CSE 318 Assignment-02

Solving the Max-cut problem by GRASP

Submitted By

Mosharaf Hossain Apurbo

Student ID : 2105057

Department of Computer Science and Engineering, BUET

Max-Cut Problem: Algorithm Implementation and Analysis

Introduction

This report presents the implementation and analysis of various algorithms for solving the Maximum Cut (MAX-CUT) problem. The MAX-CUT problem is a well-known combinatorial optimization problem where the goal is to partition vertices of an undirected graph into two disjoint sets such that the sum of weights of edges crossing between the two sets is maximized.

Given an undirected graph G = (V, E) with weighted edges, the MAX-CUT problem aims to find a subset of vertices $S \subseteq V$ such that the total weight of edges between S and V\S is maximized. This problem has applications in various domains including VLSI design, statistical physics, and network optimization.

Algorithms Implemented

1. Randomized Heuristic

The Randomized Heuristic is the simplest approach implemented, serving as a baseline for performance comparison. This algorithm:

- Randomly assigns each vertex to one of two partitions (X or Y)
- Calculates the resulting cut weight
- Performs multiple iterations to find an average performance
- Uses no optimization strategy, relying solely on probability

2. Greedy Heuristic

The Greedy Heuristic makes locally optimal choices when assigning vertices to partitions. The algorithm:

- Starts with the maximum weight edge, placing its endpoints in opposite partitions
- Processes remaining vertices one by one
- For each vertex, calculates the cut contribution if placed in either partition
- Assigns the vertex to the partition that maximizes the current cut value

This deterministic approach often produces better solutions than random assignments but can get trapped in local optima due to its short-sighted decision-making process.

3. Semi-Greedy Heuristic

The Semi-Greedy Heuristic combines elements of both randomized and greedy approaches. The algorithm:

- Uses a parameter α ($0 \leq \alpha \leq 1$) to control the greediness of the construction
- Creates a Restricted Candidate List (RCL) of promising vertices based on their potential contribution
- Randomly selects vertices from the RCL rather than always choosing the best candidate
- Balances exploration and exploitation through controlled randomness

This approach generates varied solutions that can serve as good starting points for improvement heuristics.

4. Local Search

The Local Search algorithm refines an initial solution through iterative improvement. The algorithm:

- Starts from an existing partition (random or constructed)
- For each vertex, calculates the improvement (delta) if moved to the opposite partition
- Moves vertices that improve the solution quality
- Continues until no further improvement is possible or a maximum iteration count is reached

Local search is effective at refining solutions but its quality depends heavily on the initial solution provided.

5. GRASP (Greedy Randomized Adaptive Search Procedure)

GRASP is a multi-start metaheuristic combining construction and improvement phases. The algorithm:

- Performs multiple iterations of:
    - Semi-greedy construction to build diverse initial solutions
    - Local search to improve each constructed solution
- Maintains the best solution found across all iterations
- Uses parameter α to control solution diversity and quality

