Bangladesh University of Engineering & Technology

Course No: CSE 318

Course Title: Artificial Intelligence

Offline-3: Solving Max-Cut Problem by GRASP

Submitted By:

Name: MD. Roqunuzzaman Sojib

Roll: 1905067

Department: CSE

Section: B1

Title : Solving Maximum Cut of a Weighted Graph using Local Search

Introduction:

Although the problem of finding the minimum cut in a graph can be efficiently solved in polynomial time, the maximum cut problem is a challenging problem as it falls under the category of NP-hard problems. Consequently, no polynomial time algorithm has been discovered to find the optimal solution. This problem involves the task of partitioning the vertices of an undirected graph into two distinct sets in such a way that maximizes the number of edges between these sets, commonly referred to as "cuts."

A range of optimization techniques can be used to derive partial solutions to this problem. In the context of our given offline , we tried to solve the Max-cut problem utilizing diverse algorithms, namely the full randomized approach, semi-greedy method, complete greedy strategy, and GRASP (Greedy Randomized Adaptive Search Procedure). We then conducted a comprehensive comparison of their performances and outcomes.
Since no quick-and-easy algorithm can give us the best solution for this problem, we tried out different methods and compared them. In our offline , we used many different approaches, including GRASP which uses local search in a number of iterations to make its results better.

Algorithm:

The GRASP algorithm involves several rounds of a randomized function , each one followed by local search. The goal of local search is to make the solution better than the ones we tried before.

For the max cut problem, we created three types of constructor steps. The first one, which we'll call Simple - Randomized, worked like this:

Randomized Approach :

The constructor functions differ from each other based on their choice of choosing values for Restricted Candidate list for the max-cut.This choice is represented by a symbol called alpha (α), and in this case, we set alpha to be 0. Since alpha is 0, the value of the cut-off weight for RCL , according to the equation :

cut_off weight = Wmin + alpha(Wmax-Wmin) ....(i)*

will become , *cut_off weight = Wmin.*

So, the cut-off weight becomes the minimum possible value (Wmin). This means we consider all the vertices for the next step.

In simpler terms, the "Simple - Randomized" approach in GRASP doesn't prioritize any particular choice too much.

Semi-Greedy Approach:

In this method, we start by selecting a vertex at random. Then, we use a semi greedy strategy to decide which additional vertices to add to the same group. We keep switching between the two groups, adding vertices in a way that boosts the cut value as much as possible.

To maintain the semi greedy approach, we randomize the value of alpha. By adjusting this alpha value, we control the level of randomness in the process, including what vertices are chosen. When alpha is higher, the randomness is lower, and as alpha decreases, randomness increases. This also affects RCL(Restricted Candidate List). The bigger the alpha, the smaller the RCL, and the more optimized our outcome becomes.

Full-Greedy Approach:

This method is similar to the semi-greedy approach, but we set the alpha value to 1. When alpha is 1, we're aiming for the most optimal result, which means we're being very greedy.

So ,

cut_off weight = Wmin + alpha(Wmax-Wmin)*

will become , *cut_off weight = Wmax.*

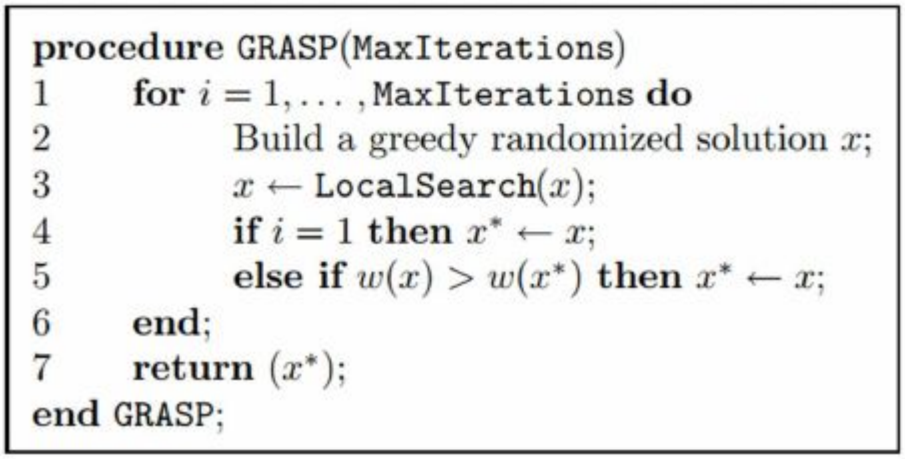
In this case, we always pick the edge with the highest weight and add it to the partition. This way, we're going for the best possible outcome at every step.

Local Search:

In the GRASP algorithm, we use a process called local search during each round to find the highest value within a specific area. In local search, we repeatedly switch the vertices connected by the edges with the highest weight between the two partitions. This helps us find the highest point in that particular region.

PseudoCode:

The procedure for GRASP :



```
procedure GRASP(MaxIterations)
    for $i = 1, \ldots, $ MaxIterations do
        Build a greedy randomized solution $x$;
        $x \leftarrow$ LocalSearch($x$);
        if $i = 1$ then $x^* \leftarrow x$;
        else if $w(x) > w(x^*)$ then $x^* \leftarrow x$;
    end;
    return ($x^*$);
end GRASP;
```

