Solving the Max-Cut Problem Using Heuristic Algorithms

Student ID: 2105009

May 12, 2025

1 Introduction

Given an undirected weighted graph $G = (V, E)$, the MAX-CUT problem involves finding a partition of vertices into two disjoint subsets such that the sum of weights of edges crossing between the partitions is maximized.

Formally, for a partition $(S, \bar{S})$ where $S \subset V$ and $\bar{S} = V \setminus S$, the objective is to maximize:

$$w(S, \bar{S}) = \sum_{u \in S, v \in \bar{S}} w_{uv} \tag{1}$$

Since MAX-CUT is NP-hard, exact solutions are computationally prohibitive for large instances. This report explores and compares several heuristic approaches for solving this problem, with a focus on the GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic.

2 Algorithm Descriptions

2.1 Randomized Algorithm

The randomized algorithm represents the simplest approach to the MAX-CUT problem:

- Each vertex is assigned to either partition X or Y with equal probability (0.5)
- No optimization is performed, making this method extremely fast but typically yielding poor-quality solutions
- Multiple trials are conducted, and the average cut weight is reported

This algorithm serves as a baseline for evaluating more sophisticated approaches.

2.2 Greedy Algorithm

The greedy algorithm follows a deterministic construction approach:

- Initially, the edge with maximum weight is identified, and its endpoints are placed in opposite partitions
- Remaining vertices are considered sequentially, with each vertex placed in the partition that maximizes the immediate increase in cut weight
- The process continues until all vertices are assigned

While this method is computationally efficient, it often gets trapped in local optima due to its myopic decision strategy.

2.3 Semi-Greedy Algorithm

The semi-greedy algorithm introduces controlled randomness to the greedy approach:

- Instead of always selecting the best candidate vertex, it constructs a Restricted Candidate List (RCL) containing promising vertices
- The RCL includes vertices whose greedy function value satisfies:

$$\text{value} \geq w_{min} + \alpha \cdot (w_{max} - w_{min})$$

- A vertex is randomly selected from the RCL, balancing exploitation and exploration
- The parameter $\alpha \in [0, 1]$ controls the trade-off between greediness and randomness

This approach aims to overcome the limitations of purely greedy construction by introducing diversity in the solution space.

2.4 Local Search

Local search is an iterative improvement technique:

- Starting from an initial solution (partition), it systematically evaluates vertex moves from one partition to the other
- For each vertex v, the gain in cut weight $\delta(v)$ is calculated using:

$$\delta(v) = \begin{cases} \sigma_S(v) - \sigma_{\bar{S}}(v), & \text{if } v \in S \\ \sigma_{\bar{S}}(v) - \sigma_S(v), & \text{if } v \in \bar{S} \end{cases}$$

- The best improving move is executed, and the process continues until no further improvement is possible

Local search is valuable for refining solutions generated by construction heuristics.

2.5 GRASP

GRASP (Greedy Randomized Adaptive Search Procedure) is a multi-start metaheuristic combining construction and improvement phases:

- The construction phase uses the semi-greedy approach to generate diverse initial solutions
- The improvement phase applies local search to refine each constructed solution
- The process is repeated for multiple iterations, and the best overall solution is retained

The pseudo-code for GRASP is as follows:

Algorithm 1 GRASP for MAX-CUT

```
1:  procedure GRASP($G, MaxIterations$)
2:      $best\_solution \leftarrow null$
3:      $best\_cut\_weight \leftarrow 0$
4:      for $i = 1$ to $MaxIterations$ do
5:          $solution \leftarrow$ SemiGreedyConstruction($G$)
6:          $solution \leftarrow$ LocalSearch($solution$)
7:          if $w(solution) > best\_cut\_weight$ then
8:              $best\_solution \leftarrow solution$
9:              $best\_cut\_weight \leftarrow w(solution)$
10:         end if
11:     end for
12:     return $best\_solution$
13: end procedure
```

GRASP combines the strengths of both semi-greedy construction and local search, making it a powerful approach for solving the MAX-CUT problem.

3 Experimental Results

All five algorithms were implemented and evaluated on 54 benchmark graphs from the literature. The experiments were conducted with the following parameters:

- Randomized algorithm: 1000 trials per graph
- Semi-greedy algorithm: $\alpha = 0.8$
- Local Search algorithm: $n\_iter = 5$
- GRASP: $max\_iter = 50$, with $\alpha = 0.8$ for the semi-greedy construction phase

3.1 Performance Comparison

The figures below present the cut weights achieved by each algorithm across different graph instances, alongside known best solutions or upper bounds where available.

