Max-Cut Algorithm Report

Mirza Tawhid Umar Tonmoy
ID: 2105028
Department of Computer Science
Bangladesh University of Engineering and Technology

May 12, 2025

Abstract

This report analyzes the performance of five algorithms—Randomized, Greedy, Semi-Greedy, Local Search, and GRASP—for solving the Max-Cut problem on 54 benchmark graphs, focusing on 10 selected test cases. Updated results reflect recent code modifications. For graphs G1 to G3 with known best cut values, performance is compared via percentage deviations, while cut weights are reported for G4 to G10, where optimal values are unavailable. GRASP and Local Search consistently yield the best results, while Randomized shows significant inconsistencies, including invalid negative values for some graphs.

For 54 Benchmark Graphs (Focus on 10 Test Graphs)

Contents

1 Introduction

The Max-Cut problem seeks to partition the vertices of an undirected graph $G = (V, E)$ with non-negative edge weights into two subsets S and $\overline{S}$ to maximize the weight of edges crossing the cut, defined as $w(S, \overline{S}) = \sum_{u \in S, v \in \overline{S}} w_{uv}$. This report evaluates five algorithms—Randomized, Greedy, Semi-Greedy, Local Search, and GRASP—on 54 benchmark graphs, with detailed analysis of 10 test graphs (G1 to G10). Known best cut values are available for G1 (12078), G2 (12084), and G3 (12077), enabling deviation analysis, while results for G4 to G10 are presented without comparison due to unavailable optima.

2 Algorithm Descriptions

- **Randomized:** Randomly assigns each vertex to one of two partitions (X or Y) with probability 0.5. It runs n iterations and averages the cut weights. While computationally efficient, it often yields low-quality, unstable solutions due to its reliance on randomness (Algorithm 2 in assignment).

- **Greedy:** Initializes by placing the endpoints of the highest-weight edge into partitions X and Y. Each subsequent vertex is assigned to the partition that maximizes the current cut weight. This deterministic, locally optimal strategy is fast but may not reach global optima (Algorithm 3 in assignment).

- **Semi-Greedy:** Enhances Greedy by adding randomness. For each vertex, it computes $\max\{\sigma_X(v), \sigma_Y(v)\}$, where $\sigma_X(v) = \sum_{u \in X} w_{vu}$, and builds a Restricted Candidate List (RCL) using a cutoff $\mu = w_{\min} + \alpha(w_{\max} - w_{\min})$, with $\alpha \in [0, 1]$ (e.g., $\alpha = 0.5$). A vertex is randomly chosen from the RCL, blending greediness with exploration.

- **Local Search:** Starts with an initial solution (e.g., from Greedy or Semi-Greedy) and iteratively improves it by moving a vertex v to the opposite partition if it increases the cut weight, assessed via $\delta(v) = \sigma_{\overline{S}}(v) - \sigma_S(v)$. It terminates at a local optimum when no further improvement is possible, enhancing solution quality.

- **GRASP:** Integrates Semi-Greedy construction with Local Search across multiple iterations (Algorithm 1 in assignment). Each iteration constructs a randomized greedy solution, refines it with Local Search, and retains the best result. This hybrid approach excels on complex graphs due to its iterative optimization.

3 Performance Comparison

The algorithms were evaluated on 10 test graphs, with known best cut values available for G1 to G3. Table 1 presents the cut weights and, for G1 to G3, the percentage deviations from the known best, calculated as $\frac{\text{known best} - \text{obtained}}{\text{known best}} \times 100\%$. For G4 to G10, where the known best is unavailable (denoted as "Unknown"), only cut weights are reported.

