# Report on Max-Cut Problem Solving using GRASP

1905113-Anamul Hoque Emtiaj

August 19, 2023

## 1 Objective

The objective of this report is to analyze the results obtained by applying the Greedy Randomized Adaptive Search Procedure (GRASP) to solve the Maximum Cut (Max-Cut) problem. The Max-Cut problem involves partitioning the vertices of an undirected graph into two sets, such that the sum of the weights of the edges crossing the partition is maximized.

## 2 Algorithm Analysis

The provided C++ code implements a GRASP to solve the Max-Cut problem. The algorithm consists of multiple phases, including a semi-greedy construction phase, local search phase, and iterations of the GRASP process. Here's a breakdown of the key components:

1. **Semi-Greedy Construction Phase:**
    - The semi-greedy construction phase aims to create an initial solution with controlled randomness.
    - Greedy values for each vertex are calculated based on the sum of the edge weights incident on the vertex. Vertices are ranked by these greedy values.
    - A Restricted Candidate List (RCL) is formed, containing vertices with greedy values within a certain range.
    - A vertex is randomly selected from the RCL and added to the solution set.
2. **Local Search Phase:**
    - The local search phase aims to optimize the solution obtained from the construction phase.
    - The heuristic calculates the improvement in cut value for moving a vertex between the two partitions (cuts).
    - The delta value is calculated for each vertex, representing the change in cut value if the vertex is moved.
    - Vertices with positive delta values are moved to the partition that leads to the most improvement in cut value.
    - This process continues iteratively until no further improvement can be made.
3. **GRASP Iterations:**

- The GRASP process involves multiple iterations, each consisting of constructing an initial solution using the semi-greedy approach and applying local search to refine the solution.
- The best solution across all iterations is retained.

## 3 Output Observations

The algorithm is evaluated on a variety of problem instances with varying sizes and edge densities. The following observations are made regarding the performance of different heuristics:

| Problem Name | V | E | Randomized | Greedy | Semi-greedy | local-Search | GRASP | Upper Bound |
|---|---|---|---|---|---|---|---|---|
| g22 | 2000 | 19990 | 12355.7 | 12809 | 12755.7 | 12896 | 13011 | 14123 |
| g25 | 2000 | 19990 | 12346.3 | 12849 | 12759.2 | 12906 | 13005 | N. A. |
| g24 | 2000 | 19990 | 12354.3 | 12806 | 12760.5 | 12909 | 13002 | 14131 |
| g23 | 2000 | 19990 | 12341.5 | 12758 | 12757.6 | 12889 | 12986 | 14129 |
| g26 | 2000 | 19990 | 12343.1 | 12838 | 12759 | 12874 | 12973 | N. A. |
| g3 | 800 | 19176 | 11012.2 | 11234 | 11223.1 | 11460 | 11500 | 12077 |
| g5 | 800 | 19176 | 11024.6 | 11275 | 11228.9 | 11450 | 11479 | N. A. |
| g4 | 800 | 19176 | 11009.5 | 11256 | 11223.5 | 11495 | 11467 | N. A. |
| g1 | 800 | 19176 | 11025 | 11195 | 11222 | 11442 | 11466 | 12078 |
| g2 | 800 | 19176 | 11003.5 | 11271 | 11227.6 | 11410 | 11460 | 12084 |

Table 1: Summary of Results for Different Algorithms

- **Simple Randomized:**
  Merits: Provides a quick initial solution, simple to implement.
  Demerits: Lacks optimization due to its random nature, may not produce competitive solutions.
- **Simple Greedy:**
  Merits: Incorporates greedy choices, resulting in improved cut values.
  Demerits: Still limited by deterministic nature, may not find global optima.
- **Semi-greedy:**
  Merits: Balances exploration and exploitation, introduces controlled randomness.
  Demerits: Still deterministic to some extent, may not escape local optima.
- **Local Search:**
  Merits: Optimizes solutions locally, refines initial solutions.
  Demerits: Can get stuck in local optima, requires efficient neighborhood exploration.
- **GRASP:**
  Merits: Combines exploration (semi-greedy) and exploitation (local search), consistently produces competitive cut values.
  Demerits: Relies on proper parameter tuning, may still struggle with complex instances.

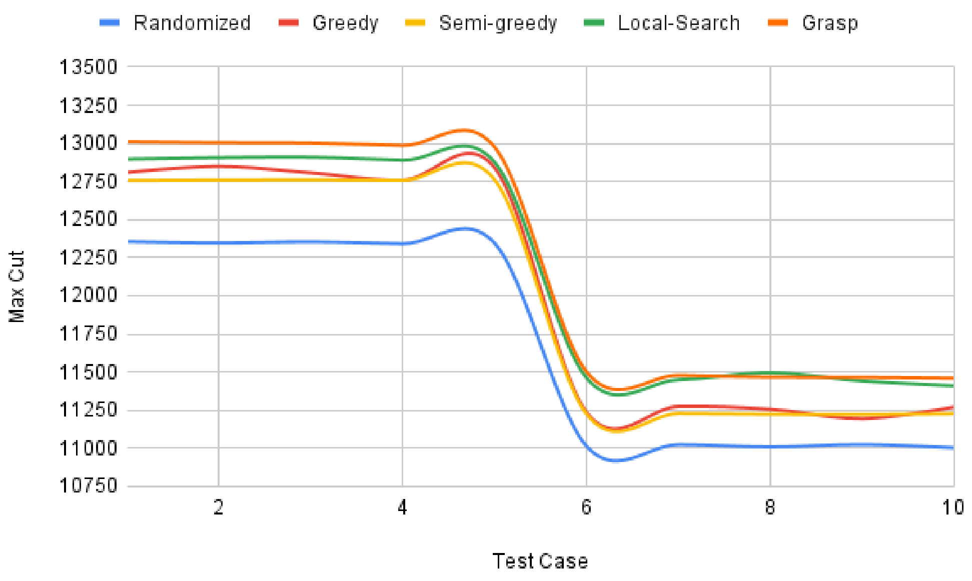

Figure 1: Comparison of Algorithms: Randomized, Semi-greedy, Greedy, Local Search, and GRASP

The number of iterations for GRASP affects the final cut value, as more iterations may lead to better solutions.

## 4 Conclusion

The GRASP heuristic demonstrates its effectiveness in solving the Max-Cut problem by producing competitive cut values compared to other simpler algorithms. The combination of a semi-greedy construction phase, local search for optimization, and multiple iterations in the GRASP process contributes to improved solution quality. It is observed that the heuristic's performance is influenced by parameters such as the number of iterations and the alpha value used in the semi-greedy phase. Experimentation and fine-tuning of these parameters could potentially lead to further improvements in solution quality.