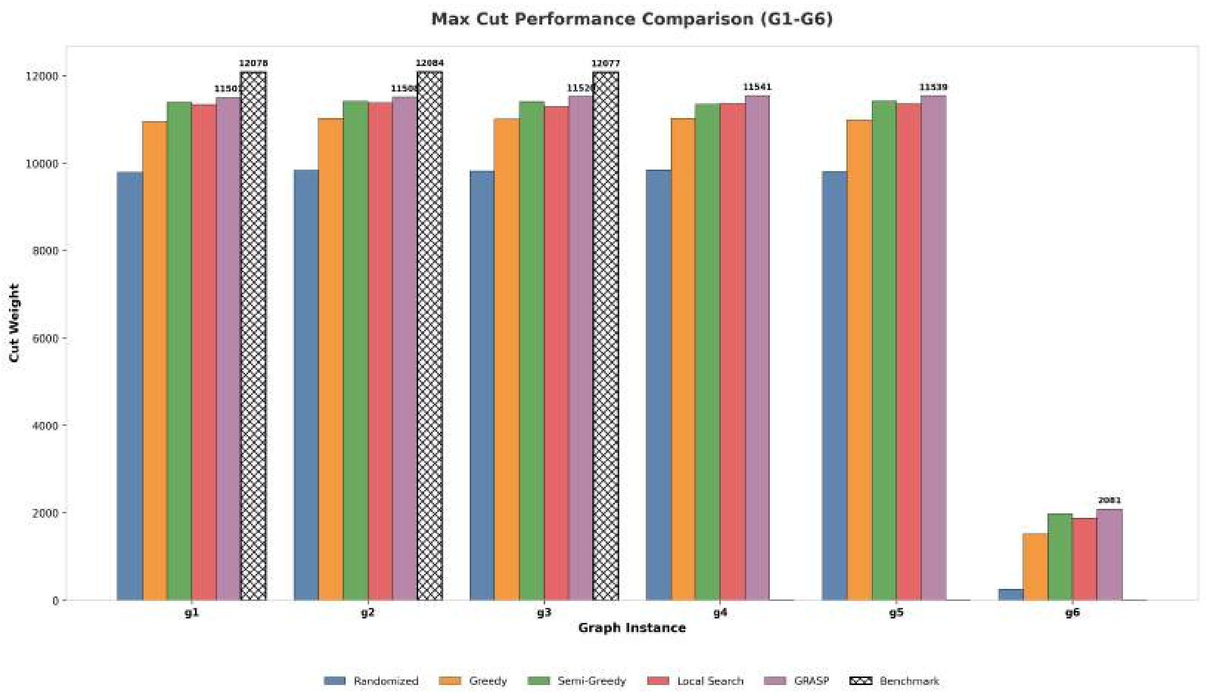

Figure 1: Max Cut Performance Comparison (G1-G6)

