PACE Challenge - Dominating Set

BASCI Onur, KEDADRY Yannis

***Abstract*—In this project, we explore various approaches for solving the NP-hard Dominating Set problem. Our investigation focuses on both exact and approximate solutions, leveraging two distinct strategies: one based on a reduction to SAT and the other transforming the problem into a Maximum Set Cover instance. While both of these auxiliary problems are also NP-hard, extensive research has been conducted to develop efficient techniques for computing exact solutions. We evaluate the effectiveness of these approaches and analyze their computational performance.**

I. Introduction

Computational complexity theory classifies problems based on their inherent difficulty, with NP-hard problems representing a category of decision problems that are at least as hard as the hardest problems in NP (nondeterministic polynomial time). These problems likely do not have efficient polynomial-time solutions, and solving them efficiently remains a fundamental challenge in theoretical computer science.

One well-known NP-hard problem is the **Dominating Set** problem. Given an undirected graph $G = (V, E)$, the Dominating Set problem asks whether there exists a subset $D \subseteq V$ such that every vertex in V is either in D or adjacent to at least one vertex in D. This problem has significant applications in network design, resource allocation, and biological network analysis.

The **Parameterized Algorithms and Computational Experiments (PACE) Challenge** is an annual competition designed to promote research in graph algorithms, particularly in parameterized complexity and exact exponential-time algorithms. This year, the challenge focuses on the Dominating Set problem. For this project we explored different algorithmic approaches both heuristic and exact for solving the Dominating Set problem, including reductions to the **SAT** and **Maximum Set Cover** problems.

II. Method

A. SAT reduction

The *Boolean Satisfiability Problem (SAT)* is a fundamental problem in computer science and logic. It asks whether there exists an assignment of truth values to variables that makes a given Boolean formula evaluate to true. SAT was the first problem proven to be *NP-complete*, meaning it is at least as hard as the hardest problems in the class NP (nondeterministic polynomial time). Because of this, many other computational problems can be transformed, or *reduced*, to SAT in polynomial time. SAT plays a central role in both theoretical computer science and practical applications, such as software verification, cryptography, and artificial intelligence.

The *Dominating Set* problem, another classic NP-complete problem, can be reduced to SAT by encoding the structure and constraints of a graph into a Boolean formula. Given a graph $G = (V, E)$ and an integer k, the goal is to determine whether there exists a set of at most k vertices such that every vertex is either in the set or adjacent to a vertex in the set. To reduce this to SAT, we create Boolean variables representing whether each vertex is included in the dominating set. Then, for each vertex, we construct a clause ensuring that either it or at least one of its neighbors is selected. Additional clauses enforce that no more than k variables are set to true. Solving the resulting SAT instance gives a solution to the original dominating set problem if one exists. We first approached solving the Dominating Set problem by using a SAT solver. For this, we used a solver called **PySAT**. However, this approach only allows us to answer the decision version of the problem: *does a dominating set of size k exist for a given graph?* To find the optimal (minimum) value of k, we must perform an additional binary search over possible values of k, which adds a complexity of $O(\log n)$ to the overall process. Due to this added overhead, we decided to pursue an alternative approach based on minimum set cover reductions.

B. Minimum Cover Set reduction

The **Minimum Set Cover** problem is a classical NP-complete problem in theoretical computer science and combinatorics. It is defined as follows: given a finite universe U and a collection $\mathcal{S}$ of subsets of U, the task is to find the smallest possible subcollection of $\mathcal{S}$ whose union is equal to U. In other words, we seek the fewest number of sets from $\mathcal{S}$ that together contain every element of the universe. Due to its NP-completeness, finding an exact solution is computationally difficult, and it is believed that no polynomial-time algorithm exists for all instances unless P = NP.

The **Dominating Set** problem can be reduced to the Minimum Set Cover problem using a simple transformation. Given a graph $G = (V, E)$, the goal in the Dominating Set problem is to find a minimum subset $D \subseteq V$ such that every vertex in the graph is either in D or adjacent to a vertex in D. To reduce this to a set cover instance, we treat each vertex $v \in V$ as both an element of the universe and a potential set in the collection. Each vertex v defines a set consisting of itself and all its neighbors — that is, the closed neighborhood $N[v]$. The universe is simply the set of all vertices V, and the task becomes selecting the smallest number of these neighborhood sets to cover all vertices

