CSE 318 Assignment-02

Solving the Max-Cut Problem by GRASP

Name: Shemanty Mahjabin

Student ID: 2105091

Level 3, Term 1

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology (BUET)

1 Problem Overview

The Maximum Cut (MAX-CUT) problem involves partitioning the vertex set V of an undirected graph $G = (V, E)$ into two disjoint subsets S and $\bar{S}$, such that the sum of the weights of the edges crossing the partition (i.e., one endpoint in S, one in $\bar{S}$) is maximized.

2 Algorithms Implemented

2.1 Randomized Heuristic

The Randomized Max-Cut algorithm generates random cuts by randomly assigning each vertex to either of the two sets X or Y. It repeats this process multiple times, keeping the best cut weight found during the trials. This algorithm attempts to explore the solution space by introducing randomness, which may help avoid local optima.

2.2 Greedy Heuristic

The Greedy Max-Cut algorithm is a simple constructive algorithm that starts by selecting the largest edge in the graph and placing its vertices into two separate sets X and Y. Then, for each of the remaining vertices, it computes the cut weight with respect to both sets and assigns the vertex to the set that results in a larger cut weight. This approach is greedy because it always chooses the immediate best option.

2.3 Semi-Greedy Heuristic

The semi-greedy method builds upon the greedy heuristic by adding randomness. Vertices are selected from a Restricted Candidate List (RCL), which includes candidates whose greedy value is above a threshold:

$$\mu = w_{\min} + \alpha(w_{\max} - w_{\min}),$$

where $\alpha \in [0, 1]$. We used value-based selection with different values of α (e.g., 0.3, 0.5, 0.7).

2.4 Local Search

Starting from an initial feasible solution (e.g., from greedy or semi-greedy), local search explores the neighborhood by flipping one vertex from its current partition to the other. The gain $\delta(v)$ is calculated, and the vertex with the maximum positive gain is moved. The search terminates when no such improvement exists.

2.5 GRASP (Greedy Randomized Adaptive Search Procedure)

GRASP combines semi-greedy construction with local search in an iterative multistart framework. For each iteration:

1. A solution is built using the semi-greedy method.

2. Local search is applied to improve it.

3. The best overall solution is recorded.

We ran GRASP for a fixed number of iterations (e.g., 10,100) per graph.

3 Observations

- **Greedy**: This algorithm is fast but may not always provide the best solution. It is prone to getting stuck in local optima. It works best on simpler problems or when quick approximations are required.
- **Randomized**: This algorithm performs better in some cases because of its random nature, helping it avoid local minima. However, it may require more trials to find an optimal or near-optimal solution. It is useful when exploring a large solution space.
- **Semi-Greedy**: The semi-greedy algorithm strikes a balance between randomization and greedy selection. It can perform better than the greedy algorithm but still suffers from local optima. It is good for problems where both exploration and exploitation are needed.
- **Local Search**: This algorithm improves a given solution iteratively, refining it to a local optimum. It is effective when combined with other techniques (like semi-greedy or GRASP). It is slower and might get stuck in local minima, making it more suitable when fine-tuning solutions.
- **GRASP**: GRASP combines the best of both worlds by using a greedy construction step followed by a local search phase. It has a higher computational cost but tends to produce better solutions than the individual components. It is generally the most reliable among the discussed algorithms.

