Maximum Cut using GRASP Algorithm

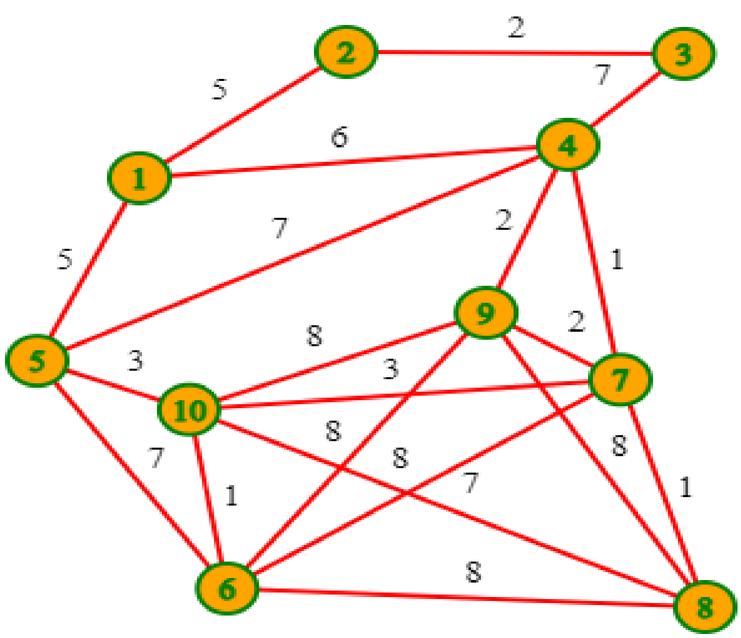
Introduction

For a graph, a maximum cut is a cut whose size is at least the size of any other cut. That is, it is a partition of the graph's vertices into two complementary sets *S* and *T*, such that the number of edges between *S* and *T* is as large as possible. Finding such a cut is known as the **max-cut problem**.

GRASP is Greedy Randomized Adaptive Search Procedures. A GRASP for the MAX-CUT problem consists of repeatedly constructing a cut (S, $\underline{S}$) with a semi-greedy algorithm, applying local search from (S, $\underline{S}$) to produce a locally maximal solution.

Input Graph

We conducted the algorithm using several varying parameters upon the following undirected weighted graph –



Input:

10 20

6 9 8

4 7 1

9 10 8

5 6 7

4 9 2

1 2 5

8 9 8

6 8 8

7 9 2

6 10 1

1 4 6

2 3 2

4 5 7

1 5 5

3 4 7

7 8 1

6 7 8

7 10 3

8 10 7

5 10 3

Output (Selected one Randomly)

Set-1: 2 4 6 9 10

Set-2: 1 3 5 7 8

Total Cut Value: 82 (taken from row-1 in the following table)

Statistics:

Alpha	Type	Depth	S1(After Semi-Gree	S2(After Semi-	S1	S2	Itr	Total	Time
0	1	3	(2 4 6 9 10)	(1 3 5 7 8)	(2 4 6 10)	(1 3 5 7 8 9	2	82	0
0	1	7	(2 4 6 9 10)	(1 3 5 7 8)	(2 4 6 10)	(1 3 5 7 8 9	2	82	0
0	2	3	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
0	2	7	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
0	3	3	(1 3 6 9)	(2 4 5 7 8 10)	(1 3 6 9 10)	(2 4 5 7 8)	4	79	0
0	3	7	(2 4 5 6 8 9)	(1 3 7 10)	(2 4 5 6 8 9)	(1 3 7 10)	8	82	0
0.1	1	3	(2 4 5 7 8)	(1 3 6 9 10)	(2 4 5 7 8 9)	(1 3 6 10)	2	77	0
0.1	1	7	(1 3 6 9 10)	(2 4 5 7 8)	(1 3 6 10)	(2 4 5 7 8 9	2	77	0
0.1	2	3	(2 4 6 7 8)	(1 3 5 9 10)	(2 4 5 7 8 9)	(1 3 6 10)	4	77	0
0.1	2	7	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
0.1	3	3	(1 3 5 7 8 9 10)	(2 4 6)	(1 3 5 7 8 9 10	(2 4 6)	4	76	0
0.1	3	7	(1 3 5 7 9)	(2 4 6 8 10)	(1 3 5 7 9)	(2 4 6 8 10)	8	76	0
0.2	1	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	1	82	16
0.2	1	7	(2 4 6 9)	(1 3 5 7 8 10)	(2 4 6 10)	(1 3 5 7 8 9	3	82	0
0.2	2	3	(2 3 5 7 9 10)	(1 4 6 8)	(2 3 5 7 8 9)	(1 4 6 10)	3	79	3.2
0.2	2	7	(2 4 6 8 10)	(1 3 5 7 9)	(2 4 6 10)	(1 3 5 7 8 9	2	82	1.1
0.2	3	3	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	4	82	1
0.2	3	7	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9	8	82	1.1
0.3	1	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	1	82	1.1
0.3	1	7	(2 4 6 7 8 10)	(1 3 5 9)	(2 4 5 7 8 9)	(1 3 6 10)	5	77	0
0.3	2	3	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9	1	79	0
0.3	2	7	(1 3 5 6 8 9)	(2 4 7 10)	(1 3 5 7 8 9)	(2 4 6 10)	5	82	1.1
0.3	3	3	(2 3 4 6 7 8 9)	(1 5 10)	(2 3 4 6 7 8 9)	(1 5 10)	4	74	1.1
0.3	3	7	(2 4 7 8 10)	(1 3 5 6 9)	(2 4 7 8 10)	(1 3 5 6 9)	8	76	1.1
0.4	1	3	(2 4 5 7 8 10)	(1 3 6 9)	(2 4 5 7 8 9)	(1 3 6 10)	3	77	1.1
0.4	1	7	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	1.1
0.4	2	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	1	82	1
0.4	2	7	(1 3 6 9 10)	(2 4 5 7 8)	(1 3 6 10)	(2 4 5 7 8 9	2	77	1.1
0.4	3	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	4	82	1.1
0.4	3	7	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	8	82	1.1

0.5	1	3	(2 3 5 7 8 10)	(1 4 6 9)	(2 3 5 7 8 9)	(1 4 6 10)	3	79	0
0.5	1	7	(2 4 6 9)	(1 3 5 7 8 10)	(2 4 6 10)	(1 3 5 7 8 9	3	82	1.1