1) Heuristic solution: Given the NP-hard nature of the Dominating Set problem, we employed a greedy heuristic approach to obtain an approximate solution efficiently. Greedy algorithms provide a practical way to solve large instances by iteratively making locally optimal choices. The heuristic we implemented is based on a greedy approach to the equivalent

Minimum Set Cover formulation. The core idea is to iteratively select the set that maximizes coverage of uncovered vertices and add its elements to the cover set. This process continues until all vertices are included. A critical aspect of the greedy approach is the cost function used to determine which set to select at each step. We experimented with different cost functions:

- **Size of sets**: Prioritizing sets with the largest number of elements.
- **Frequency of elements**: Selecting sets based on the frequency of their elements in other sets.
- **Number of newly covered elements**: Choosing the set that adds the maximum number of new elements to the cover.

After multiple testing, we found that using the number of newly covered elements as the cost function produced the best results. This ensures that each selected vertex maximally contributes to covering the remaining uncovered nodes.

2) Exact solution: The trivial set cover algorithm is a basic recursive strategy that explores all possible combinations of sets to find a minimum set cover. At each step, it selects a set S of maximum cardinality from the collection and branches into two subproblems: one where S is included in the cover, and one where it is not. If S is included, the algorithm removes all elements covered by S from the universe and removes from the collection any sets that now contain only already-covered elements. If S is excluded, the algorithm simply removes it from the collection. These two subproblems are solved recursively, and the smallest valid set cover found is returned. The process continues until no sets are left to choose from, at which point the algorithm checks whether the remaining universe is empty—indicating a valid cover.

Algorithm 1 A trivial set cover algorithm

Require: A set cover instance $(\mathcal{S},\mathcal{U})$
Ensure: A minimum set cover of $(\mathcal{S},\mathcal{U})$
function MSC($\mathcal{S},\mathcal{U}$)
  if $\mathcal{S} = \emptyset$ **then**
    if $\mathcal{U} = \emptyset$ **then**
      return $\emptyset$
    else
      return No cover
    end if
  end if
  Let $S \in \mathcal{S}$ be a set of maximum cardinality
  $\mathcal{C}_1 \leftarrow \{S\} \cup \text{MSC}(\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\}, \mathcal{U} \setminus S)$
  $\mathcal{C}_2 \leftarrow \text{MSC}(\mathcal{S} \setminus \{S\}, \mathcal{U})$
  return the smallest set cover from $\mathcal{C}_1$ and $\mathcal{C}_2$, or **No cover** if none exist
end function

Our implementation followed the ideas presented in **[RB11]**. Their idea is to reduced the time complexity of Algorithm **??** by adding different stopping condition for the branch and bounds approach. In the following we present these different reduction rules presented in the original paper.

Reduction Rule 1 improves the basic set cover algorithm by eliminating unnecessary branching when dealing with unique elements in the universe. Specifically, if an element $e \in \mathcal{U}$ appears in only one set $R \in \mathcal{S}$, then R must be included in any valid set cover, since no other set can cover e. Instead of considering both the inclusion and exclusion of R as in the trivial algorithm, this rule directly includes R in the set cover and removes e along with all elements of R from the universe. It also removes or updates any affected sets accordingly. This significantly reduces the search space and eliminates redundant computations, as any solution must contain R. The application of this rule leads to the improved version of the algorithm.

Algorithm 2 An improved version of Algorithm 1 by adding Reduction Rule 1

Require: A set cover instance $(\mathcal{S},\mathcal{U})$
Ensure: A minimum set cover of $(\mathcal{S},\mathcal{U})$
function MSC($\mathcal{S},\mathcal{U}$)
  if $\mathcal{S} = \emptyset$ **then**
    return $\emptyset$
  end if
  Let $S \in \mathcal{S}$ be a set of maximum cardinality
  if there exists an element $e \in \mathcal{U}$ of frequency one **then**
    Let $R \in \mathcal{S}$ be the set containing e
    return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}, \mathcal{U} \setminus R)$
  end if
  $\mathcal{C}_1 \leftarrow \{S\} \cup \text{MSC}(\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\}, \mathcal{U} \setminus S)$
  $\mathcal{C}_2 \leftarrow \text{MSC}(\mathcal{S} \setminus \{S\}, \mathcal{U})$
  return the smallest cover among $\mathcal{C}_1$ and $\mathcal{C}_2$
end function

Reduction Rule 2 targets a simple but important special case in the Set Cover problem: when all remaining sets have cardinality one. In this situation, each remaining element in the universe is covered by exactly one singleton set, and there is no further need to branch. Since no two elements share a covering set, the optimal solution is to take one singleton set per uncovered element. Thus, if the current set of maximum cardinality has size one (i.e., $|S| \leq 1$), the algorithm can immediately return a set cover consisting of singleton sets, one for each element in the universe. This rule eliminates unnecessary recursive calls and significantly reduces the computational effort in such trivial subcases.