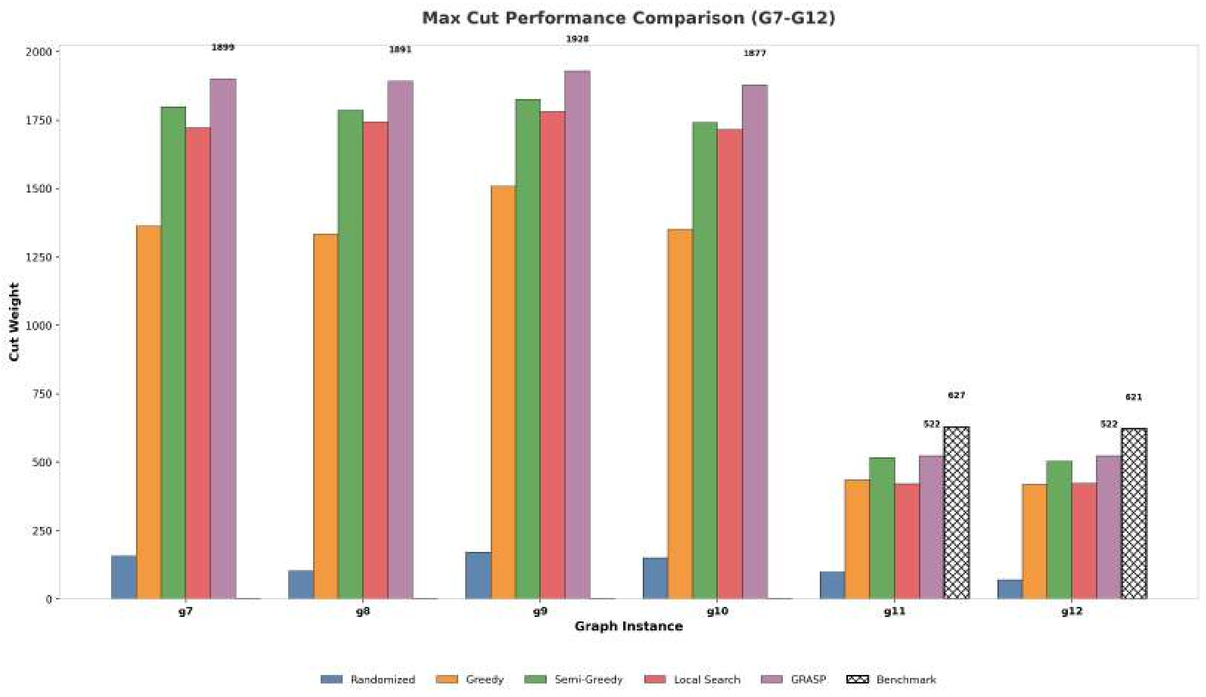

Figure 2: Max Cut Performance Comparison (G7-G12)

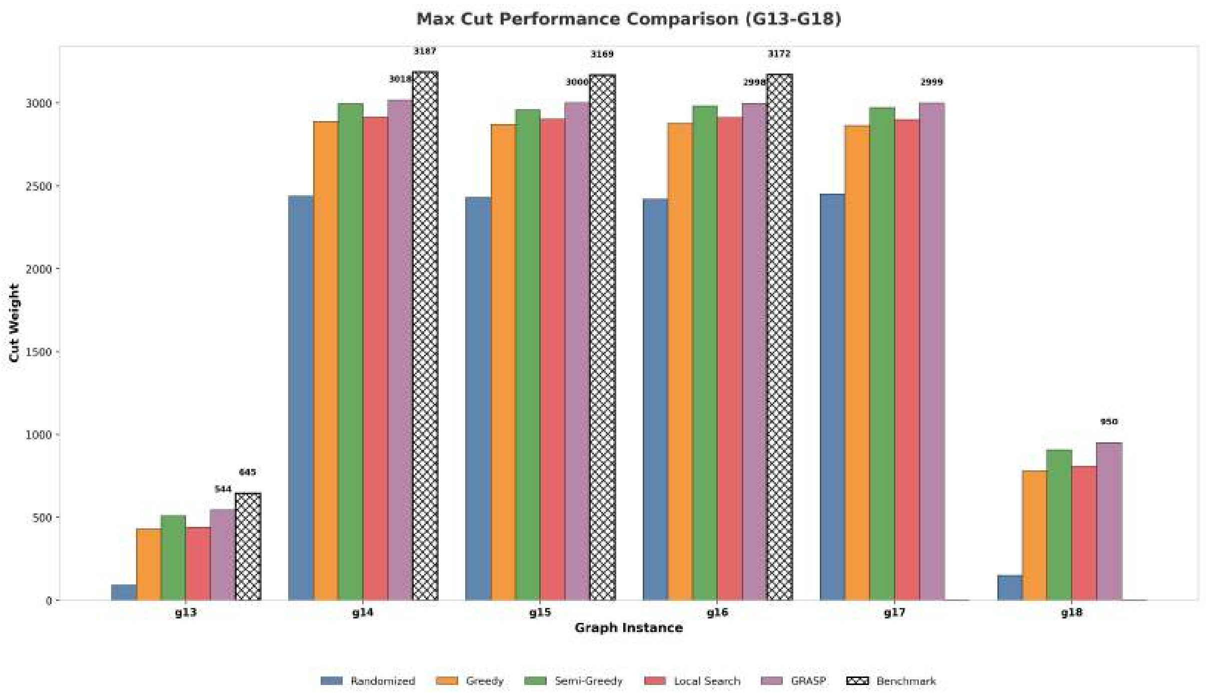

Figure 3: Max Cut Performance Comparison (G13-G18)

