Bangladesh University of Engineering and Technology

CSE 318
Artificial Intelligence Sessional

Solving the Max-cut problem by GRASP

A Brief Report Explaining the Output

Md. Raihan Sobhan

Student ID: 1905095
Subsection: B2

August 20, 2023

Techniques implemented:

I have implemented the following algorithms to find a better solution of the max-cut problem of a graph,

1. **Simple Greedy Heuristic:** The algorithm starts by placing one vertex to each partition X and Y (initially both are empty) such that each contains an endpoint of a largest-weight edge. The remaining $|V|$ - 2 vertices are examined one by one to find out the placement of which vertex to either one of the two partitions maximizes the weight of the partial cut constructed so far. At each iteration, one vertex is placed to either set X or set Y.
   Here, the partitioning factor, alpha is set to 1. So, this heuristic greedily assigns each node to the set that would maximize the number of cut edges if the node was moved.

2. **Simple Randomized Greedy Heuristic**: Initially, both partitions X and Y are empty. At each iteration, one vertex is placed to either set X or set Y with uniform randomness, i.e., with probability 0.5. The procedure terminates when all vertices are placed either in X or Y. Here, the partitioning factor, alpha is 0.
3. **Semi Greedy Heuristic:** It choose a randomized partitioning factor, α here. It uses value based method which place all candidates having greedy values better than α × best_value in restricted candidate list, RCL, where $\alpha \in [0,1]$). A vertex is randomly selected from RCL. Based on α , vertices are placed to either set X or set Y. The procedure terminates when all vertices are placed either in X or Y.
4. **Simple Local Search:** The Local Search algorithm iteratively explores the neighborhood of the current solution, making moves that improve the number of cut edges. It continues until no further improvement is possible within a predefined number of iterations. We are stuck in a local optima in this algo.
5. **GRASP (Greedy Randomized Adaptive Search Procedure) Technique:** GRASP combines the strengths of greedy algorithms with randomization. It's a randomized multistart iterative method, which tries to get closer to the global optima from a local optima by starting from multiple spaces.
   In my implementation, it repeatedly constructs solutions using a semi greedy phase and then refines them using local search.

Output Description:

From the benchmark data set, let's take the graph, "G1" to explain my output.
In, G1, there are 800 nodes and 19176 edges.
The outputs of different algorithms for this graph are,

1. Average Max Cut using Randomized Greedy Algo: 11024 (50 iterations)
2. Average Max Cut using Simple Greedy Heuristic: 11292
3. Average Max Cut using SemiGreedy Heuristic (value based method): 11184
4. Average number of Iterations in simple local search: 118
5. Average Max Cut of local search solutions: 11388
6. Number of iteration for GRASP: 50
7. Achieved Best Value of Max-cut after 50 iterations of GRASP: 11467

For this graph, Known best solution or upper bound is 12078.

The results generate the following decisions:
1. The Simple Greedy algorithm performed better than the Randomize Greedy algorithm, showcasing the effectiveness of a deterministic approach over randomness.
2. The Semi Greedy algorithm, combining both randomization and determinism, showed a consistent performance close to the Simple Greedy approach.
3. The Local Search algorithm's performance improvement over the other heuristics suggests that exploring the solution neighborhood yields significant enhancements in max cut value.
4. The GRASP technique achieved the best max cut value among all the heuristics as it takes the maximum of the output of all local searches. It proves that random multistart can improve the solution and helps us to get out of a local optima, though it can get stuck in another local optima. However, we still can't surely say that we have achieved the global optima.
5. The max cut value of the GRASP technique doesn't exceed the known best solution or upper bound. It proves that we are still stuck in a local optima, though we've improved a lot from other heuristics.

If we notice another 2 examples, G48 and G49, except Randomized, all other heuristics have been successful to reach the upper bound. Actually it's the global optima too. As the sum of the weights of all edges in these graphs are also 6000. So, it's not guaranteed whether we achieve the global optima or stuck in local optima, but we can achieve global optima in some special cases.