Reduction Rule 2:
Let $S \in \mathcal{S}$ be a set of maximum cardinality
if $|S| \leq 1$ **then**
  return $\{\{e\} \mid e \in \mathcal{U}\}$
end if

Reduction Rule 3 eliminates redundant sets by leveraging subset relationships. If a set R is entirely contained within another set Q (i.e., $R \subseteq Q$), then R is redundant. Including Q in the cover ensures all elements of R are already covered, making R unnecessary. Any valid set cover containing R can be replaced by a smaller or equally sized cover by substituting R with Q or removing R entirely. This rule simplifies the problem by pruning the search space, reducing the number of sets to consider. Combined with Reduction Rule 1, it also removes all singleton sets: if a singleton set's element has

frequency one, Rule 1 applies; otherwise, the singleton is a subset of another set and is removed by Rule 3.

Reduction Rule 3:
if there exist $Q, R \in \mathcal{S}$ such that $R \subseteq Q$ **then**
  Remove R from $\mathcal{S}$
  return MSC($\mathcal{S} \setminus \{R\}$,$\mathcal{U}$)
end if

Reduction Rule 4 applies to the special case where all remaining sets have cardinality at most two. In this case, the minimum set cover problem can be efficiently reduced to a graph-theoretic problem involving maximum matching. Specifically, we construct a graph where each element in the universe is a vertex, and each set of size two corresponds to an edge between its two elements. A maximum matching in this graph gives the largest collection of disjoint sets that can be included in the cover without overlaps. Once the matching is found, the corresponding sets are added to the cover, and for any unmatched vertex, an additional set covering it is selected. Since maximum matching can be computed in polynomial time, this reduction rule provides an efficient way to solve the set cover problem in this restricted case, avoiding unnecessary branching and recursion.

Algorithm 3 Reduction Rule 4

Require: A set cover instance $(\mathcal{S},\mathcal{U})$
Ensure: A minimum set cover of $(\mathcal{S},\mathcal{U})$
function MSC($\mathcal{S},\mathcal{U}$)
  if $\mathcal{S} = \emptyset$ **then**
    return $\emptyset$
  end if
  Let $S \in \mathcal{S}$ be a set of maximum cardinality
  if there exists an element $e \in \mathcal{U}$ of frequency one **then**
    Let $R \in \mathcal{S}$ be the set such that $e \in R$
    return $\{R\} \cup$ MSC($\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}$,$\mathcal{U} \setminus R$)
  else if there exist sets $Q, R \in \mathcal{S}$ such that $R \subseteq Q$ **then**
    return MSC($\mathcal{S} \setminus \{R\}$,$\mathcal{U}$)
  else if $|S| \leq 2$ **then**
    return a minimum set cover computed in polynomial time by using maximum matching
  else
    $\mathcal{C}_1 \leftarrow \{S\} \cup$ MSC($\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\}$,$\mathcal{U} \setminus S$)
    $\mathcal{C}_2 \leftarrow$ MSC($\mathcal{S} \setminus \{S\}$,$\mathcal{U}$)
    return the smallest cover from $\mathcal{C}_1$ and $\mathcal{C}_2$
  end if
end function

Reduction Rule 5 introduces the concept of *element subsumption* to further simplify set cover instances. We say an element e_1 is subsumed by e_2 if every set containing e_1 also contains e_2 (i.e., $\mathcal{S}(e_1) \subseteq \mathcal{S}(e_2)$). In such cases, any cover that includes e_1 will automatically cover e_2 as well. Therefore, we can safely remove e_2 from all sets and the universe without affecting the solution quality. This rule is particularly effective when dealing with multiple frequency-two elements whose occurrences are correlated across sets.