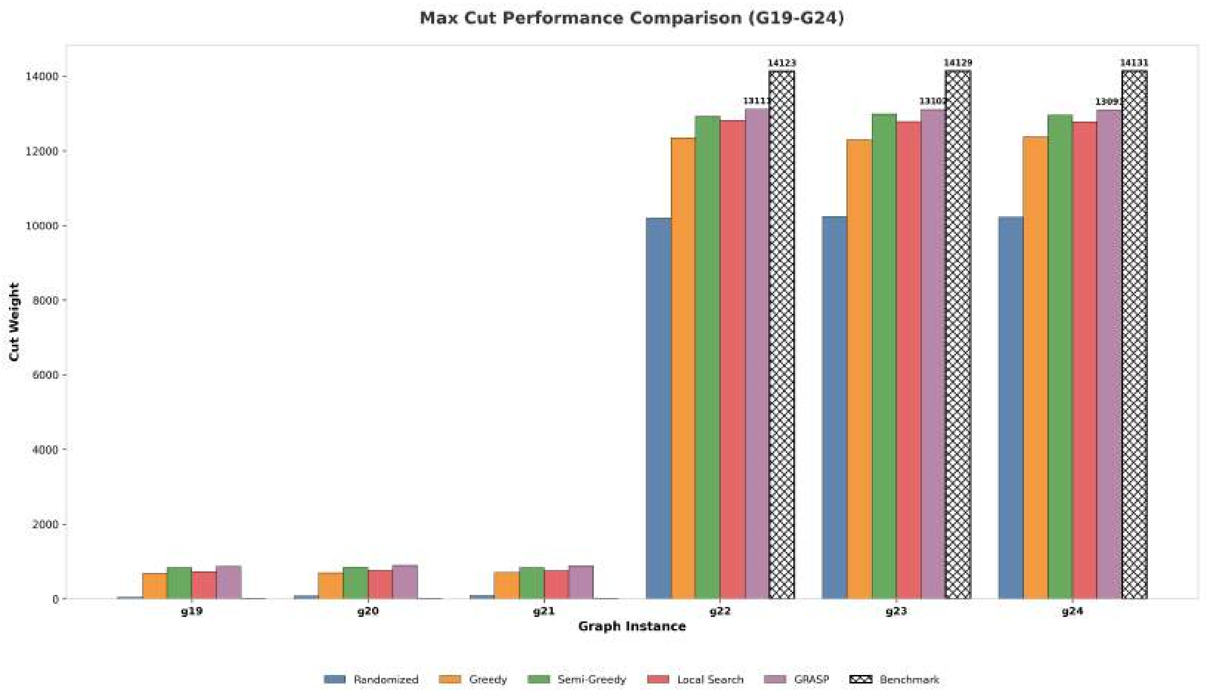

Figure 4: Max Cut Performance Comparison (G19-G24)

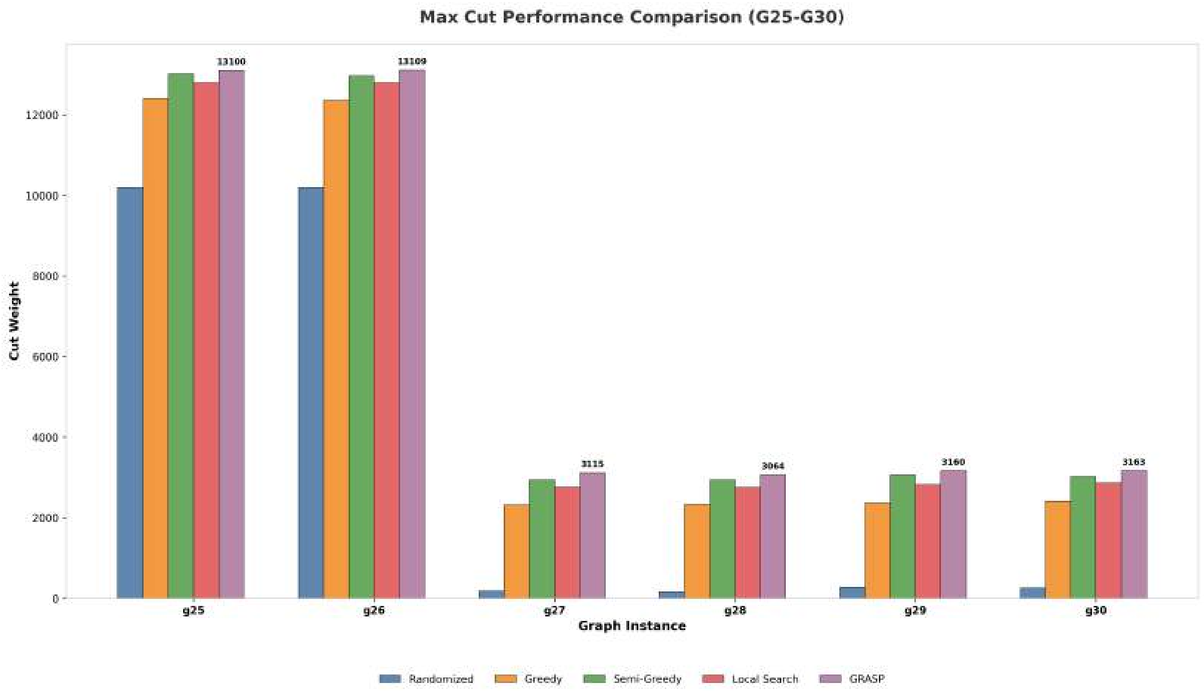

Figure 5: Max Cut Performance Comparison (G25-G30)