4 Sample Results

Problem	$\|V\|$	$\|E\|$	Simple Random	Simple Greedy	Semi Greedy	LS Iter	LS Best	GRASP Iter	GRASP Best	Best Known
g1	800	19176	9479	10954	11086	6	11382	100	11469	12078
g2	800	19176	9604	11046	11165	5	11337	100	11477	12084
g3	800	19176	9538	11076	11237	5	11408	100	11494	12077
g4	800	19176	9673	11098	10969	6	11274	100	11474	
g5	800	19176	9712	10992	10995	6	11335	100	11490	
g6	800	19176	140	1506	1766	6	1973	100	2026	
g7	800	19176	-5	1355	1512	4	1658	100	1839	
g8	800	19176	-194	1383	1392	9	1687	100	1878	
g9	800	19176	37	1474	1658	6	1808	100	1895	
g10	800	19176	-43	1428	1369	8	1715	100	1840	
g11	800	1600	14	434	408	2	446	100	510	627
g12	800	1600	18	418	470	2	482	100	498	621
g13	800	1600	18	432	478	2	486	10	506	645
g14	800	4694	2356	2903	2944	3	2976	10	2982	3187
g15	800	4661	2310	2859	2910	3	2940	10	2966	3169
g16	800	4672	2319	2878	2918	3	2946	10	2967	3172
g17	800	4667	2348	2863	2925	3	2960	10	2968	
g18	800	4694	45	790	683	4	808	10	907	
g19	800	4661	-74	700	641	4	742	10	799	
g20	800	4672	-38	724	766	4	817	10	851	
g21	800	4667	-20	675	720	4	813	10	831	
g22	2000	19990	9999	12335	12339	8	12874	10	12952	14123
g23	2000	19990	9905	12304	12765	8	12935	10	13003	14129
g24	2000	19990	9953	12344	12784	4	12910	10	12918	14131
g25	2000	19990	9970	12426	12615	6	12852	10	12932	
g26	2000	19990	10060	12328	12700	6	12909	10	12983	
g27	2000	19990	-46	2339	2452	10	2838	10	2933	
g28	2000	19990	-4	2302	2330	5	2756	10	2892	
g29	2000	19990	-6	2371	2711	6	2993	10	2993	
g30	2000	19990	24	2384	2794	6	2983	10	2978	
g31	2000	19990	-204	2298	2519	7	2872	10	2928	
g32	2000	4000	62	1058	1186	2	1202	10	1212	1560
g33	2000	4000	20	998	988	3	1058	10	1210	1537
g34	2000	4000	-42	988	990	2	1052	10	1192	1541
g35	2000	11778	5911	7222	7315	3	7415	10	7469	8000
g36	2000	11766	5850	7230	7402	2	7464	10	7449	7996
g37	2000	11785	5837	7255	7304	3	7411	10	7465	8009
g38	2000	11779	5929	7261	7234	4	7399	10	7459	
g39	2000	11778	-3	1874	2018	4	2154	10	2096	
g40	2000	11766	-52	1858	1826	5	2090	10	2151	
g41	2000	11785	-124	1876	1688	4	1988	10	2171	
g42	2000	11779	-7	1876	1716	4	2072	10	2191	
g43	1000	9990	5092	6187	6169	7	6419	10	6462	7027
g44	1000	9990	4993	6173	6322	4	6432	10	6498	7022
g45	1000	9990	4962	6183	6261	11	6481	10	6479	7020
g46	1000	9990	4903	6155	6363	5	6458	10	6482	
g47	1000	9990	4952	6214	6379	5	6464	10	6501	
g48	3000	6000	3108	6000	6000	1	6000	10	6000	6000
g49	3000	6000	3024	6000	6000	1	6000	10	6000	6000
g50	3000	6000	2960	5880	5880	1	5880	10	5880	5988
g51	1000	5909	2916	3633	3697	3	3728	10	3745	
g52	1000	5916	2925	3672	3690	3	3735	10	3748	
g53	1000	5914	2922	3632	3699	3	3743	10	3744	
g54	1000	5916	2994	3619	3691	3	3735	10	3737	

