Investigating the Effect of Different Construction Algorithm for Local Search for Solving Max Cut Problem

Zarif Ikram
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka
zarifikram003@gmail.com

1 Introduction

Local search has been used for solving computationally hard optimization problem. It is a heuristic method which finds optimal solution around a specific solution's neighbors. Max cut Sivaramakrishnan and Karimi (2007) is an NP-Hard problem, and thus no optimal polynomially obtainable solutions exist. Hence, local search is an appropriate candidate to solve this problem. A construction algorithm is one that builds the initial solution that follows a local search. There are numerous construction algorithm such as random selection, greedy, and semi-greedy. Notably, Greedy Randomized Adaptive Search Procedure (GRASP) Feo and Resende (1995) is one such algorithm in which a semi-greedy initial solution follows a local search procedure. In this study, we contribute the following.

- Understand the affect of different construction algorithms to the quality of solution.
- Analyze the affect of iteration number to the quality of solution.

2 Methodology

Given an undirected graph $G(V, U)$ where V is the set of vertices, U is the set of edges, and weights w_{uv} associated with each edge $(u, v) \in U$, a max-cut problem consists of finding non-empty subsets of vertices $S \subset V (S \neq \emptyset)$, such that weight of the cut $(S, \bar{S})$, given by $\sum_{u \in S, v \in \bar{S}} w_{uv}$ is maximized. In a local search algorithm we start by finding an initial solution (constructive phase) and improve it by visiting its neighbors. We iteratively improve the final solution by varying the initial solution. Hence different construction algorithm may affect the quality of the final solution depending on the utilization and exploration of the state space.

We use three construction algorithms for local search. These are the following.

1. Simple Randomized
2. Simple Greedy
3. Randomized Greedy

2.1 Simple Randomized

In this method, we pick $v \sim V$ uniformly and sequentially update S and $\bar{S}$.

2.2 Simple Greedy

In this method, we use greedy method to choose vertices. Initially, we pick (u, v) such w_{uv} is the maximum and append u and v to S and $\bar{S}$, respectively. Next, we greedily pick $v \in V$ so that adding v to S or $\bar{S}$ maximizes the cut.

Preprint. Under review.

2.3 Randomized Greedy

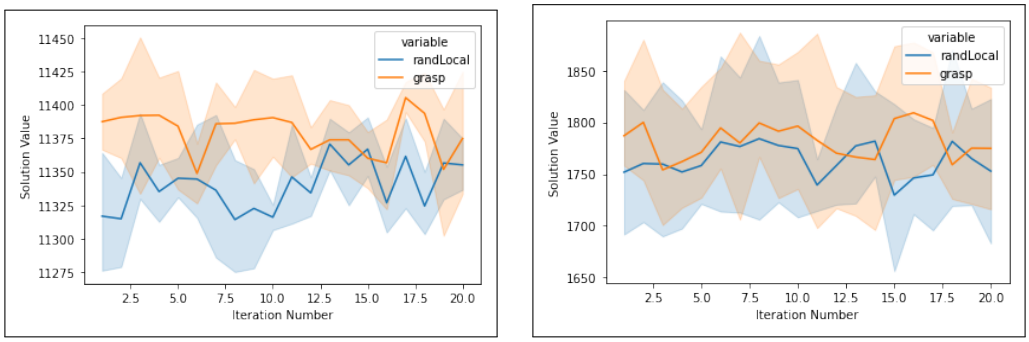
Randomized greedy Hart and Shogan (1987) is similar to simple greedy, but instead of picking the max weighted edge initially, we pick a random edge from the αth percentile max weighted edges (Restricted Candidate List(RCL)) where α is sampled uniformly. Subsequently, similar to simple greedy, we iteratively the pick the rest of the vertices except, like before, instead of pick the best vertex, we pick a random vertex from RCL.

3 Results

To investigate the initial solution of quality of different constructive algorithms, we run the inputs provided to us. There were total 54 input graphs. To mitigate the associated with simple randomized and randomized greedy algorithms, we run the algorithm for 20 iterations and report the mean max cut found. Table 1 reports the findings. It is evident that simple greedy outperforms other two constructive algorithms in terms of solution quality. The solutions provided by simple greedy, while not completely equal to upper bound, are quite close to it. Notably, simple randomized performs poorly in some cases, especially for negative weighted graphs. It is due to the random selection of vertices that the vertices are not separated into different cuts when they have negative weights among themselves. Randomized greedy is comparatively more stable than simple randomized despite being a randomized algorithm. It performs similar to simple greedy despite not outperforming it.