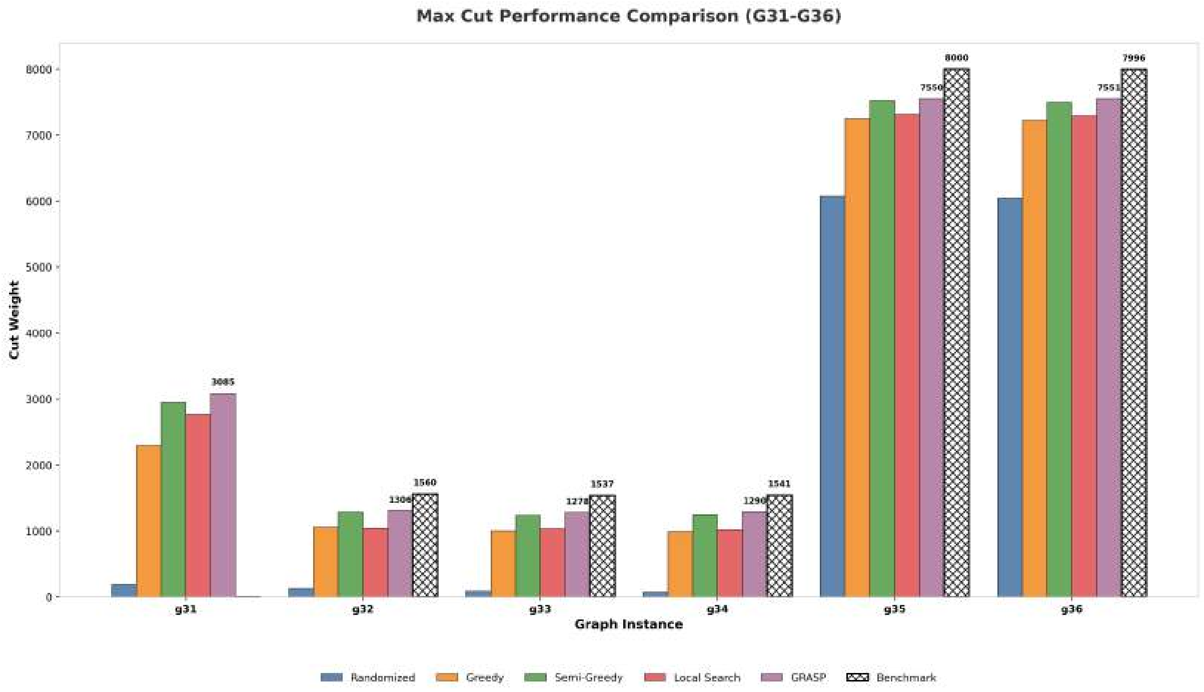

Figure 6: Max Cut Performance Comparison (G31-G36)