0.5	2	3	(2 3 4 6 8 10)	(1 5 7 9)	(2 4 6 10)	(1 3 5 7 8 9	3	82	0
0.5	2	7	(2 3 5 7 8 10)	(1 4 6 9)	(2 3 5 7 8 9)	(1 4 6 10)	3	79	0
0.5	3	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	4	82	0
0.5	3	7	(2 4 6 8 10)	(1 3 5 7 9)	(2 4 6 8 10)	(1 3 5 7 9)	8	76	0
0.6	1	3	(2 4 6 9 10)	(1 3 5 7 8)	(2 4 6 10)	(1 3 5 7 8 9	2	82	0
0.6	1	7	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	1	82	0
0.6	2	3	(2 4 5 7 8 9)	(1 3 6 10)	(2 4 5 7 8 9)	(1 3 6 10)	1	77	0
0.6	2	7	(2 4 6 8 10)	(1 3 5 7 9)	(2 4 6 10)	(1 3 5 7 8 9	2	82	0
0.6	3	3	(1 3 5 7 9 10)	(2 4 6 8)	(1 3 5 7 9 10)	(2 4 6 8)	4	82	0
0.6	3	7	(2 4 6 9 10)	(1 3 5 7 8)	(2 4 6 9 10)	(1 3 5 7 8)	8	76	0
0.7	1	3	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
0.7	1	7	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9	1	82	0
0.7	2	3	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
0.7	2	7	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
0.7	3	3	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	4	82	0
0.7	3	7	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9	8	82	0
0.8	1	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	1	82	0
0.8	1	7	(1 4 6 9 10)	(2 3 5 7 8)	(1 4 6 10)	(2 3 5 7 8 9	2	79	0
0.8	2	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	1	82	0
0.8	2	7	(2 4 6 8 10)	(1 3 5 7 9)	(2 4 6 10)	(1 3 5 7 8 9	2	82	0
0.8	3	3	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	4	82	0
0.8	3	7	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	8	82	0
0.9	1	3	(1 3 5 7 9)	(2 4 6 8 10)	(1 3 5 7 8 9)	(2 4 6 10)	2	82	0
0.9	1	7	(1 3 6 9 10)	(2 4 5 7 8)	(1 3 6 10)	(2 4 5 7 8 9	2	77	0
0.9	2	3	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
0.9	2	7	(2 4 6 8 10)	(1 3 5 7 9)	(2 4 6 10)	(1 3 5 7 8 9	2	82	0
0.9	3	3	(1 3 5 7 8 9)	(2 4 6 10)	(1 3 5 7 8 9)	(2 4 6 10)	4	82	0
0.9	3	7	(2 4 6 7 8 10)	(1 3 5 9)	(2 4 6 7 8 10)	(1 3 5 9)	8	82	16
1	1	3	(2 3 4 6 8 10)	(1 5 7 9)	(2 4 6 10)	(1 3 5 7 8 9	3	82	0
1	1	7	(2 3 4 6 8 10)	(1 5 7 9)	(2 4 6 10)	(1 3 5 7 8 9	3	82	0
1	2	3	(2 4 5 7 8 9)	(1 3 6 10)	(2 4 5 7 8 9)	(1 3 6 10)	1	77	0
1	2	7	(2 3 5 7 8 9)	(1 4 6 10)	(2 3 5 7 8 9)	(1 4 6 10)	1	79	0
1	3	3	(2 4 5 7 8 9)	(1 3 6 10)	(2 4 5 7 8 9)	(1 3 6 10)	4	82	0
1	3	7	(1 4 6 8 10)	(2 3 5 7 9)	(1 4 6 8 10)	(2 3 5 7 9)	8	76	0