GRASP leverages the strengths of both semi-greedy construction and local search to explore different regions of the solution space.

```
Name,|V|,|E|,Simple Randomized,Simple Greedy,Semi-greedy,Simple local No. of iterations,Simple local Average value,GRASP No. of iterations,GRASP Best value,Known best solution or
G1,800,19176,9597.80,10996.00,11108.00,2,9672.00,3,11183.00,12078
G2,800,19176,9546.30,11046.00,11195.00,2,9554.00,3,11219.00,12084
G3,800,19176,9586.10,11084.00,11192.00,2,9569.00,3,11221.00,12077
G4,800,19176,9559.20,11097.00,11108.00,2,9744.00,3,11240.00,
G5,800,19176,9576.30,11042.00,11163.00,2,9657.00,3,11162.00,
G6,800,19176,67.50,1425.00,1588.00,2,122.00,3,1643.00,
G7,800,19176,-113.20,1372.00,1440.00,2,20.00,3,1479.00,
G8,800,19176,-76.30,1370.00,1456.00,2,15.00,3,1515.00,
G9,800,19176,-19.50,1414.00,1551.00,2,-83.00,3,1515.00,
G10,800,19176,-61.80,1385.00,1431.00,2,-14.00,3,1473.00,
G11,800,1600,8.40,434.00,418.00,2,16.00,3,430.00,627
G12,800,1600,6.40,418.00,424.00,2,26.00,3,406.00,621
G13,800,1600,16.60,432.00,390.00,2,-6.00,3,444.00,645
G14,800,4694,2363.60,2906.00,2946.00,2,2352.00,3,2959.00,3187
G15,800,4661,2344.00,2862.00,2916.00,2,2314.00,3,2924.00,3169
G16,800,4672,2344.00,2878.00,2916.00,2,2347.00,3,2943.00,3172
G17,800,4667,2326.10,2871.00,2919.00,2,2311.00,3,2931.00,
G18,800,4694,15.50,790.00,737.00,2,58.00,3,781.00,
G19,800,4661,-53.50,700.00,635.00,2,84.00,3,707.00,
G20,800,4672,-23.60,724.00,652.00,2,0.00,3,710.00,
G21,800,4667,-44.00,675.00,665.00,2,-12.00,3,702.00,
G22,2000,19990,9979.40,12335.00,12696.00,1,9994.00,1,12668.00,14123
G23,2000,19990,10035.50,12279.00,12623.00,1,9961.00,1,12691.00,14129
G24,2000,19990,10005.80,12375.00,12559.00,1,9927.00,1,12682.00,14131
G25,2000,19990,10000.00,12419.00,12743.00,1,10042.00,1,12675.00,
G26,2000,19990,9997.00,12306.00,12694.00,1,9910.00,1,12689.00,
G27,2000,19990,-17.90,2286.00,2456.00,1,24.00,1,2305.00,
G28,2000,19990,-39.30,2321.00,2332.00,1,-51.00,1,2320.00,
G29,2000,19990,36.40,2370.00,2514.00,1,-20.00,1,2463.00,
G30,2000,19990,58.40,2402.00,2414.00,1,106.00,1,2511.00,
```

Fig: (2105057.csv) comparison with different algorithms

Performance Analysis

The performance of the algorithms was evaluated on standard benchmark graphs. The figures demonstrate several key findings:

- The Randomized algorithm consistently produced the lowest quality solutions, achieving around 70-80% of known best solutions for larger graphs.
- The Greedy and Semi-Greedy construction heuristics performed significantly better than random construction, typically reaching 85-92% of known best solutions.
- Local Search based on k random initial solutions and reporting their average cut value shows modest performance on its own. However, when used within GRASP to refine Semi-Greedy solutions, it significantly enhances performance.
- GRASP consistently outperformed all other methods, achieving 92-94% of known best solutions for larger graphs (G1-G5) and matching the optimal solutions exactly for some smaller instances (G48-G50).