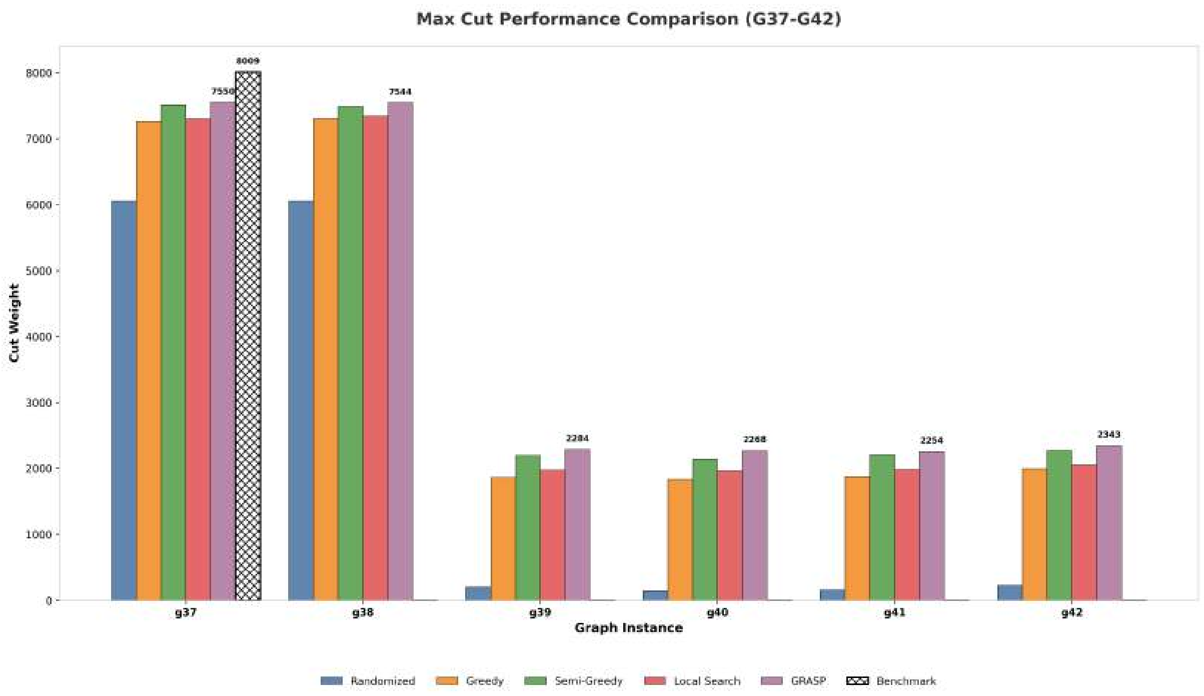

Figure 7: Max Cut Performance Comparison (G37-G42)

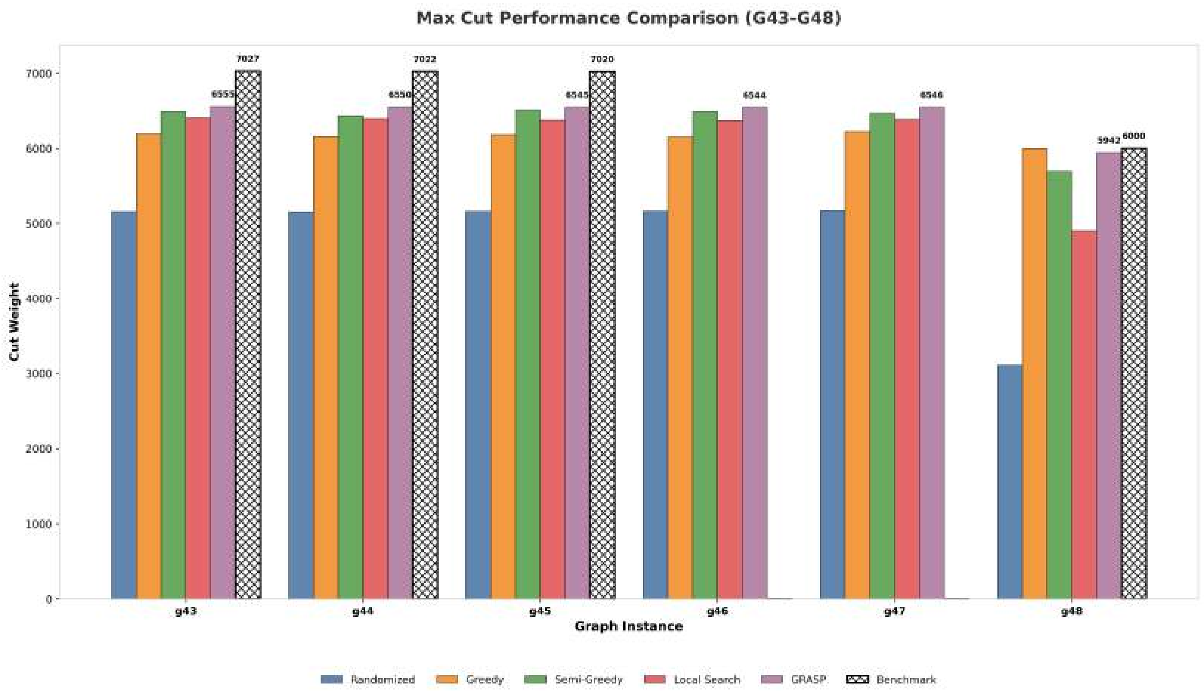

Figure 8: Max Cut Performance Comparison (G43-G48)

