Report on
CSE 318 Assignment-02
Solving the Max-cut problem by GRASP

Abdullah Al Mahmud
Student ID : 2105120

May 2025

1 Introduction

This report presents a comparative analysis of several heuristic algorithms applied to the Maximum Cut problem on a set of benchmark graph instances. The algorithms under consideration include Randomized Algorithm, Greedy Algorithm, Semi-Greedy Algorithm, Local Search, and GRASP (Greedy Randomized Adaptive Search Procedure). Each algorithm has been implemented and evaluated across multiple graph inputs to assess its performance.
The goal of this study is to understand the strengths and limitations of these approaches, both individually and in combination. By analyzing their outcomes and visualizing their performance trends, we aim to identify which methods yield near-optimal solutions consistently and under what conditions they excel. The results, along with CSV summaries and plots, provide insights into their comparative effectiveness on diverse graph instances.

2 High level description of the algorithms

2.1 Randomized Algorithm

This algorithm randomly assigns each vertex to one of two partitions, building a potential cut without considering the graph's structure. Each vertex has an equal chance of being placed in either partition, ensuring an unbiased split. To account for the variability of this random process, the algorithm is executed n times, and the average cut value is reported. While this method is computationally cheap, it serves as a baseline and often yields lower-quality solutions compared to more strategic approaches.

2.2 Greedy Algorithm

The greedy algorithm begins by initializing the cut with the two endpoints of the edge with the maximum weight, placing each in separate partitions X and Y. It then proceeds iteratively over the remaining $|V| - 2$ unassigned vertices. At each step, the algorithm evaluates the potential contribution of each unassigned vertex to the current partial cut and assigns it to the partition where its inclusion yields the greatest increase in cut weight. This locally optimal decision-making is repeated until all vertices are assigned. While efficient, the greedy strategy may converge to suboptimal solutions due to its myopic nature.

2.3 Semi-Greedy Algorithm

The semi-greedy algorithm enhances the traditional greedy approach by introducing controlled randomness to avoid poor local optima. It begins similarly by selecting the edge with the maximum weight and placing its endpoints into two empty partitions X and Y. Then, instead of deterministically choosing the vertex with the highest contribution to the cut, it constructs a Restricted Candidate List (RCL) containing vertices whose contributions are within a threshold defined by a parameter $\alpha \in [0, 1]$. A vertex is selected uniformly at random from this list and added to the appropriate partition. This blend of greediness and randomness promotes diversity in the search space and increases the chance of finding better solutions compared to purely greedy methods.

2.4 Local search

The local search algorithm seeks to iteratively improve an existing solution by exploring its immediate neighborhood. Starting from an initial partition—typically generated by a constructive heuristic like the semi-greedy algorithm—it repeatedly evaluates the effect of moving individual vertices from one partition to the other. If a move results in a higher cut value, it is accepted, and the solution is updated accordingly. This process continues until no single vertex move can further improve the cut, indicating a local optimum. Although local search does not guarantee a globally optimal solution, it effectively enhances initial solutions.

2.5 GRASP

GRASP is a randomized multistart iterative method used to compute high-quality solutions for combinatorial optimization problems. Each iteration of GRASP consists of two main phases: the construction phase and the local search phase. In the construction phase, a feasible solution is built using a semi-greedy or greedy algorithm. In the local search phase, starting from the constructed solution, iterative improvement is applied until a locally optimal solution is found. This process is repeated multiple times, and the best solution obtained across all iterations is selected.

3 Comparison of algorithms

Based on the experimental results across the benchmark instances, the GRASP algorithm generally provides the best solutions. This is due to its two-phase approach: it first constructs a reasonably good solution using randomness and greediness, and then improves it through local search. As a result, it can explore a wide range of solutions and refine them effectively.
The Local Search algorithm also produces strong results, especially when it starts from a good initial solution. It improves an existing solution by making small changes, leading to a better cut value in many cases.
The Semi-Greedy algorithm performs better than the basic greedy approach by introducing controlled randomness. This helps avoid some poor decisions that a purely greedy method might make.
The Greedy algorithm, although fast and simple, often gets stuck in local optima because it makes decisions based only on immediate benefit without considering long-term impact.
The Randomized algorithm gives the weakest results overall, as it assigns vertices completely at random. However, it serves as a useful baseline for comparison with more advanced heuristics.

