Report

Name : Ajoy Dey

Student ID : 1905038

Subject : **Solving the Max-cut problem by GRASP**

Date : 20/8/23

The summary table:

Problem				Constructive Algorithm			Local Search						Grasp		Known Best Solution (upper bound)
Name	No. of vertex	No. of edges	Weight_Range	Randomized-1	Greedy-1	Semi Greedy	random-local-search		greedy-local-search		semi-greedy-local-search		Grasp-1		
							No. of iteration	Best_value	No. of iteration	Best_value	No. of iteration	Best_value	No. of iteration	Best_value	
G1	800	19176	1 to 1	9580	11245	11134	625	11374	70	11372	119	11370	50	11453	12078
G2	800	19176	1 to 1	9591	11233	11128	616	11365	89	11383	124	11379	50	11467	12084
G3	800	19176	1 to 1	9591	11252	11140	623	11369	67	11377	115	11372	50	11460	12077
G11	800	1600	-1 to 1	10	496	431	183	424	5	504	13	456	50	490	627
G12	800	1600	-1 to 1	-4	476	413	188	418	3	480	13	439	50	484	621
G13	800	1600	-1 to 1	15	520	439	187	438	3	524	15	468	50	500	645
G14	800	4694	1 to 1	2347	2964	2926	267	2921	6	2971	28	2963	50	3000	3187
G15	800	4661	1 to 1	2326	2931	2904	272	2905	14	2951	29	2943	50	2965	3169
G16	800	4672	1 to 1	2340	2942	2910	264	2906	15	2957	29	2947	50	2970	3172
G22	2000	19990	1 to 1	10000	12817	12590	1139	12816	104	12995	173	12889	50	12997	14123
G23	2000	19990	1 to 1	9984	12807	12590	1139	12818	102	12989	171	12887	50	13005	14129
G24	2000	19990	1 to 1	10003	12864	12554	1132	12813	80	12997	182	12876	50	13009	14131
G32	2000	4000	-1 to 1	13	1198	1068	460	1053	10	1216	30	1130	50	1228	1560
G33	2000	4000	-1 to 1	-18	1216	1050	467	1033	12	1238	30	1111	50	1220	1537
G34	2000	4000	-1 to 1	-21	1214	1047	467	1031	8	1228	31	1110	50	1208	1541
G35	2000	11778	1 to 1	5895	7391	7336	662	7338	34	7433	71	7428	50	7479	8000
G36	2000	11766	1 to 1	5891	7409	7334	668	7314	38	7452	68	7422	50	7458	7996
G37	2000	11785	1 to 1	5890	7413	7349	666	7325	26	7441	67	7434	50	7481	8009
G43	1000	9990	1 to 1	4999	6359	6278	569	6395	52	6458	86	6427	50	6498	7027
G44	1000	9990	1 to 1	4996	6367	6280	557	6391	37	6426	88	6432	50	6506	7022
G45	1000	9990	1 to 1	4989	6417	6284	573	6401	38	6481	80	6425	50	6488	7020
G48	3000	6000	1 to 1	3000	6000	5403	836	5018	1	6000	30	5469	50	5846	6000
G49	3000	6000	1 to 1	3003	6000	5459	834	5013	1	6000	23	5511	50	5934	6000
G50	3000	6000	1 to 1	2988	5880	5397	834	5002	1	5880	29	5464	50	5812	5988

The constructive Algorithms:

In this MAX-CUT problem, I use 3 constructive algorithms to solve. They are Simple Randomized, Simple Greedy , Semi – Greedy heuristic .

In greedy heuristic function , firstly largest-weight edge is selected and put its end points in two set. 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. I call this function only once because this function generate same solution for same graph.

In Simple Randomized heuristic function , iterate through all vertex and place each in partition X or Y with uniform randomness (probability ½) . I call this function for 50 times and calculate the average max cut value.

In Semi Greedy heuristic function , select the edges for a restricted candidate list (RCL) comparing their weight with a cut-off value , then select one edge randomly from RCL and put its end points in two partitions. Each vertex v, define $\sigma_X(v) = \Sigma_{u \in Y} W_{vu}$ and $\sigma_Y(v) = \Sigma_{u \in X} W_{vu}$, the greedy function value of v = max { $\sigma_X(v)$, $\sigma_Y(v)$} . In this function , for the remaining vertices, select the vertices for a restricted candidate list (RCL) comparing their greedy function value with a cut-off value , then select one vertex randomly from RCL and put it in any partition based on its max { $\sigma_X(v)$, $\sigma_Y(v)$}.

In this table , we can see that random construction is high variance , low quality and almost always sub-optimal, where the greedy construction is low variance , good quality and usually sub-optimal. As we always select from well-ranked elements of RCL at random and add it to our solution in semi-greedy heuristic, this heuristic tries to get around convergence to non-global local maxima.

Local Search:

I run 50 iteration and for each iteration, I call Simple Randomized heuristic function, semi-greedy heuristic function separately and get a solution on each time and call Local Search function for each solution and calculate the average max cut value and the average number of iterations.

I call only greedy heuristic function once and call local search function for this solution.

After calling local search function , we get the best local optimal solution of given solution. In this table, we see that we get best local maximum value.

In this table , we can see that local search from random starting solution is higher variance and slower convergence than other two heuristics . That's why , the average number of iteration for local search is more than other two heuristics.

Grasp:

GRASP (greedy randomized adaptive search procedure) is a randomized multi start iterative method. Each GRASP iteration is usually made up of a construction phase, where a feasible solution is constructed, and a local search
phase which starts at the constructed solution and applies iterative improvement until a locally optimal solution is found. After all iterations are completed, we get best local optimal solution which may be global optimal solution or not .

I set max iteration to 50 and choose semi greedy heuristic for construction phase. In this table we can see that GRASP tries to capture good features of greedy & random constructions.