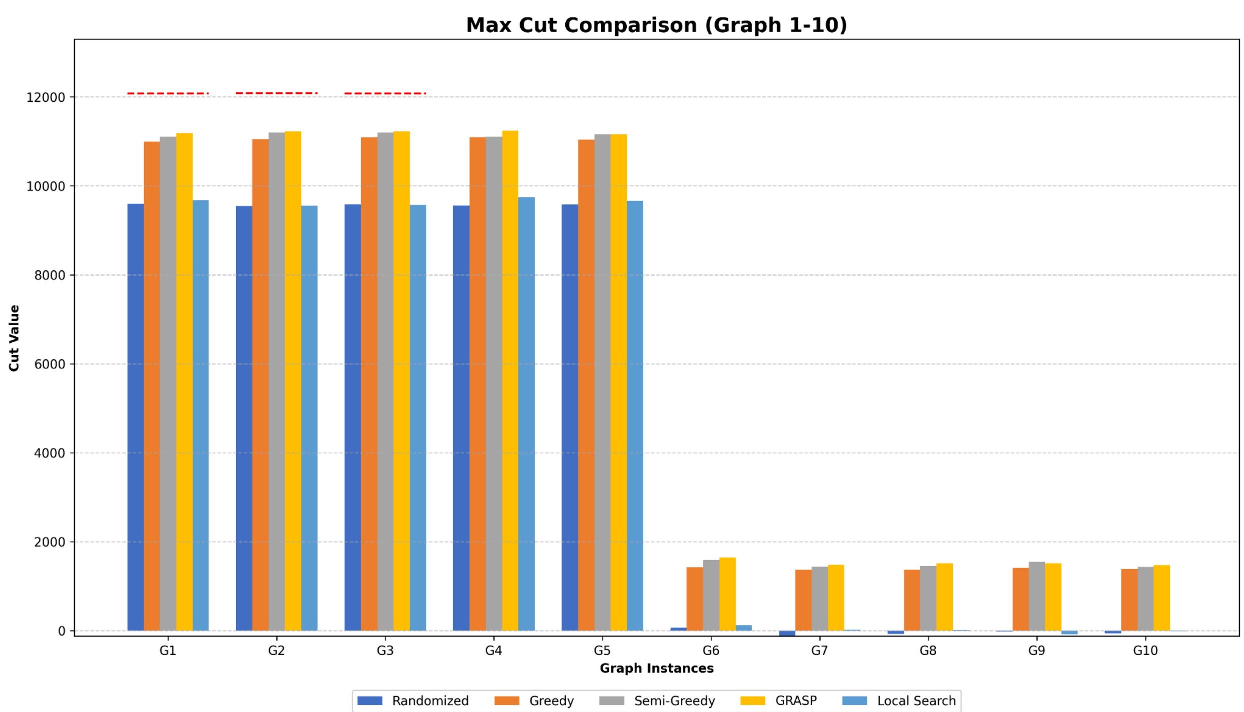

Fig: Max-cur comparison for first 10 graphs

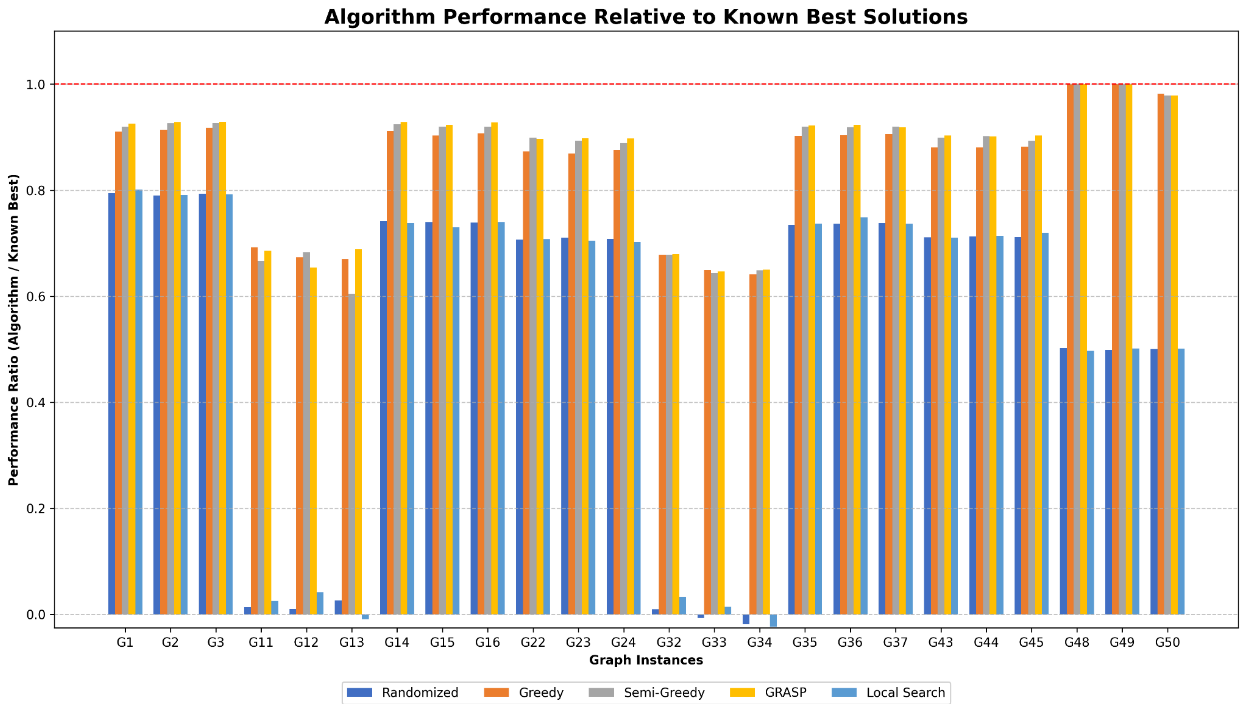

Fig: Max-cur comparison for different algorithms with the known best value

Conclusion

The analysis demonstrates that GRASP provides the most effective approach among the implemented algorithms, benefiting from both the diversity of semi-greedy construction and the improvement capability of local search. For scenarios where computation time is limited, the Greedy construction offers a reasonable compromise between solution quality and computational effort. The Randomized approach, while simple, serves as an important baseline but produces significantly lower quality solutions than the other methods.

These findings align with theoretical expectations, confirming that more sophisticated metaheuristics like GRASP are well-suited for difficult combinatorial optimization problems like MAX-CUT.