4 Plots

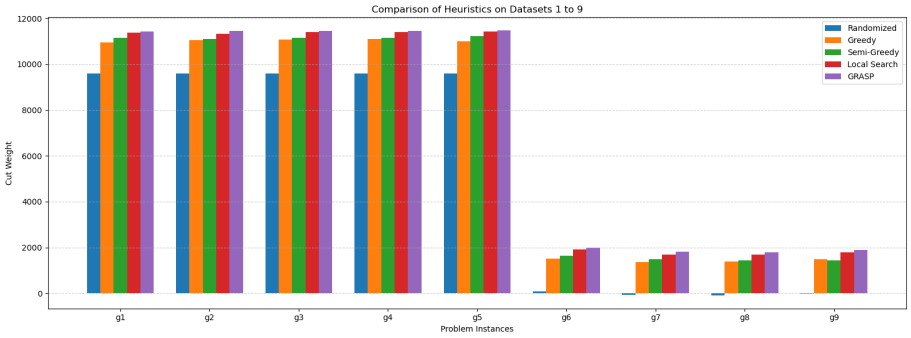

Figure 1: Comparison : Graph 1 - 9

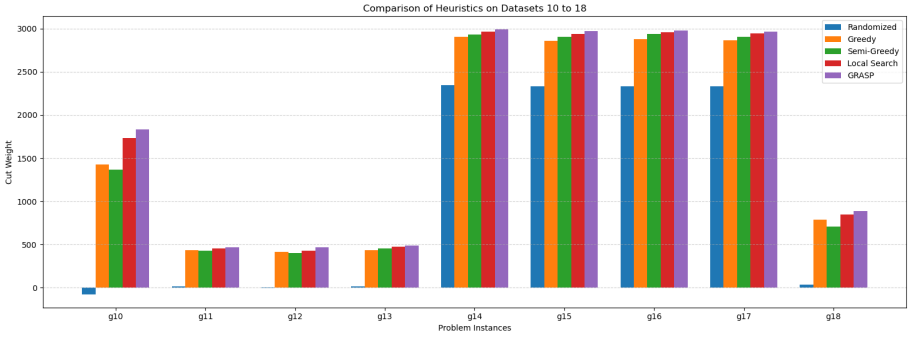

Figure 2: Comparison : Graph 10 - 18

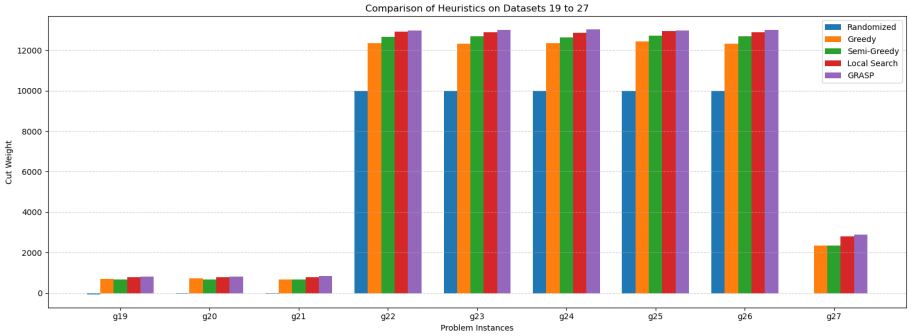

Figure 3: Comparison : Graph 19 - 27

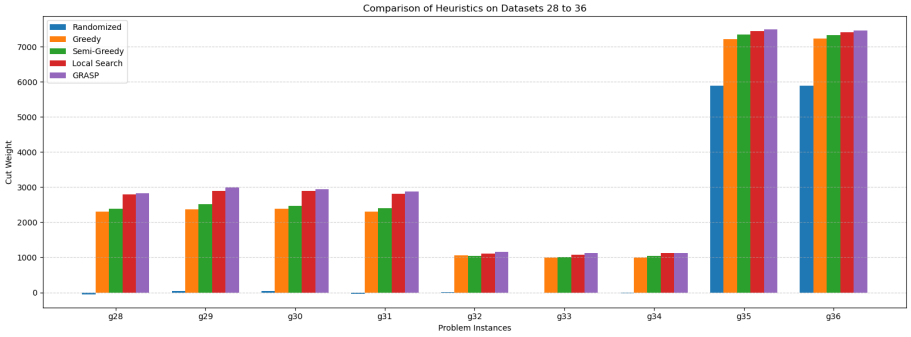

Figure 4: Comparison : Graph 28 - 36

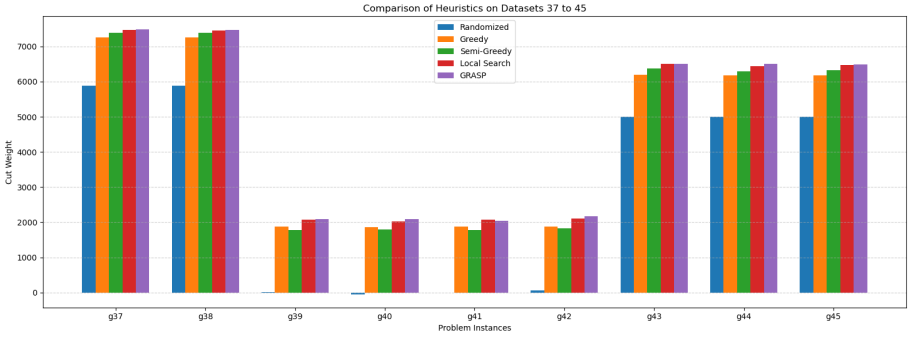

Figure 5: Comparison : Graph 37 - 45

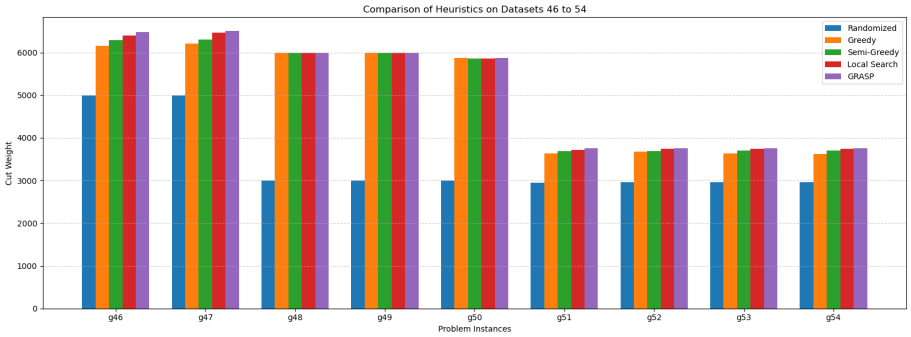

Figure 6: Comparison : Graph 46 - 54

5 Conclusion

This report presents the implementation and evaluation of several heuristic algorithms for the Maximum Cut problem, including Randomized, Greedy, Semi-Greedy, Local Search, and GRASP. Through both qualitative descriptions and quantitative experiments on benchmark datasets, we have observed varying levels of performance across the algorithms. The results clearly indicate that GRASP, with its iterative refinement and adaptive construction strategy, consistently achieves superior cut values. Local Search and Semi-Greedy approaches also perform well, striking a balance between solution quality and computational effort. Simpler methods like Randomized and Greedy serve as useful baselines but are generally outperformed by more sophisticated techniques. Overall, this study highlights the effectiveness of combining construction heuristics with iterative improvement for tackling combinatorial optimization problems like Max-Cut.