Table 1: Benchmark Results for 54 Graph Instances

5 Visualizations

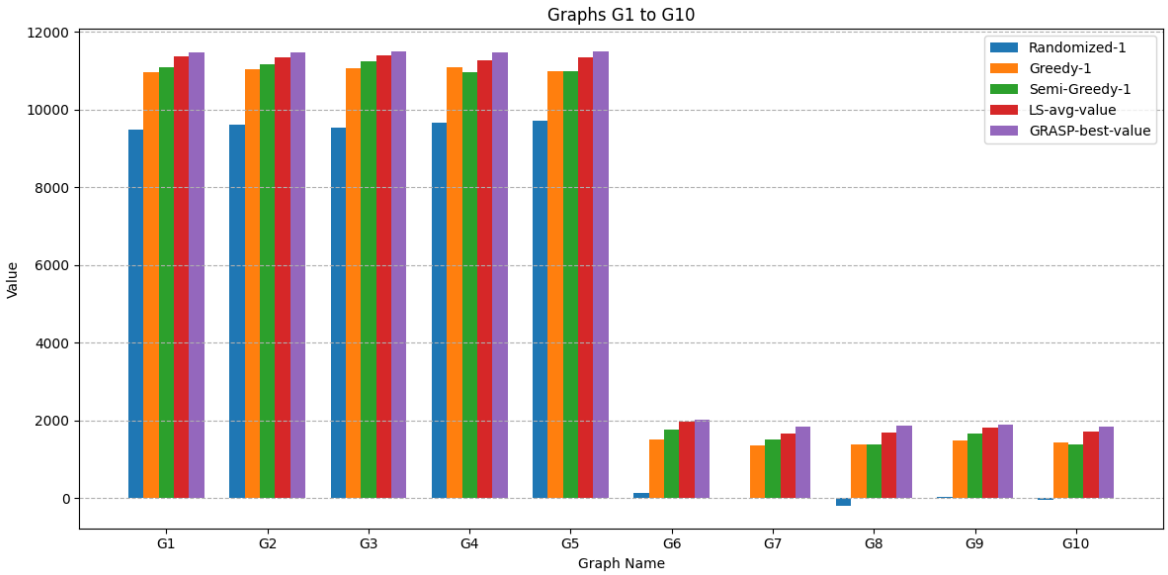

Figure 1: Cut value comparison on graphs G1 to G10

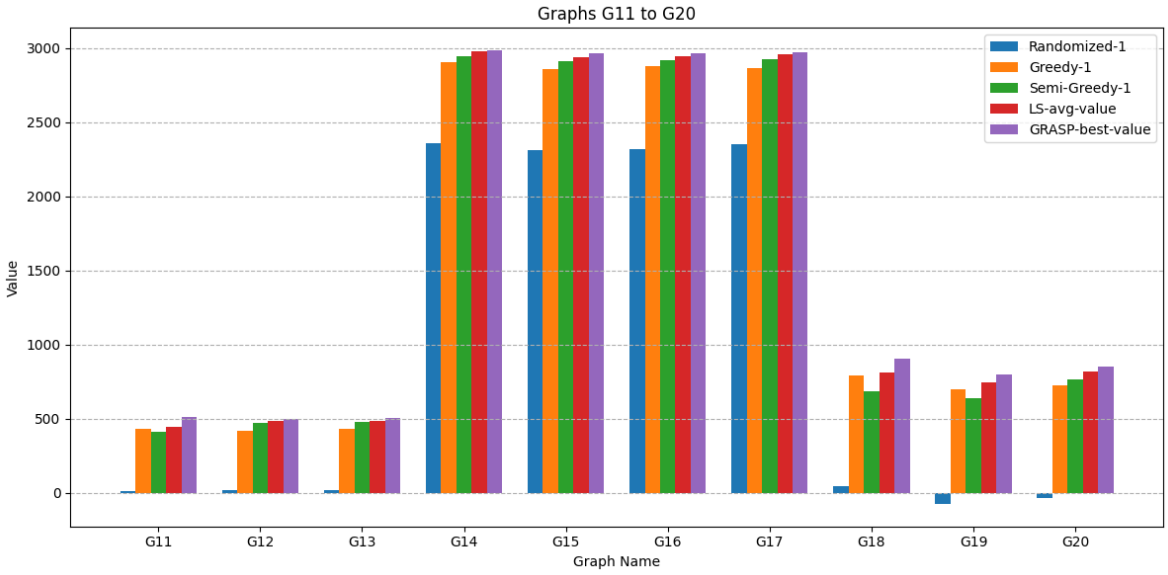

Figure 2: Cut value comparison on graphs G11 to G20

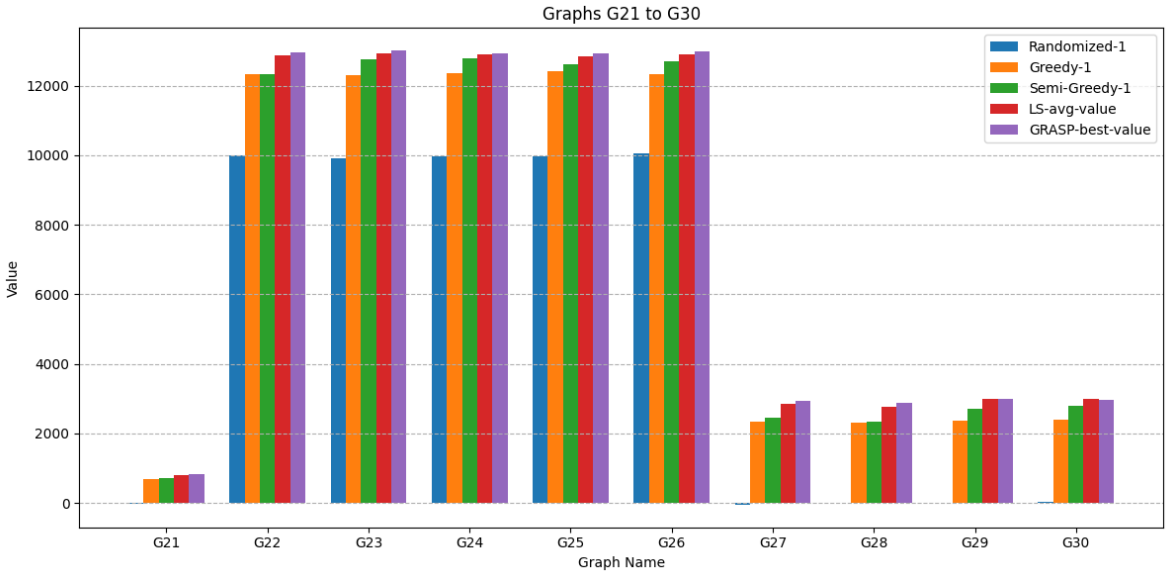

Figure 3: Cut value comparison on graphs G21 to G30

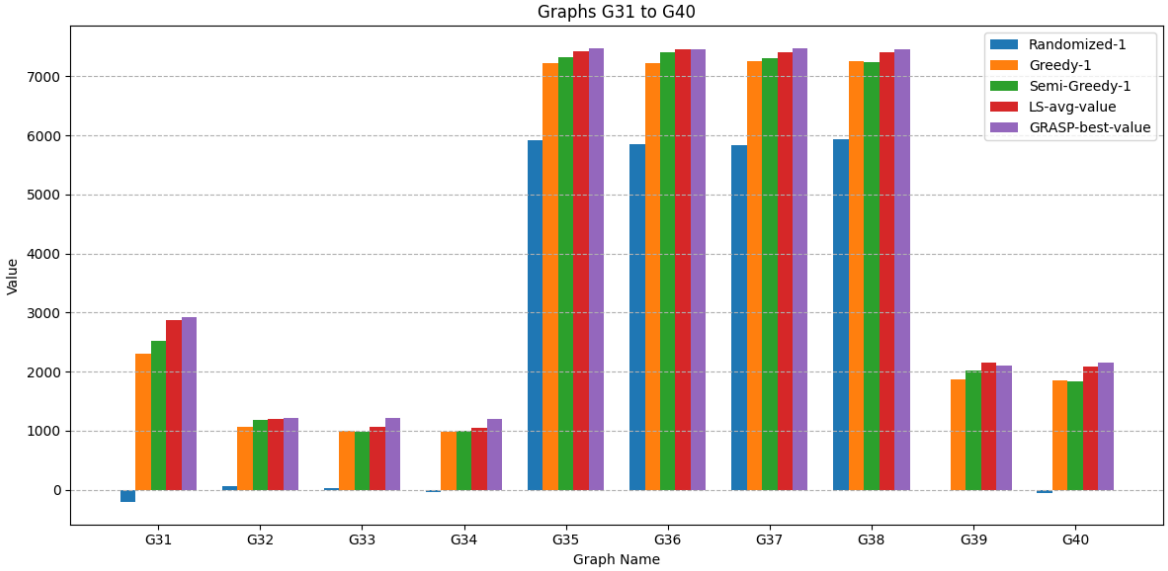

Figure 4: Cut value comparison on graphs G31 to G40

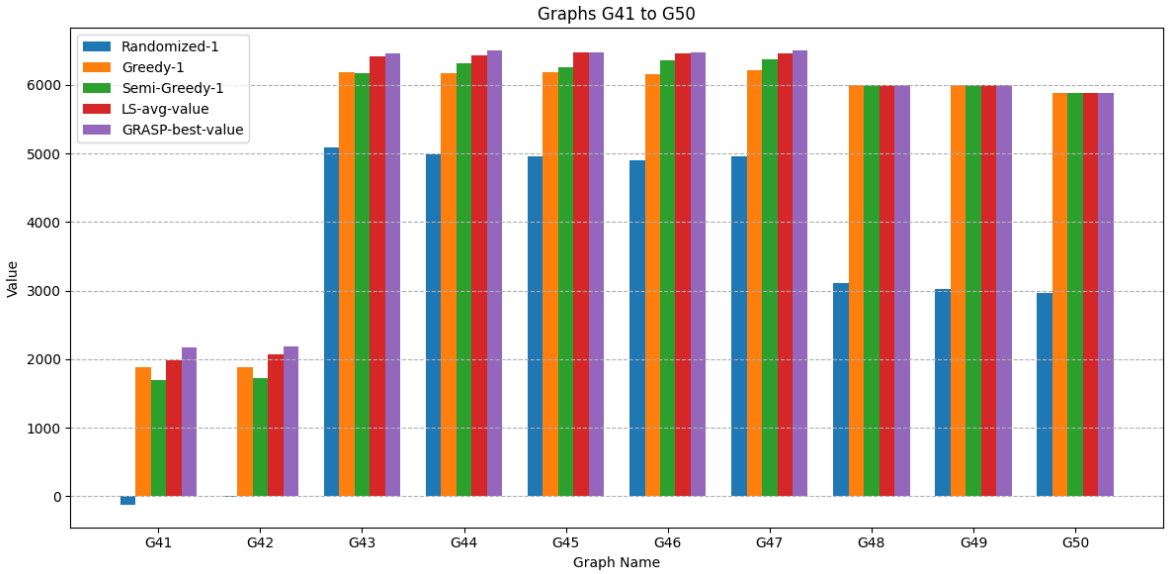