The Summary Table:

Problem			Constructive Algorithm			Local Search		GRASP		Known best solution or upper bound
						Simple Local or Local-1		GRASP-1		
Name	|V| or n	|E| or m	Random-ized-1	Greedy-1	Semi-greedy-1	No. of Iterations	Average Value	No. of Iterations	Best Value	
G1	800	19176	11024	11292	11184	118	11388	50	11467	12078
G2	800	19176	11042	11260	11150	114	11385	50	11498	12084
G3	800	19176	11018	11239	11155	115	11379	20	11451	12077

G4	800	19176	11023	11300	11204	101	11405	20	11458	
G5	800	19176	11024	11265	11131	120	11377	20	11432	
G6	800	19176	1543	1846	1671	120	1919	20	2009	
G7	800	19176	1390	1587	1479	137	1764	20	1827	
G8	800	19176	1363	1612	1466	139	1751	20	1829	
G9	800	19176	1420	1678	1520	137	1796	20	1880	
G10	800	19176	1384	1568	1457	135	1737	20	1816	
G11	800	1600	408	482	440	11	461	20	494	627
G12	800	1600	398	486	433	10	452	20	478	621
G13	800	1600	420	496	447	12	470	20	508	645
G14	800	4694	2871	2949	2942	24	2972	20	2985	3187
G15	800	4661	2846	2948	2909	26	2943	20	2958	3169
G16	800	4672	2856	2915	2917	27	2953	20	2975	3172
G17	800	4667	2850	2925	2911	29	2947	20	2970	
G18	800	4694	703	850	756	56	848	20	911	
G19	800	4661	617	781	663	60	762	20	809	
G20	800	4672	652	775	698	58	793	20	868	
G21	800	4667	635	795	694	61	793	20	847	
G22	2000	19990	12362	12780	12659	141	12904	20	12992	14123
G23	2000	19990	12364	12852	12634	147	12893	20	12961	14129
G24	2000	19990	12362	12777	12614	162	12896	20	12964	14131
G25	2000	19990	12361	12900	12637	156	12903	20	13039	
G26	2000	19990	12350	12793	12694	126	12909	20	12970	
G27	2000	19990	2297	2704	2527	192	2869	20	2963	
G28	2000	19990	2278	2700	2480	197	2824	20	2898	
G29	2000	19990	2377	2735	2610	184	2937	20	3085	
G30	2000	19990	2388	2750	2583	188	2922	20	3018	
G31	2000	19990	2312	2685	2447	222	2843	20	2923	
G32	2000	4000	1018	1188	1091	27	1146	20	1216	1560
G33	2000	4000	979	1170	1061	29	1120	20	1216	1537
G34	2000	4000	970	1172	1033	35	1103	20	1182	1541
G35	2000	11778	7199	7391	7369	56	7442	20	7467	8000
G36	2000	11766	7175	7415	7357	61	7433	20	7459	7996
G37	2000	11785	7189	7387	7341	70	7432	20	7464	8009
G38	2000	11779	7180	7374	7374	59	7447	20	7469	
G39	2000	11778	1688	2014	1899	119	2091	20	2152	
G40	2000	11766	1667	2048	1714	173	2010	20	2124	
G41	2000	11785	1667	2054	1720	168	2006	20	2091	
G42	2000	11779	1726	2082	1925	132	2139	20	2225	
G43	1000	9990	6177	6387	6279	85	6427	20	6507	7027
G44	1000	9990	6181	6399	6298	84	6439	20	6493	7022
G45	1000	9990	6167	6364	6263	92	6427	20	6507	7020
G46	1000	9990	6173	6397	6330	65	6442	20	6506	
G47	1000	9990	6166	6366	6326	72	6449	20	6503	
G48	3000	6000	4914	6000	6000	1	6000	20	6000	6000

G49	3000	6000	4906	6000	6000	1	6000	20	6000	6000
G50	3000	6000	4905	5880	5842	3	5846	20	5880	5988
G51	1000	5909	3600	3706	3681	35	3726	20	3748	
G52	1000	5916	3604	3699	3672	41	3724	20	3756	
G53	1000	5914	3613	3704	3685	32	3726	20	3738	
G54	1000	5916	3599	3694	3680	31	3721	20	3744	

Summary:

In this experimental comparison of max-cut heuristic algorithms, we found that the GRASP technique outperformed the others in obtaining the highest max cut value for the given benchmark data sets. An effective strategy for solving the Max-Cut problem was found to be the combination of greedy randomized construction and adaptive local search refinement. The best heuristic for a given problem instance can be chosen to get a better solution in real life, but the trade-offs between algorithmic complexity and performance should be carefully taken into account.