated to reflect the new uncovered element set.

Since the number of sets that can cover an element differs from the number of sets that actually cover it in a solution, the original importance score may be biased. To correct this, after obtaining a feasible solution, we refine the score of each element e_i by multiplying it with a scaling factor, i.e., $\mathrm{IS}'(e_i) = \mathrm{IS}(e_i) \cdot \frac{c_{\max}}{c_i}$, where c_i is the number of times element e_i is actually covered in the solution, and $c_{\max}$ is the maximum coverage count among all elements. Through multiple rounds of score refinement and reconstruction, higher-quality initial solutions can typically be obtained, albeit with increased construction time.

2.3 Element-Weighted Local Search

Starting from a feasible initial cover, we iteratively remove a randomly selected set and attempt to reconstruct a feasible cover without increasing the number of sets used. Once such a reconstruction is successful, we obtain an improved solution. Thus, our optimization focuses on how to achieve full coverage using a fixed number of sets.

2.3.1 Weighting Technique

Weighting techniques have demonstrated strong effectiveness in various set covering–related problems [2, 3, 4]. Our solver adopts a similar strategy: we assign weights to currently uncovered elements, and seek to minimize the total weight of uncovered elements during local search. Compared to minimizing just the count of uncovered elements, the weighted objective yields a smoother search landscape and better optimization performance.

2.3.2 Neighborhood Search

The neighborhood is defined by a pairwise swap operation: removing one set from the current solution and adding another. In each iteration:

- Randomly select an uncovered element e such that the subsequent swap operation ensures it will be covered. This random selection strategy enhances the diversification of the search while simultaneously reducing the evaluation time complexity.
- For each element e, we consider adding a set that covers it and simultaneously removing one set from the current solution. We evaluate all swap pairs and select the one that minimizes the total weight of uncovered elements after the move.

When the search reaches a local optimum, we increase the weight of a random uncovered element, encouraging its coverage in future iterations. Tabu search is a well-known metaheuristic for combinatorial optimization [1]. We integrate a one-iteration recency-based tabu mechanism that temporarily forbids recently involved sets from participating in swaps, thus encouraging search diversification and preventing cycling. Upon finding a new feasible solution (i.e., all elements are covered), we proactively remove a random set, and the optimization moves to the new bottleneck of one fewer set. The search procedure is repeated until the time limit is reached and the best solution found so far is returned.

2.4 Additional Enhancements

For instances that contain multiple connected components after reduction, we introduce two dedicated optimization strategies:

- Reducing the number of connected components: We apply a simple branch-and-bound algorithm to exactly solve connected components that involve fewer than k sets (with $k = 23$ in our implementation).
- Restricting active components: Initially, the removal of a set introduces uncovered elements in only one connected component. In subsequent swap iterations, if the added set fails to restore full coverage within this component while the removed set belongs to a different component, we revert the current component to its last feasible state.

References

1 Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

2 Zhouxing Su, Qingyun Zhang, Zhipeng Lü, Chu-Min Li, Weibo Lin, and Fuda Ma. Weighting-based variable neighborhood search for optimal camera placement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12400–12408, 2021.

3 Qingyun Zhang, Zhipeng Lü, Zhouxing Su, and Chumin Li. A vertex weighting-based double-tabu search algorithm for the classical p-center problem. *Computers & Operations Research*, 160:106373, 2023.

4 Qingyun Zhang, Zhipeng Lü, Zhouxing Su, Chumin Li, Yuan Fang, and Fuda Ma. Vertex weighting-based tabu search for p-center problem. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 1481–1487, 2021.