To investigate the effect of constructive algorithms with local search, we run the local search algorithm with the constructive algorithms mentioned above. We refrain from using simple greedy constructive algorithm with the local search as it provides a fixed solution and is not suitable method for local search. Hence, we provide results for (a) Local search with randomized start, and (b) Local search with randomized greedy (GRASP).

Furthermore, we showcase the effect of iteration number on solution quality in figure 1. While more iteration does increase the likelihood of a better solution, it is not fixed as we use uniform sampling for both cases. As discussed before, it is evident that GRASP performs better and provides a better solution than local search with randomized start. However, as expected max-cut value decreases (Figure 1b), the difference between two algorithms start to diminish.



(a) (b)

Figure 1: Comparison of local search with random start and local search with randomized greedy (GRASP) for (a) Graph G1-G5 (b) G6-G10

.

4 Conclusion

In this work, we provide an analysis of different local search algorithms in the context of the max-cut problem. We first provide a through analysis of different construction algorithms by running them on 54 provided graphs. Then, we compare the performance of simple randomized local search and semi-greedy local search (GRASP) for max-cut problem. Finally, we showcase the effect of iteration numbers on different local search algorithms. We find simple greedy algorithm to be better performing when local search is not used and GRASP to be a better algorithm overall.

Table 1: Comparison of initial solution to upper bound. Upper bounds not provided are kept blank.

Problem			Construction Algorithm			Upper Bound
Name	$\|V\|$	$\|U\|$	Simple Randomized	Simple Greedy	Randomized Greedy	
G1	800	19176	9602	11267	11155	12078
G2	800	19176	9640	11244	11087	12084
G3	800	19176	9567	11334	11138	12077
G4	800	19176	9541	11279	11162	
G5	800	19176	9670	11311	11141	
G6	800	19176	34	1790	1622	
G7	800	19176	-42	1589	1493	
G8	800	19176	-233	1578	1485	
G9	800	19176	47	1727	1535	
G10	800	19176	-114	1665	1472	
G11	800	1600	2	472	440	627
G12	800	1600	-30	484	432	621
G13	800	1600	-2	510	439	645
G14	800	4694	2368	2950	2935	3187
G15	800	4661	2289	2930	2913	3169
G16	800	4672	2382	2918	2911	3172
G17	800	4667	2358	2930	2910	
G18	800	4694	24	855	772	
G19	800	4661	-11	750	667	
G20	800	4672	-34	759	726	
G21	800	4667	82	780	706	
G22	2000	19990	10075	12795	12637	14123
G23	2000	19990	9982	12799	12662	14129
G24	2000	19990	10005	12761	12650	14131
G25	2000	19990	10051	12846	12618	
G26	2000	19990	10011	12747	12570	
G27	2000	19990	79	2693	2453	
G28	2000	19990	-68	2632	2394	
G29	2000	19990	137	2745	2534	
G30	2000	19990	1	2786	2503	
G31	2000	19990	-13	2645	2422	
G32	2000	4000	-20	1198	1080	1560
G33	2000	4000	-18	1190	1076	1537
G34	2000	4000	-10	1190	1043	1541
G35	2000	11778	5879	7364	7333	8000
G36	2000	11766	5807	7381	7353	7996
G37	2000	11785	5823	7378	7364	8009
G38	2000	11779	5859	7365	7356	
G39	2000	11778	93	2022	1748	
G40	2000	11766	-77	2019	1852	
G41	2000	11785	-135	1988	1718	
G42	2000	11779	85	2056	1909	
G43	1000	9990	5014	6347	6266	7027
G44	1000	9990	5020	6377	6289	7022
G45	1000	9990	5063	6363	6323	7020
G46	1000	9990	4964	6420	6308	
G47	1000	9990	4993	6383	6289	
G48	3000	6000	3000	6000	6000	6000
G49	3000	6000	3000	6000	6000	6000
G50	3000	6000	3000	5880	5866	5988
G51	1000	5909	2925	3657	3677	
G52	1000	5916	2957	3683	3690	
G53	1000	5914	2954	3692	3679	
G54	1000	5916	2909	3686	3683	

References

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Hart, J. P. and Shogan, A. W. (1987). Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114.

Sivaramakrishnan, K. K. and Karimi, S. (2007). Max-cut problem.