CSE 318: Max-Cut by GRASP - Report (2105106)

1. High-Level Description of Implemented Algorithms

Randomized Heuristic

The Randomized Heuristic algorithm attempts to generate a valid solution by assigning each vertex to one of the two sets (set1 or set2) entirely at random. In our implementation, this process is repeated 10 times, and the average of the cut values from each run is taken as the final result. This approach is extremely fast, but its quality is usually quite poor because it does not consider the edge weights or any structural information about the graph. It's useful as a quick baseline for performance comparison.

Greedy Heuristic

The Greedy Heuristic begins by finding the edge with the highest weight and places its two vertices into opposite partitions (set1 and set2). For the remaining unassigned vertices, the algorithm computes the total edge weight that would result if the vertex were added to each partition. It then places the vertex in the partition that maximizes the contribution to the cut weight. This approach ensures a locally optimal choice at each step, but it may result in suboptimal global solutions since early decisions constrain future options. Nonetheless, it often produces better results than the Randomized Heuristic and serves as a strong baseline.

Semi-Greedy Heuristic

The Semi-Greedy Heuristic is an enhancement of the Greedy method that introduces controlled randomness to improve solution diversity. It uses a Restricted Candidate List (RCL) which contains nodes whose greedy value is above a certain threshold. The threshold is determined using the formula $\mu = w\_min + \alpha * (w\_max - w\_min)$, where α is a tunable parameter (0.5 in our implementation). Instead of always picking the best candidate, the algorithm randomly selects from this RCL, which allows for more exploration and a better chance of escaping poor local optima. This makes it a great choice for producing high-quality initial solutions.

Local Search

Local Search is a refinement strategy that starts with a candidate solution (generated using Semi-Greedy in our case). It iteratively checks whether moving a vertex from one partition to the other improves the overall cut weight. The move that results in the largest gain (delta) is applied, and the process continues until no improving move is found. This method helps reach a local optimum and usually improves the quality of the semi-greedy solution. To control execution time, we use fewer iterations for large graphs and more for smaller graphs.

GRASP

GRASP (Greedy Randomized Adaptive Search Procedure) is the most sophisticated method implemented. It combines the strengths of both construction and improvement phases. Each iteration of GRASP consists of a Semi-Greedy heuristic for constructing an initial solution, followed by Local Search to optimize it. The algorithm runs for a number of iterations (based on graph size: 50 for small, 10 for large), and the best solution among all iterations is selected. This combination of exploration (via randomized construction) and exploitation (via local search) enables GRASP to consistently produce high-quality solutions, often nearing the known best cut values.

2. Comparison of Algorithm Performance

All five algorithms were tested on 54 benchmark graphs of varying sizes and edge densities. The results reveal several important insights into the performance of each algorithm:

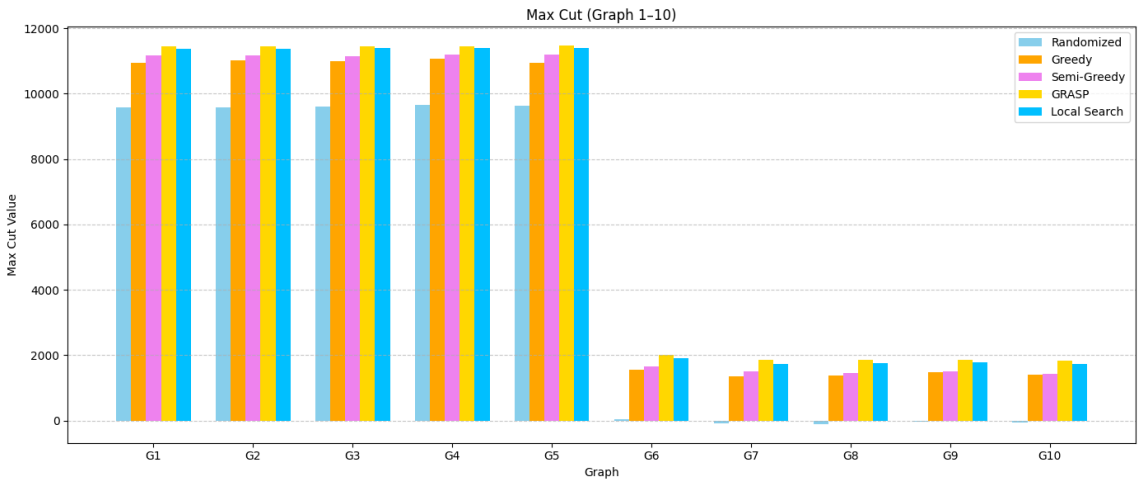
- The Randomized Heuristic is the fastest but also the most inconsistent and least effective. It serves primarily as a baseline for comparison and is not suitable for high-quality solutions.

- The Greedy Heuristic improves upon Randomized by making decisions based on edge weights, producing more consistent and stronger results. However, because it always chooses the local best move, it may get trapped in local optima.

- The Semi-Greedy Heuristic introduces randomness while still making informed decisions. It consistently performs better than the standard Greedy method, especially for graphs where the optimal cut is far from greedy decisions. It is a good compromise between randomness and structure.

- Local Search proves to be a powerful post-processing step that boosts the performance of semi-greedy solutions. It can bring solutions closer to local optima by refining partition assignments iteratively.

- GRASP consistently delivers the best results across nearly all benchmarks. It uses semi-greedy solutions as a starting point and refines them with local search. GRASP outperforms all other methods, often achieving values very close to the known best. Its multi-iteration design allows it to escape local optima and explore a broader solution space effectively.



To balance accuracy and runti

me, graphs with fewer than 1000 vertices were evaluated with 50 iterations for Local Search and GRASP, while larger graphs were evaluated using 10 iterations. This adaptive design ensures high performance without excessive runtime for large inputs.