Here,

Alpha means the fraction towards the highest weight value from the lowest one. Alpha ranges between 0 to 1. We increased alpha by 0.1 per iteration for inspecting the outputs.

Type is chosen to denote one of the following choices-

1. Best first (neighbor's value > my value)

2. Best first with equality (neighbor's value >= my value)

3. Select any neighbor who has the highest delta value.

To emphasize the effectiveness of the local search optimization over semi-greedy approach, the conditions of the two sets S1 and S2 are presented in detail. Total time taken and the number of calls to function local_search() are also shown. Finally, the total cut value also appears in the Total field.

Discussion (For the single graph):

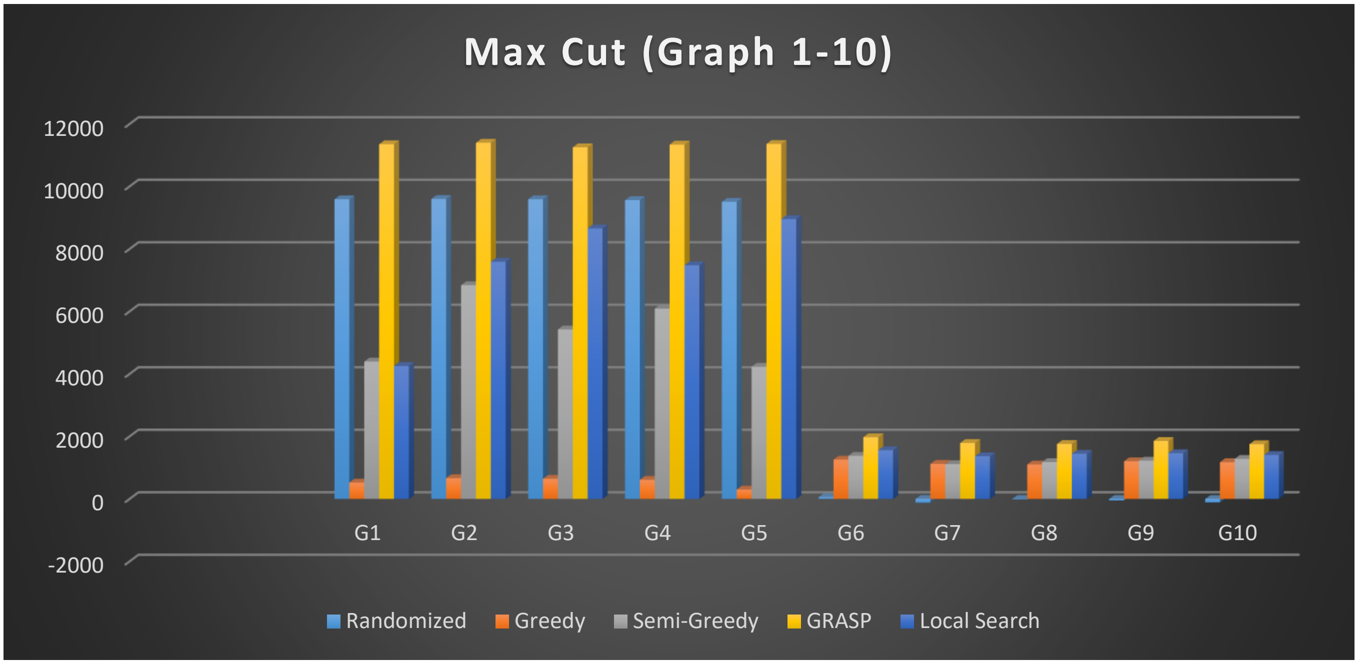
The semi-greedy method is used to add some randomness to the search procedure. The local search method tries to update the final state by bringing some 'small-hop' changes. Sometimes the local search fails in the trap region and random restart is required. In our case, when we were trying to move to any neighbor with best delta possible, no bound is imposed and so it infinitely runs and that is not practical. So we added maximum number of iterations to forcefully stop the search procedure.

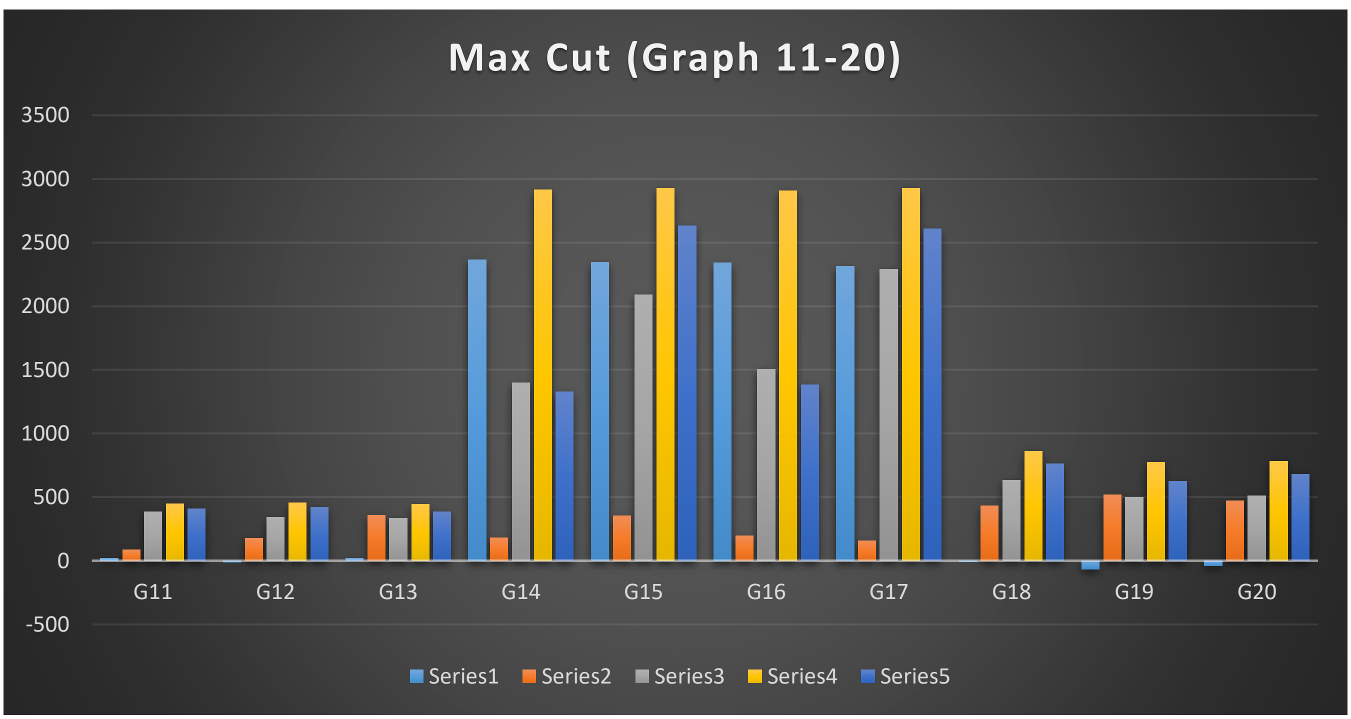
Benchmark Dataset:

