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

Nakib Arman
Student ID: 2105128

May 12, 2025

1 Introduction

This assignment was based on solving the Maximum Cut (Max-Cut) problem using the GRASP algorithm. The Max-Cut problem involves finding a partition of vertices in an undirected graph $G = (V, E)$ with edge weights w_{uv} such that the total weight of edges crossing the partitions is maximized. We implemented five algorithms: Randomized, Greedy, Semi-Greedy, Local Search, and GRASP—and evaluated their performance on 54 benchmark graphs provided in the specification.

2 Algorithms

2.1 Randomized Heuristic

The Randomized heuristic builds a solution by assigning each vertex randomly to one of two partitions, X or Y , with equal probability (0.5). This process is fast and simple but doesn't guarantee a good solution. So it is repeated $n = 50$ times, and the average cut weight is computed for a better result.

2.2 Greedy Heuristic

The Greedy heuristic solves the problem with a deterministic approach and tries to maximize the cut in each step. The two vertices of the highest valued edge are assigned into two different partitions X and Y.Then for each of the remaining vertices

it checks the the potential cut increase if placed in X or Y and assigns the vertex to the partition that maximizes the cut.

2.3 Semi-Greedy Heuristic

The Semi-Greedy heuristic extends the Greedy approach by introducing randomness. For each unassigned vertex v a Restricted Candidate List (RCL) is built using a value-based method with $\mu = w_{\min} + \alpha(w_{\max} - w_{\min})$, and vertices with greedy value $\geq \mu$ are included. A vertex is randomly selected from the RCL and assigned to the partition that maximizes the cut.

2.4 Local Search

The Local Search algorithm starts at a constructed solution and applies iterative improvement until a locally optimal solution is found. For each vertex v, it computes the change in cut value $\delta(v)$ if v is moved to the other partition ($\sigma_{\text{current}}(v) - \sigma_{\text{other}}(v)$). The vertex with the largest positive $\delta(v)$ is moved, and this process repeats until no improving move exists, indicating a local optimum.

2.5 GRASP

GRASP (Greedy Randomized Adaptive Search Procedure) is a randomized multi-start iterative method for computing good quality solutions of combinatorial optimization problems. For a given number of iterations, it constructs a solution using the Semi-Greedy heuristic, applies Local Search to reach a local optimum, and keeps the best solution found across iterations. GRASP balances exploration (via randomness in Semi-Greedy) and exploitation (via Local Search) to find high-quality solutions.

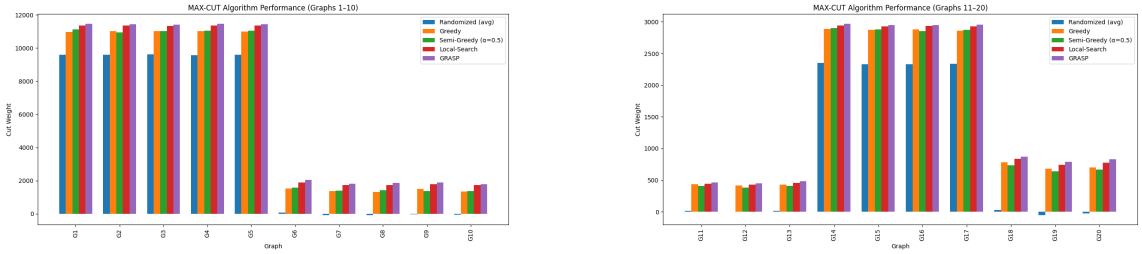
3 Comparison of Algorithms

The performance of the algorithms was evaluated on 54 benchmark graphs, with results summarized in `2105128.csv`.
Analysis:

- **Randomized** consistently underperforms as it lacks any greedy optimization.
- **Greedy** improves over Randomized by introducing greedy approach, due to its focus on maximizing the cut at each step.

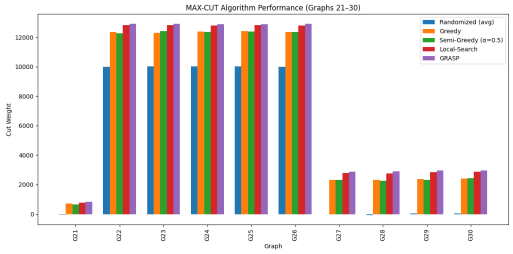
- **Semi-Greedy** performs slightly better than Greedy (e.g., 10956 for greedy vs. 11123 for semi-greedy for input g1), as the RCL introduces randomness that helps escape poor local decisions.
- **Local Search**, applied after Semi-Greedy in GRASP, further improves the cut by 1–2% (e.g., 11360 for input g1), as it refines the initial solution.
- **GRASP** achieves the highest cut values, such as 11472 for input g1 , outperforming others by 5–10%. However, it still falls short of the known best solutions (e.g., 12078 for input g1), indicating room for improvement in the heuristic design or parameter tuning (α, iterations).

4 Graph

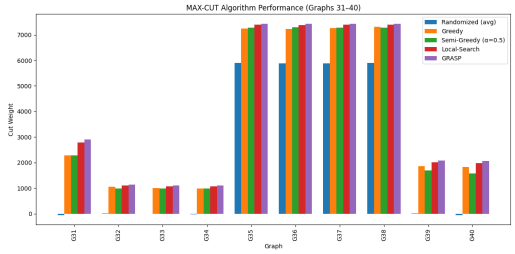


(a) Input: g1 (b) Input: g2

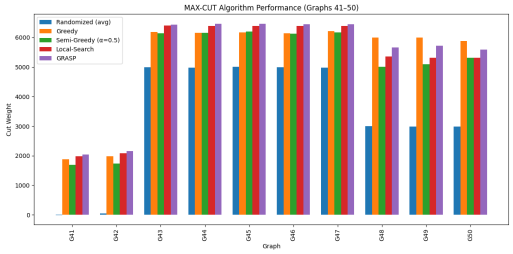
Figure 1: Performance comparison of the five algorithms on inputs g1 and g2.



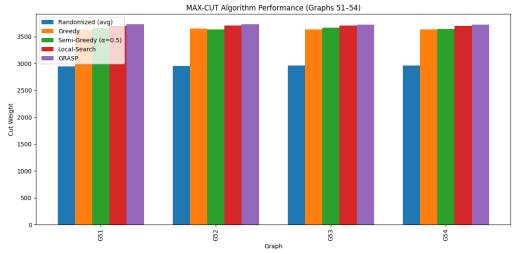
(c) Input: g3



(d) Input: g4



(e) Input: g5



(f) Input: g6

Figure 2: Performance comparison of the five algorithms on inputs g3 to g6.

5 Conclusion

This assignment implemented five algorithms to solve the Max-Cut problem on 54 benchmark graphs. GRASP proved to be the most effective, consistently achieving the highest cut values, though it did not reach the known best solutions in most cases. The visualizations highlight GRASP's superiority, while also showing the incremental improvements from Greedy to Semi-Greedy to Local Search. Challenges included tuning the α parameter for Semi-Greedy and handling large graphs efficiently. Future work could explore adaptive α values or more iterations to close the gap to the known best solutions.