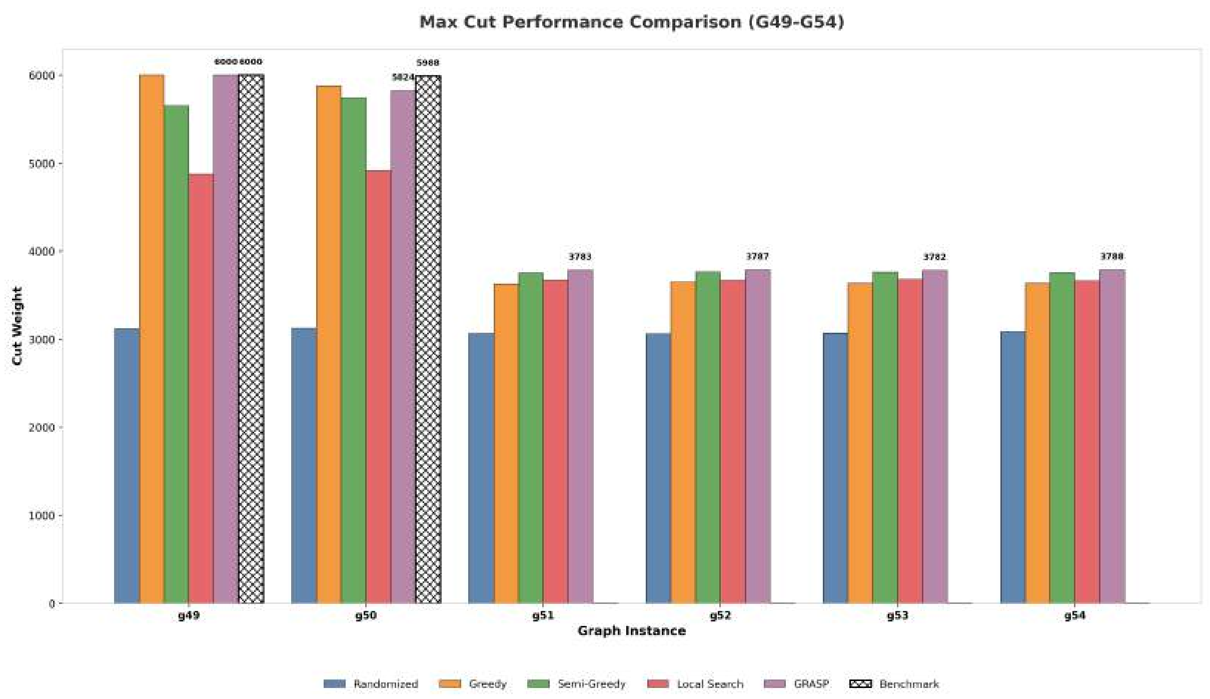

Figure 9: Max Cut Performance Comparison (G49-G54)