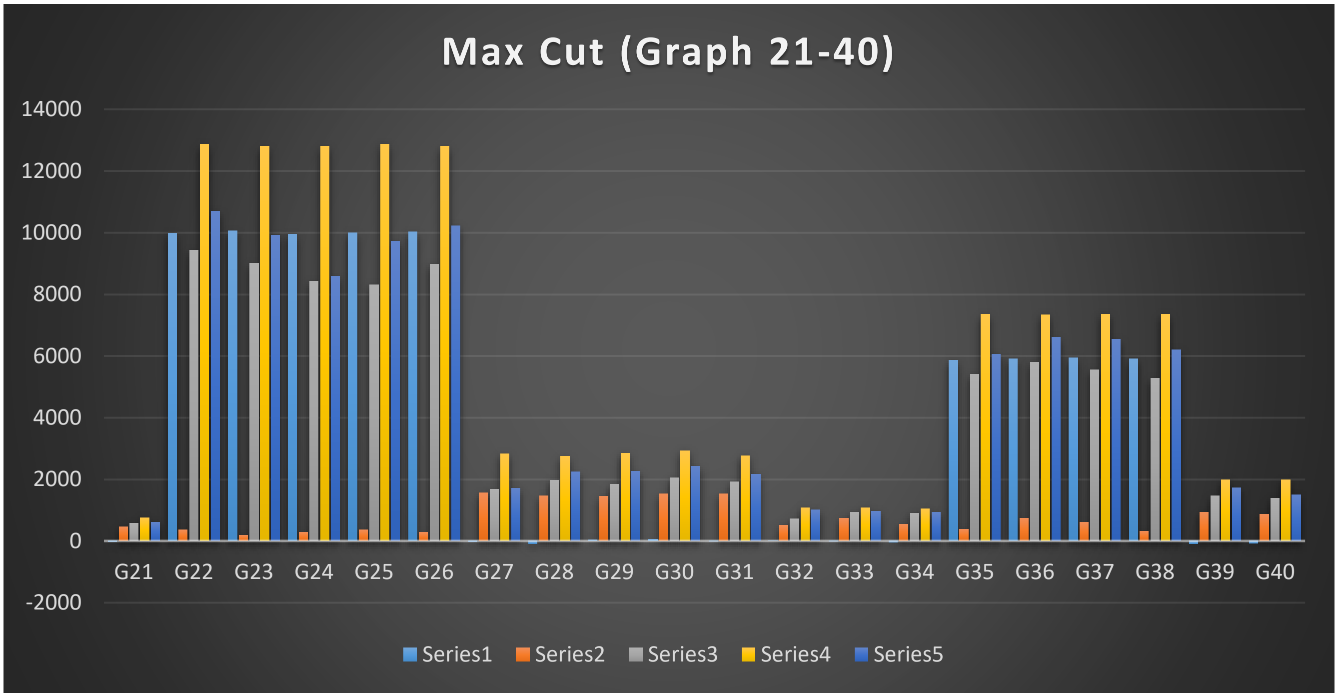
Problem			Constructive Algorithm			Local Search		GRASP		
Name	|V| or n	|E| or m	Randomized	Greedy	Semi-Greedy	No of itr	Cut val	No of itr	Cut val	Best val
G1	800	19176	9592	524	4396	284	4257	844	11353	12078
G2	800	19176	9603	667	6841	243	7591	720	11399	12084
G3	800	19176	9593	645	5427	203	8660	600	11255	12077
G4	800	19176	9566	613	6092	236	7479	700	11340	N/A
G5	800	19176	9509	301	4233	193	8958	570	11361	N/A
G6	800	19176	58	1259	1380	83	1560	241	1976	N/A
G7	800	19176	-123	1124	1113	96	1368	279	1791	N/A
G8	800	19176	-14	1102	1180	75	1443	216	1759	N/A
G9	800	19176	-50	1206	1226	100	1463	290	1855	N/A
G10	800	19176	-112	1178	1282	76	1405	218	1751	N/A
G11	800	1600	21	86	386	14	410	32	448	627
G12	800	1600	-8	179	342	13	420	30	456	621
G13	800	1600	21	360	336	18	385	46	442	645
G14	800	4694	2365	181	1398	134	1329	394	2916	3187
G15	800	4661	2345	352	2092	65	2633	186	2926	3169
G16	800	4672	2340	196	1505	131	1384	383	2908	3172
G17	800	4667	2314	158	2290	67	2607	191	2927	N/A
G18	800	4694	-6	433	633	36	763	98	864	N/A
G19	800	4661	-67	518	501	47	627	133	774	N/A
G20	800	4672	-39	474	512	35	681	97	782	N/A
G21	800	4667	-20	466	589	48	621	134	764	N/A
G22	2000	19990	9985	365	9437	321	10700	953	12858	14123
G23	2000	19990	10062	188	9014	366	9922	1090	12796	14129
G24	2000	19990	9957	299	8430	455	8596	1355	12801	14131
G25	2000	19990	10002	370	8319	413	9724	1230	12877	N/A
G26	2000	19990	10030	288	8977	336	10230	999	12799	N/A
G27	2000	19990	-28	1562	1674	234	1712	692	2834	N/A

