MaxCut Algorithm

1. **Randomized:** A simple probabilistic approach that randomly assigns vertices to two sets (S and S_bar). For each vertex, a random decision is made to place it in either set with equal probability. The algorithm runs multiple trials and returns the maximum cut weight found. The randomized nature helps avoid getting stuck in local optima but lacks any strategic vertex placement. To get a reliable estimate of the cut size, the algorithm is run n times and the results are averaged.

2. **Greedy:** A deterministic algorithm that starts by selecting the edge with maximum weight and placing its endpoints in opposite sets. It then iteratively processes each remaining vertex by calculating the sum of edge weights connecting it to vertices in each set (S and $\bar{S}$) and places the vertex in the set that maximizes the cut weight. This approach makes locally optimal decisions at each step but may not find the global optimum**.**

3. **Semi-Greedy:** An extension of the greedy approach that introduces controlled randomness through a parameter α (0≤α≤1). Instead of always selecting the best vertex, it creates a Restricted Candidate List (RCL) containing vertices whose greedy function value ≥ μ. A vertex is then randomly selected from this list. This balances between pure greedy (α=0) and pure random (α=1) strategies, allowing the algorithm to explore different solution paths while still maintaining some quality control.

4. **Local Search:** An improvement algorithm that starts with an initial solution (typically from Semi-Greedy) and iteratively enhances it. It examines each vertex to determine if moving it from its current set to the opposite set would increase the cut weight. If an improvement is found, the move is made, and the process continues until no further improvements are possible (local optimum). This technique refines

existing solutions by making small, beneficial changes but can get trapped in local optima. Multiple random restarts are often used to mitigate this limitation.

5. **GRASP:** A multi-start metaheuristic that combines Semi-Greedy construction with Local Search improvement. Each iteration consists of two phases: (1) construction phase using Semi-Greedy to build an initial solution with controlled randomness, and (2) improvement phase using Local Search to refine the solution to a local optimum. The best solution across all iterations is returned. GRASP effectively balances exploration (through randomized construction) and exploitation (through local optimization), making it particularly effective for combinatorial optimization problems like MaxCut.

Algorithm Performance Comparison Analysis

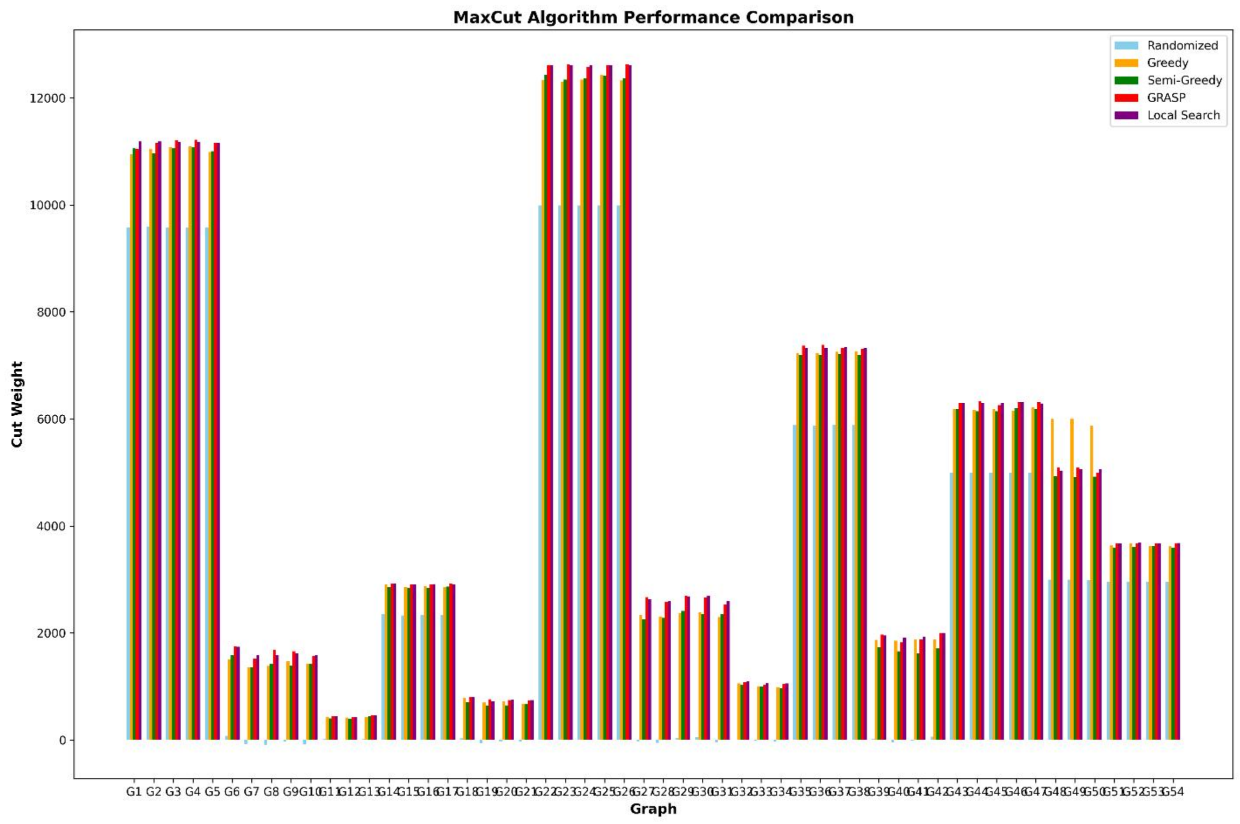
1. **GRASP Performance:**
   - Consistently achieves the best results across different graph sizes
   - Combines the benefits of Semi-Greedy construction and Local Search improvement
   - Higher computational cost due to multiple iterations and local search phase
   - Most effective for complex graphs with many vertices and edges

2. **Local Search:**
   - Second-best performer after GRASP
   - Very effective at improving existing solutions
   - Performance heavily depends on initial solution quality
   - Shows good consistency across multiple runs

3. **Semi-Greedy:**
    - Better than pure Greedy approach due to controlled randomness (α = 0.7)
    - Good balance between solution quality and computational speed
    - More robust than Greedy due to exploration of different solution paths

4. **Greedy:**
    - Fast and deterministic
    - Generally outperforms Randomized approach
    - Can get stuck in local optima
    - Performance degrades for larger, more complex graphs
    -
5. **Randomized:**
    - Provides baseline performance.
    - Least consistent results due to pure randomness
    - Fastest execution time but poorest solution quality
    - Useful mainly as a benchmark for other algorithms

The results show a clear hierarchy: GRASP > Local Search > Semi-Greedy > Greedy > Randomized, with GRASP consistently finding the best solutions at the cost of higher computational time.


MaxCut Algorithm Performance Comparison
Randomized
Greedy
Semi-Greedy
GRASP
Local Search
Cut Weight
Graph
0
2000
4000
6000
8000
10000
12000
G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14 G15 G16 G17 G18 G19 G20 G21 G22 G23 G24 G25 G26 G27 G28 G29 G30 G31 G32 G33 G34 G35 G36 G37 G38 G39 G40 G41 G42 G43 G44 G45 G46 G47 G48 G49 G50 G51 G52 G53 G54