Reduction Rule 5:
if there exist elements $e_1, e_2 \in \mathcal{U}$ such that $\mathcal{S}(e_1) \subseteq \mathcal{S}(e_2)$ **then**
  return MSC($\{R \setminus \{e_2\} \mid R \in \mathcal{S}\}$,$\mathcal{U} \setminus \{e_2\}$)
end if

Reduction Rule 6 is based on a refined counting argument applied in the case where all sets have cardinality at most two. It considers a set R that contains elements of frequency two, each of which also appears in exactly one other set of size two. The rule examines how many new elements—denoted as q—can be covered by these alternative sets that contain the frequency-two elements from R, but are not part of R itself. If the number of such additional elements q is strictly less than the number of frequency-two elements in R, then including R in the set cover is guaranteed to be at least as good as, and often better than, discarding it. This is because taking R allows us to cover its elements using a single set, while discarding it would require using at least r_2 sets to cover the same elements, along with any others. Hence, in this scenario, we can safely include R in the cover without branching, which improves the efficiency of the algorithm by reducing unnecessary exploration of suboptimal branches.

Reduction Rule 6:
for all sets $R \in \mathcal{S}$ **do**
  Let $r_2 = \{e \in R \mid f(e) = 2\}$
  Let $Q = \bigcup_{e \in r_2} Q_e$, where Q_e is the other set containing e
  if $|Q \setminus R| < |r_2|$ **then**
    return $\{R\} \cup$ MSC($\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}$, $\mathcal{U} \setminus R$)
  end if
end for

Reduction Rule 7 provides an optimized handling of sets containing two frequency-two elements, which represent a common bottleneck in the branching process. When encountering a 2-element set $R = \{e_1, e_2\}$ where both elements have frequency two, the rule performs a specialized transformation: it replaces R and its companion sets R_1, R_2 (containing e_1 and e_2 respectively) with a new composite set $Q = (R_1 \cup R_2) \setminus R$. The rule then recursively solves the transformed instance and adjusts the solution based on whether Q was selected in the cover.

Reduction Rule 7:
if there exists a set $R \in \mathcal{S}$ of cardinality two $R = \{e_1, e_2\}$ with $f(e_1) = f(e_2) = 2$ **then**
  Let $\mathcal{S}(e_i) = \{R, R_i\}$ for $i \in \{1, 2\}$
  $Q = (R_1 \cup R_2) \setminus R$
  $\mathcal{C} =$ MSC($(\mathcal{S} \setminus \{R, R_1, R_2\}) \cup \{Q\}$,$\mathcal{U} \setminus R$)
  if $Q \in \mathcal{C}$ **then**
    return $(\mathcal{C} \setminus \{Q\}) \cup \{R_1, R_2\}$
  else
    return $\mathcal{C} \cup \{R\}$
  end if
end if

Overall, the complete algorithm is presented in Algorithm **??**.

III. Results

To compare the efficiency of the reduction rules, we tested them on a series of graphs with increasing complexity. Specif-

Algorithm 4 Final Algorithm for Set Cover Modelling of Dominating Set

Require: A set cover instance $(\mathcal{S},\mathcal{U})$
Ensure: A minimum set cover of $(\mathcal{S},\mathcal{U})$

function MSC($\mathcal{S},\mathcal{U}$)
  if $\mathcal{S} = \emptyset$ **then**
    return $\emptyset$
  end if
  Let $S \in \mathcal{S}$ be a set of maximum cardinality
  if $\exists e \in \mathcal{U}$ of frequency one **then**
    Let R be the set containing e
    return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}, \mathcal{U} \setminus R)$
  else if $\exists Q, R \in \mathcal{S}$ such that $R \subseteq Q$ **then**
    return $\text{MSC}(\mathcal{S} \setminus \{R\}, \mathcal{U})$
  else if $\exists e_1, e_2 \in \mathcal{U}$ such that $\mathcal{S}(e_1) \subseteq \mathcal{S}(e_2)$ **then**
    return $\text{MSC}(\{R \setminus \{e_2\} \mid R \in \mathcal{S}\}, \mathcal{U} \setminus \{e_2\})$
  else if $\exists R \in \mathcal{S}$ with $\left|\bigcup_{e \in R, f(e)=2, Q \in \mathcal{S}(e)} Q \setminus R\right| < |\{e \in R \mid f(e) = 2\}|$ **then**
    return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}, \mathcal{U} \setminus R)$
  else if $\exists R \in \mathcal{S}$ with $|R| = 2$ and $f(e_1) = f(e_2) = 2$ for $R = \{e_1, e_2\}$ **then**
    Let $\mathcal{S}(e_i) = \{R, R_i\}$ for $i = 1, 2$
    $Q \leftarrow (R_1 \cup R_2) \setminus R$
    $\mathcal{C} \leftarrow \text{MSC}((\mathcal{S} \setminus \{R, R_1, R_2\}) \cup \{Q\}, \mathcal{U} \setminus R)$
    if $Q \in \mathcal{C}$ **then**
      return $(\mathcal{C} \setminus \{Q\}) \cup \{R_1, R_2\}$
    else
      return $\mathcal{C} \cup \{R\}$
    end if
  else if $|S| \leq 2$ **then**
    return minimum set cover computed via maximum matching
  else
    $\mathcal{C}_1 \leftarrow \{S\} \cup \text{MSC}(\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\}, \mathcal{U} \setminus S)$
    $\mathcal{C}_2 \leftarrow \text{MSC}(\mathcal{S} \setminus \{S\}, \mathcal{U})$
    return the smallest cover between $\mathcal{C}_1$ and $\mathcal{C}_2$
  end if
end function