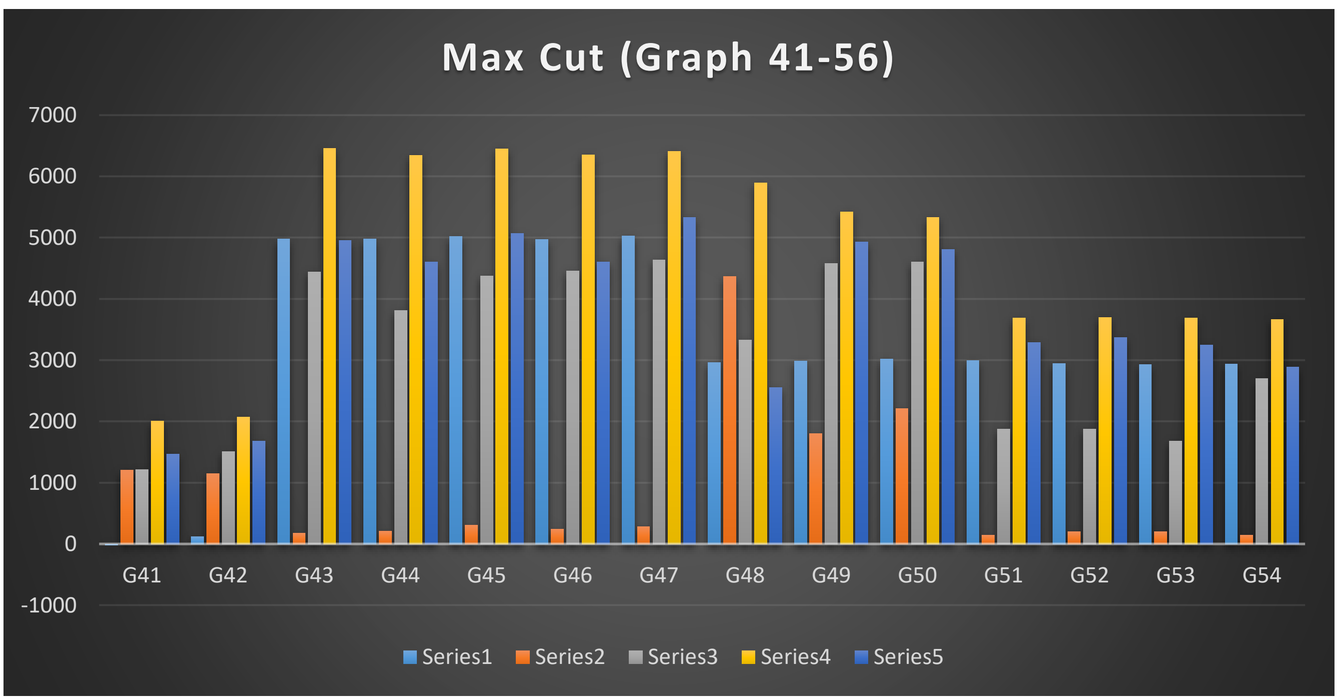
G28	2000	19990	-95	1478	1980	120	2248	351	2747	N/A
G29	2000	19990	40	1455	1845	157	2263	463	2846	N/A
G30	2000	19990	59	1529	2043	130	2432	381	2928	N/A
G31	2000	19990	-2	1525	1921	151	2168	445	2776	N/A
G32	2000	4000	16	513	730	22	1014	58	1088	1560
G33	2000	4000	-4	736	936	31	973	85	1084	1537
G34	2000	4000	-41	550	907	32	923	87	1038	1541
G35	2000	11778	5863	390	5411	193	6062	570	7348	8000
G36	2000	11766	5908	736	5807	152	6605	447	7338	7996
G37	2000	11785	5955	605	5551	156	6542	458	7363	8009
G38	2000	11779	5920	327	5275	196	6212	580	7351	N/A
G39	2000	11778	-85	933	1478	80	1733	230	1992	N/A
G40	2000	11766	-73	869	1399	129	1495	379	1991	N/A
G41	2000	11785	-20	1201	1216	146	1468	430	2005	N/A
G42	2000	11779	119	1152	1504	108	1685	316	2073	N/A
G43	1000	9990	4975	180	4445	193	4955	571	6454	7027
G44	1000	9990	4982	212	3811	202	4598	597	6344	7022
G45	1000	9990	5018	307	4377	194	5065	574	6447	7020
G46	1000	9990	4968	248	4458	209	4607	617	6351	N/A
G47	1000	9990	5028	282	4634	158	5329	465	6406	N/A
G48	3000	6000	2960	4366	3328	456	2554	1359	5888	6000
G49	3000	6000	2986	1804	4579	92	4931	268	5420	6000
G50	3000	6000	3020	2209	4601	103	4809	300	5330	5988
G51	1000	5909	2996	145	1879	85	3285	245	3685	N/A
G52	1000	5916	2943	203	1879	74	3369	214	3699	N/A
G53	1000	5914	2932	206	1685	80	3250	232	3684	N/A
G54	1000	5916	2936	148	2695	105	2884	307	3666	N/A

