Report on using GRASP to find max cut of a graph

Name: Mayesha Rashid **Student ID:** 1905103

Working Procedure:

At first I have used three different approaches to construct two sets of vertices between whom the total edge weight will be maximized. These approaches are: randomized, greedy and semigreedy.

Randomized:

We have randomly put the vertices in either of the sets with ½ probability and then calculated the max cut. We have done this procedure 10 times and taken the average of the cuts.

Greedy:

We have first identified one of the edges with maximum weight and put their two end points in different sets, that way this edge will contribute to our max cut. After that, We have checked the vertices which are adjacent to either one of the vertices present in our current sets and chose the vertex which has the highest sum of weights with one of the sets. Then we include the vertex to the opposite set and continue this process until all the vertices have been assigned to one of the sets. The edges which connect these two sets will contribute to the max cut.

Semigreedy:

We have used value based method to construct a restricted candidate list(RCL) from all the edges, in this case I have randomly generated a real value α $[0 \leq \alpha \leq 1]$ and multiplied it to the max edge cost. The edges which have greater than equals to weight than this value are included in RCL. After that, a random edge from RCL is selected and its two endpoints are placed in different sets. Then we follow the following procedure until all vertices have been assigned to one of the two sets:

The candidate elements are vertices which are not yet assigned to set X and set Y. For each such vertex, we have calculated the cost it will add to max cut if it is added to set X or set Y. Then we have again generated a random cut off value which is α times the max possible contribution of a vertex. Then we assign the vertices which fulfills this cut off value to another RCL. Then a random vertex is chosen from this list and added to the set which will maximize the value of max cut.

As the results will differ based on the value of α, we have done this procedure 10 times and recorded the average value of max cuts.

Local Search:

After the construction phase, local search has been implemented where we try switching one vertex from one set to another and whether it improves the value of max cut or not. We continue this process until a situation arises that max cut reaches its maxima, then we terminate our process and record the max cut.

In GRASP, we select the number of max iterations and in each iteration we generate a semi greedy solution, then local search on it to improve the max cut. The max value generated among all the iterations is our desired result.

Performance Analysis:

We have run the 54 given test cases and some of the findings are:

- In the case of construction, the greedy algorithm gave a more improved max cut than the semi greedy algorithm in 90% cases, and both of them gave better performance than the randomized algorithm.

- In the case of a graph consisting of negative edges, the randomized algorithm performed very poorly and often returned a negative value of max cut.

- In case of greedy construction, the returned max cut was ≥ 65% of known upper bound on average.

- In case of semi greedy construction, the returned max cut was ≥ 60% of known upper bound on average.

- Randomized greedy construction required more iterations than semigreedy and greedy to reach an optimal state using local search.

- After applying GRASP, semi greedily constructed solutions improved more than greedily constructed solutions.

- GRASP max values after applying 100 iterations were found to be ≥ 85% of known upper bound on average. This proves that our implementation of GRASP has the ability to improve max cut after construction using different approaches.