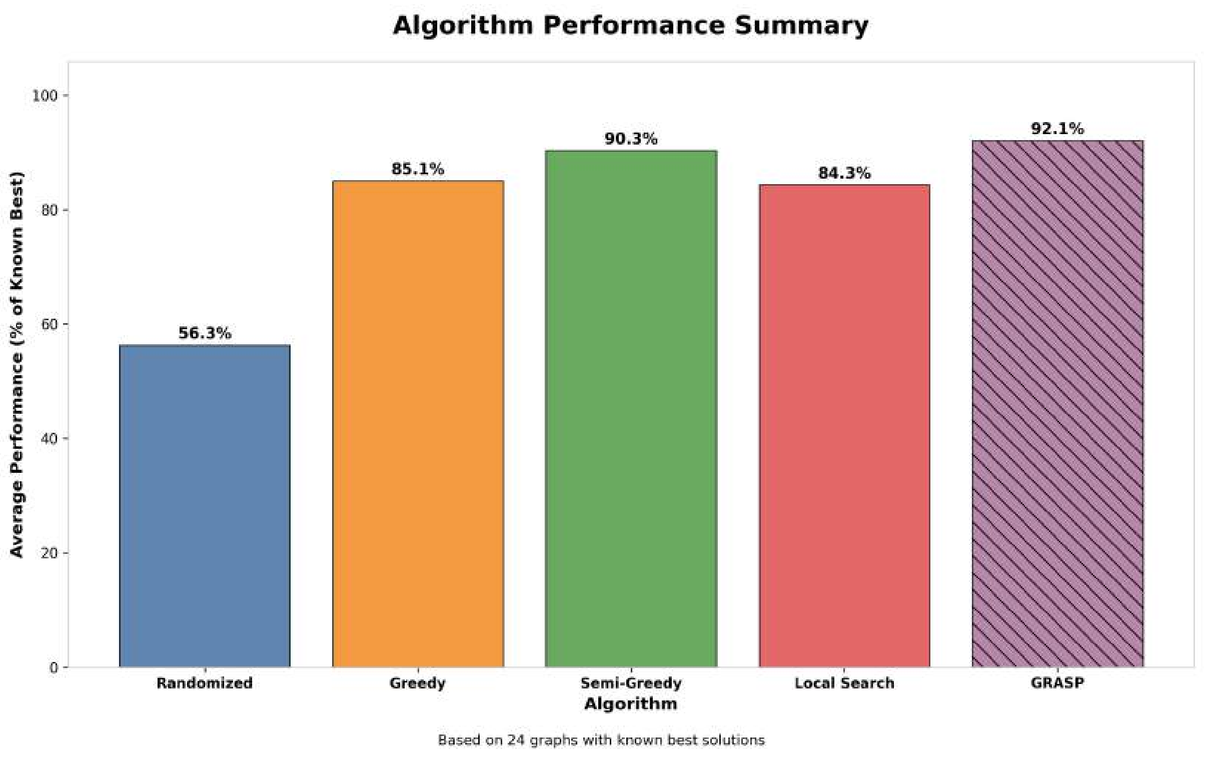

Figure 10: Algorithm Performance Summary (Percentage of Known Best)

4 Analysis and Discussion

4.1 Algorithm Performance

Based on the experimental results presented in the figures, several observations can be made:

- **GRASP** consistently outperforms other algorithms across all graph instances, achieving an average performance of 92.1% of the known best solutions. Its effectiveness can be attributed to the synergistic combination of diversification (semi-greedy construction) and intensification (local search).
- **Semi-Greedy** algorithm performs surprisingly well, reaching 90.3% of known best solutions on average. The controlled randomness introduced by the RCL mechanism allows it to explore different regions of the solution space, often finding high-quality partitions.
- **Greedy** and **Local Search** demonstrate comparable performance, achieving 85.1% and 84.3% of known best solutions, respectively. The greedy approach works well on sparse graphs with clear edge weight distinctions, while local search is effective but heavily dependent on the quality of the initial solution.
- **Randomized** algorithm, as expected, exhibits the poorest performance at 56.3% of known best solutions. However, its speed and simplicity make it valuable for generating diverse starting points for more sophisticated methods.

4.2 Graph Characteristics Impact

The relative performance of algorithms varies depending on graph characteristics:

- On dense graphs (e.g., G1-G3, G22-G26), the performance gap between algorithms narrows, with semi-greedy and GRASP maintaining a slight advantage. The higher connectivity provides more opportunities for creating balanced cuts.
- On sparse graphs (e.g., G11-G13, G32-G34), the performance disparity widens significantly. The randomized approach performs particularly poorly, while GRASP's adaptive nature allows it to identify the limited high-value cutting opportunities.
- For medium-sized graphs (e.g., G35-G38, G43-G47), GRASP and semi-greedy consistently approach the known best solutions, demonstrating their robustness across different graph topologies.

4.3 Computational Efficiency

While solution quality is paramount, computational efficiency is also a critical consideration:

- The randomized algorithm offers the fastest execution time, making it suitable for extremely large instances where approximate solutions are acceptable.
- Greedy construction provides a good balance between speed and solution quality, completing in $O(|V|^2)$ time.
- Local search's runtime is highly variable, depending on the number of improvement iterations required to reach a local optimum.
- GRASP, while computationally more intensive due to its iterative nature, delivers superior solutions that often justify the additional computational investment, especially for critical applications.

5 Conclusion

This comprehensive evaluation of heuristic algorithms for the MAX-CUT problem has demonstrated the following key findings:

- GRASP emerges as the most effective approach, consistently producing high-quality solutions across diverse graph instances. Its hybrid nature successfully balances exploration and exploitation, allowing it to escape local optima and discover near-optimal cuts.
- Semi-greedy construction proves to be a valuable standalone method, offering competitive performance with relatively low computational overhead. The randomization element helps overcome the limitations of purely deterministic approaches.
- The effectiveness of local search underscores the importance of solution refinement in combinatorial optimization. Even when seeded with moderate-quality initial solutions, local search can significantly enhance cut values.
- Simple algorithms like randomized partitioning establish important baselines but fall considerably short of more sophisticated approaches in solution quality.