Graph:


Max Cut (Graph 1-10)
12000
10000
8000
6000
4000
2000
0
-2000
G1
G2
G3
G4
G5
G6
G7
G8
G9
G10
Randomized
Greedy
Semi-Greedy
GRASP
Local Search


Max Cut (Graph 11-20)
3500
3000
2500
2000
1500
1000
500
0
-500
G11
G12
G13
G14
G15
G16
G17
G18
G19
G20
Series1
Series2
Series3
Series4
Series5


Max Cut (Graph 21-40)
14000
12000
10000
8000
6000
4000
2000
0
-2000
G21 G22 G23 G24 G25 G26 G27 G28 G29 G30 G31 G32 G33 G34 G35 G36 G37 G38 G39 G40
Series1 Series2 Series3 Series4 Series5


Max Cut (Graph 41-56)
7000
6000
5000
4000
3000
2000
1000
0
-1000
G41 G42 G43 G44 G45 G46 G47 G48 G49 G50 G51 G52 G53 G54
Series1 Series2 Series3 Series4 Series5

Discussion (for Dataset):

It is evident that GRASP performs the best for all the cases. Greedy performs even worse than randomized algorithm in most of the cases. Semi greedy and randomized perform almost similar in most of the cases.

Faria Binta Awal

1905012

Level - 3, Term – II