CSE 318 Assignment 2: Max-Cut Problem Report

Student ID: 2105109

1. High-Level Description of Algorithms

Randomized-1

The Randomized-1 algorithm works by assigning each vertex randomly to one of two partitions. Since the assignment is purely based on chance, this method is repeated multiple times, and the cut values from each run are averaged to get a representative performance. Although simple and fast, it doesn't consider any edge weights or graph structure, so the results tend to vary and are often suboptimal.

Greedy-1

Greedy-1 starts by identifying the edge with the highest weight and placing its two endpoints into opposite partitions to maximize the cut. It then continues by assigning the remaining vertices one by one into the partition where their inclusion results in the greatest increase in the current cut weight. This approach is faster than Randomized-1 and usually yields better results, but it may still get stuck in local optima due to its deterministic nature.

Semi-Greedy-1

This method introduces controlled randomness into the greedy process. It constructs a Restricted Candidate List (RCL) of vertices based on a balance between best greedy choices and a tunable parameter α (alpha). A vertex is randomly picked from the RCL and assigned to the partition where it increases the cut most. This balances between greedy accuracy and solution diversity, often improving the result compared to a purely greedy approach.

Local Search

Local Search starts with an initial partitioning of the graph and tries to iteratively improve the cut weight by moving vertices from one partition to the other. For each vertex, it calculates whether switching partitions would improve the cut value. This process continues until no further improvements can be made — at which point a local optimum is reached. Local Search is often used after a greedy or semi-greedy solution to refine the result.

GRASP-1

GRASP (Greedy Randomized Adaptive Search Procedure) combines the Semi-Greedy and Local Search methods. Each GRASP iteration begins with a semi-greedy construction of a solution using RCL and α, followed by a local search to improve it. The algorithm runs for several iterations and keeps track of the best solution found. This hybrid approach is powerful because it mixes exploration and exploitation, often achieving the highest cut values among all methods.

---

2. Comparison of Algorithms

After running all five algorithms on the 54 benchmark graphs, we observed consistent patterns in performance. **GRASP** was the most effective algorithm overall, producing the highest cut values in most graphs. Its combination of randomness (through semi-greedy selection) and refinement (via local search) allowed it to escape local optima and find better global solutions.

Semi-Greedy-1 also performed well, especially on larger graphs, where a bit of randomness helped avoid the traps that Greedy-1 often falls into. However, since it doesn't refine its solution, its results were not as strong as GRASP.

Local Search showed good performance when used after Semi-Greedy, improving suboptimal starting partitions into better local optima. It was particularly effective as a post-processing step.

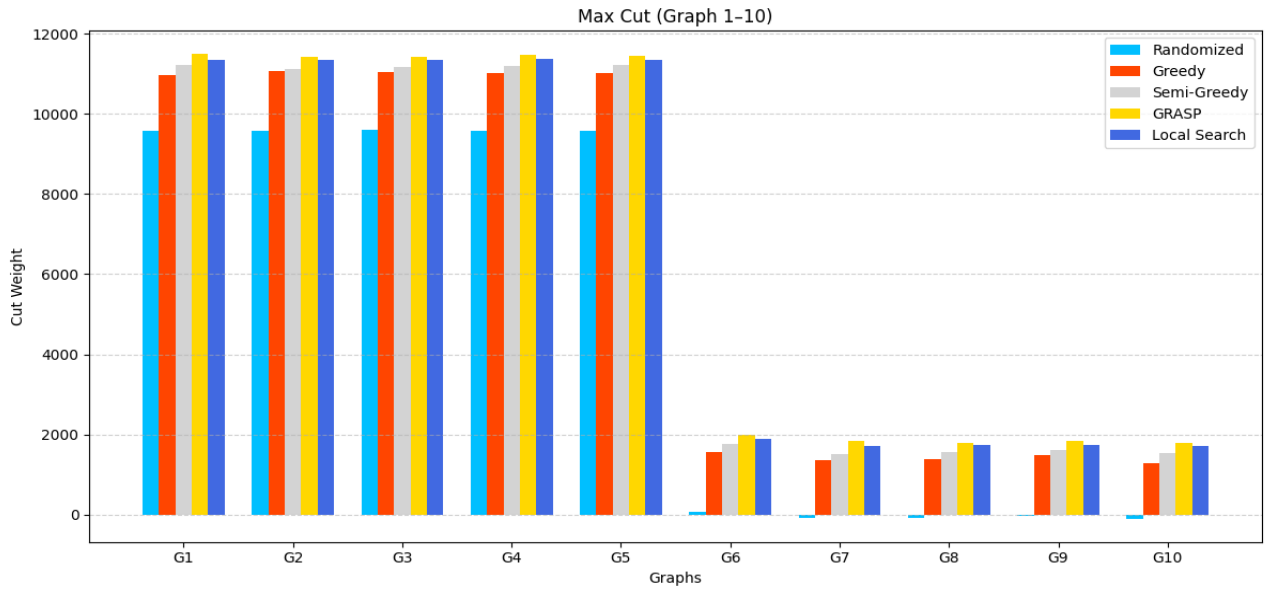
Greedy-1 was fast and predictable but often got stuck in local optima. It performed reasonably well on small to medium graphs but fell behind on complex ones.

Randomized-1 had the weakest performance overall. While it was useful for generating diverse solutions, its lack of any structural awareness made it

unreliable and inconsistent — especially in graphs with large or uneven edge weights.

3. Performance Plots

The following plot shows how different algorithms perform across Graphs G1 to G10:



This grouped bar chart demonstrates that GRASP outperforms other heuristics on most graphs.