ically, we generated multiple Erdős–Rényi graphs. An **Erdős-Rényi graph** is a type of random graph where each possible edge between a pair of n vertices is included independently with a fixed probability p. In our experiment, we generated 16 graphs with an increasing number of vertices, starting from 5 up to 80, using a fixed edge probability of $p = 0.5$. For each graph, we computed the execution time required to find the minimum set cover. This procedure was repeated for 8 different solvers: the first being the trivial solver, and the others incorporating Reduction Rules 1 through 7, respectively. Finally, we plotted the execution times for each solver to visualize and compare their performance.

We observe a clear improvement compared to the trivial algorithm, even with the application of Reduction Rule 1. The results continue to improve up to Rule 4; however, beyond Rule 4, we do not see a significant or consistent improvement. This may be due to limitations in our implementation. Some

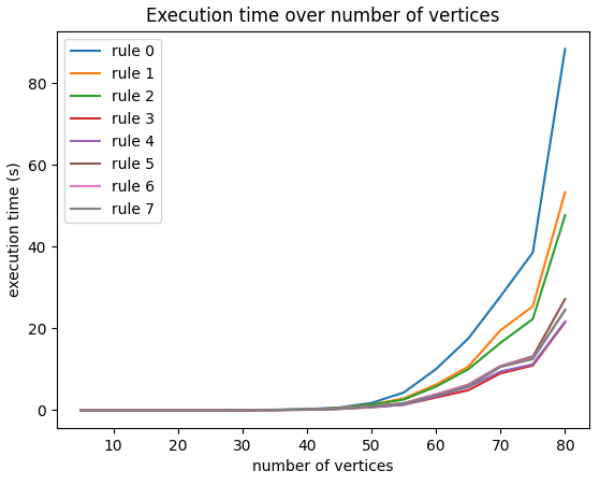

Fig. 1: Execution time over solver execution

reduction rules, such as Rule 6, involve higher computational overhead. Although they can be computed in polynomial time, they may still be costly in practice, and their benefits might only become noticeable on more complex or larger graphs.

IV. Conclusion

The Dominating Set problem, a core NP-hard challenge in graph theory, was tackled through SAT reduction and transformation into the Minimum Set Cover problem. While the SAT approach provided a theoretical foundation, its inefficiency on large graphs led us to focus on Set Cover.

Our optimized Set Cover solver, enhanced with reduction rules, significantly outperformed naive branching. Rules 1–4 notably improved execution times, while the limited gains from Rules 5–7 highlight the trade-off between complexity and efficiency. The greedy heuristic, prioritizing newly covered elements, proved effective for large instances.

Future work could refine reduction rules, explore parallelization, or combine exact and heuristic methods. Testing on structured graphs may reveal domain-specific benefits. This study underscores the power of parameterized algorithms in balancing optimality with computational feasibility.

References

[RB11] J. M. M. van Rooij and H. L. Bodlaender. “Exact Algorithms for Dominating Set”. In: *Discrete Applied Mathematics* 159 (2011), pp. 2147–2164. DOI: 10.1016/j.dam.2011.07.001. URL: https://doi.org/10.1016/j.dam.2011.07.001.

[JZ23] Hua Jiang and Zhifei Zheng. “An Exact Algorithm for the Minimum Dominating Set Problem”. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*. Ed. by Edith Elkind. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2023, pp. 5604–5612. DOI: 10.24963/ijcai.2023/622. URL: https://doi.org/10.24963/ijcai.2023/622.