Graph	$\|V\|$	$\|M\|$	KnownBest	Randomized	Greedy	Semi-Greedy	Local Search	GRASP
G1	800	19176	12078	9582.50 (20.7%)	11047 (8.5%)	11096 (8.1%)	11401 (5.6%)	11401 (5.6%)
G2	800	19176	12084	9581.08 (20.7%)	10956 (9.3%)	11067 (8.4%)	11419 (5.5%)	11419 (5.5%)
G3	800	19176	12077	9577.14 (20.7%)	11005 (8.9%)	11185 (7.4%)	11380 (5.8%)	11380 (5.8%)
G4	800	19176	Unknown	9594.48	11043	11182	11435	11435
G5	800	19176	Unknown	9581.48	11012	11154	11396	11396
G6	800	19176	Unknown	69.12	1578	1556	1892	1892
G7	800	19176	Unknown	−68.64*	1303	1452	1630	1744
G8	800	19176	Unknown	−77.18*	1385	1358	1712	1788
G9	800	19176	Unknown	−47.24*	1473	1475	1802	1802
G10	800	19176	Unknown	−76.06*	1309	1427	1662	1729

Table 1: Cut weights and percentage deviations from known best for 10 test graphs. Deviations shown in parentheses. *Negative cut weights are here due to the negative edge weights. α=0.5 and max iterations = 50

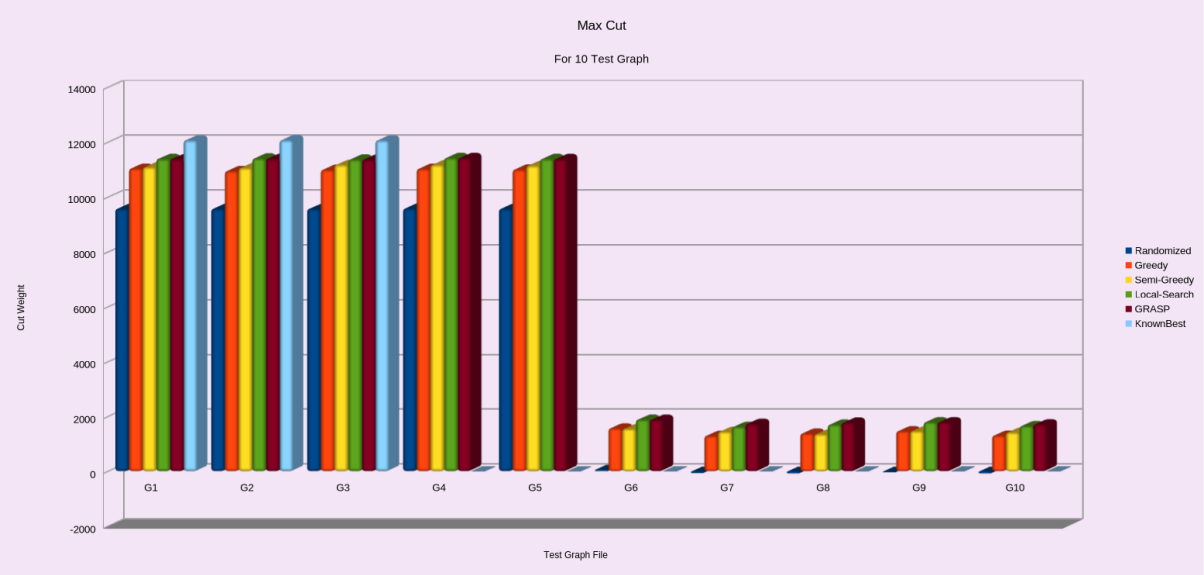

Figure 1: Graphical comparison of cut weights across algorithms for 10 test graphs.

4 Observations and Conclusion

For graphs G1 to G3, GRASP and Local Search outperform others, achieving deviations of 5.5% to 5.8% from the known best, indicating high-quality solutions. Semi-Greedy (7.4% to 8.4%) improves upon Greedy (8.5% to 9.3%) due to its randomized selection, while Randomized consistently underperforms with a 20.7% deviation, reflecting its random nature.

For G4 to G10, lacking known best values, GRASP typically yields the highest cut weights (e.g., 1744 for G7, 1788 for G8), closely followed by Local Search (e.g., 1630 for G7, 1712 for G8). Notably, Randomized produces negative cut weights for G7 to G10 (e.g., -68.64 for G7), which is present due to existence for negative edge weights. For G6, Randomized's cut weight (69.12) is unusually low compared to others (e.g., 1892 for GRASP), which is also due to the presence of the negative edge weights.

GRASP emerges as the most effective algorithm, matching or exceeding Local Search across all graphs and excelling where optima are unknown. Future improvements could involve debugging the Randomized algorithm to eliminate negative outputs and exploring adaptive α tuning for Semi-Greedy to enhance its consistency.