Max Cut Problem Using GRASP

The code implements three constructive algorithms:

Randomized – 1:

This algorithm works like a coin toss algorithm. We choose a vertex, and assign it to a set X or Y. The work is done randomly, and the vertex has the same probability to go to any of the sets. After assigning all vertices to a set, the max cut is calculated.

Greedy – 1:

This algorithm first chooses the largest weighted edge in the graph, and assigns the two vertices of that edge to two different sets. Then for each unassigned vertex, it calculates the value of the partial cut obtained by assigning the vertex to a particular set. Among all the values, the vertex getting the highest partial cut is assigned to the appropriate set. Thus, all the vertices are assigned to set X or set Y.

Semi Greedy – 1:

This algorithm adds randomness to the process by choosing a set of edges or vertices to choose from. Then it randomly selects from the set. The code chooses the set based on alpha. If the partial cut is greater than or equal to a certain value produced by alpha, the vertex is added to the restricted candidate list. Then we select a vertex until all vertices are assigned to a particular set. Then the max cut is calculated. In the code, alpha is taken randomly and we have 10, 20 and 50 iterations of semi greedy algorithm and taken the average of the best cut every time.

The code implements local search and GRASP algorithm to improve the performance.

Local Search:

This algorithm is used on semi greedy construction in the code. This algorithm chooses a vertex and checks if the cut can be increased when it is shifted to the other set. If the cut increases, the vertex is shifted to the other set. This process runs until we find a configuration where the current solution cannot be improved. Then we calculate the max cut. In the code, we have applied local search 10, 20 and 50 times in GRASP and taken the average of iterations and best value.

GRASP:

GRASP algorithm runs the semi greedy algorithm and local search algorithm. The code runs both of these algorithms 10, 20 and 50 times and takes the best value of max cut from it.

Explaining CSV:

The csv file contains the average value of 10 iterations from Randomized -1, the max cut value from greedy – 1, average value of semi greedy max cut, average local search iterations and average local search max cut for 10, 20 and 50 GRASP iterations. The best value of GRASP for these iterations and the known upper bound is also provided.

From the csv, it is evident that the randomized – 1 algorithm provides the worst solution of max cut as it chooses the vertices randomly without any prior calculation. Then there is semi greedy algorithm, as we take only the average of maximum iterations, most of the time it gives bad result than greedy. Local search improves the semi greedy approach and gives a better solution than greedy most of the time. Then there is GRASP algorithm, as we take the best value from iterations of semi greedy and local search, it provides the best result among all of them. The dataset proves that GRASP algorithm is better than the greedy algorithm, and randomization provides us with a better solution than greedy if we run the randomized algorithm many times in max cut problem. Besides from the csv, we can see that if we increase the iterations of GRASP, most of the time the max cut improves, though the improvement may be very low after some iterations. The GRASP algorithm gives quite a close value to the upper of the inputs. Example: it provides 94.5% of the upper bound result and 94.7% of the upper bound result for g1 and g2 inputs. So, we can conclude that GRASP algorithm works better than all other algorithms for the max cut problem.