Figure 5: Cut value comparison on graphs G41 to G50

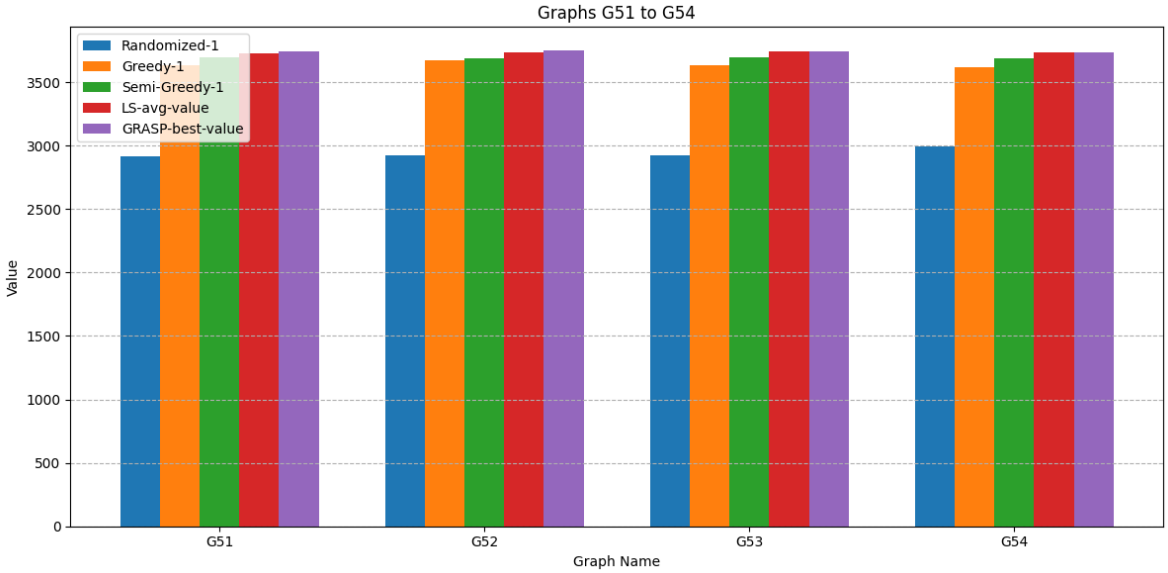

Figure 6: Cut value comparison on graphs G51 to G54

6 Results & Observations

- Greedy generally provides a strong baseline.
- Randomized performs poorly in most graphs, but it's very fast.
- Semi-Greedy performs better than pure Greedy on many instances when α is tuned well

- Local Search improves solutions significantly when started from a good initial point.
- GRASP consistently outperforms other heuristics across most graphs by combining semi-greedy diversity and local search refinement.

7 Conclusion

GRASP achieved the best performance across the benchmark set, showing the strength of combining randomized construction with local improvement. Semi-greedy alone can also provide good results, especially with carefully tuned α. Purely greedy and randomized methods are either too deterministic or too inconsistent.