For local search I used the following procedure :

$$\sigma_X(v) = \sum_{u \in X} w_{vu} \text{ and } \sigma_Y(v) = \sum_{u \in Y} w_{vu}$$

$$w_{min} = \min\{min_{v \in V'} \sigma_X(v), min_{v \in V'} \sigma_Y(v)\}$$

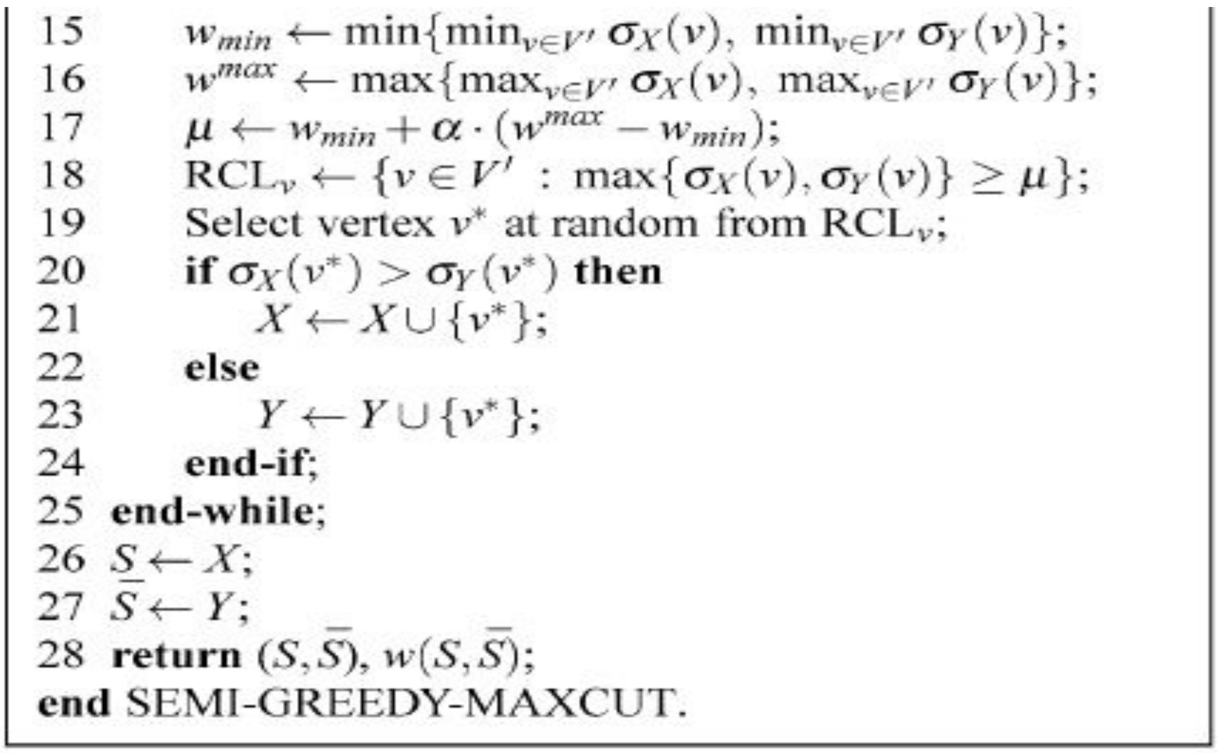
$$w_{max} = \max\{max_{v \in V'} \sigma_X(v), max_{v \in V'} \sigma_Y(v)\}$$

$$\delta(v) = \begin{cases} \sigma_S(v) - \sigma_{\bar{S}}(v), if\ v \in S \\ \sigma_{\bar{S}}(v) - \sigma_S(v), if\ v \in \bar{S} \end{cases}$$

```
procedure LocalSearch($x = \{S, \bar{S}\}$)
    $change$ ← .TRUE.
    while $change$ do;
        $change$ ← .FALSE.
        for $v = 1, \ldots, |V|$ while .NOT.$change$ circularly do
            if $v \in S$ and $\delta(v) = \sigma_S(v) - \sigma_{\bar{S}}(v) > 0$
            then do $S \leftarrow S \setminus \{v\}$; $\bar{S} \leftarrow \bar{S} \cup \{v\}$; $change$ ← .TRUE. end;
            if $v \in \bar{S}$ and $\delta(v) = \sigma_{\bar{S}}(v) - \sigma_S(v) > 0$
            then do $\bar{S} \leftarrow \bar{S} \setminus \{v\}$; $S \leftarrow S \cup \{v\}$; $change$ ← .TRUE. end;
        end;
    end;
    return ($x = \{S, \bar{S}\}$);
end LocalSearch;
```

For the constructive algorithms , the basic structure goes as follows which can be varied with respect to the value of alpha.

```
begin SEMI-GREEDY-MAXCUT;
    Generate at random a real-valued parameter $\alpha \in [0, 1]$;
    $w_{min} \leftarrow \min\{w_{ij} : (i,j) \in U\}$;
    $w^{max} \leftarrow \max\{w_{ij} : (i,j) \in U\}$;
    $\mu \leftarrow w_{min} + \alpha \cdot (w^{max} - w_{min})$;
    $\text{RCL}_e \leftarrow \{(i,j) \in U : w_{ij} \geq \mu\}$;
    Select edge $(i^*, j^*)$ at random from $\text{RCL}_e$;
    $X \leftarrow \{i^*\}$;
    $Y \leftarrow \{j^*\}$;
    while $X \cup Y \neq V$ do
        $V' \leftarrow V \setminus (X \cup Y)$;
        forall $v \in V'$ do
            $\sigma_X(v) \leftarrow \sum_{u \in Y} w_{vu}$;
            $\sigma_Y(v) \leftarrow \sum_{u \in X} w_{vu}$;
        end-forall;
```



```
        $w_{min} \leftarrow \min\{\min_{v \in V'} \sigma_X(v),\ \min_{v \in V'} \sigma_Y(v)\}$;
        $w^{max} \leftarrow \max\{\max_{v \in V'} \sigma_X(v),\ \max_{v \in V'} \sigma_Y(v)\}$;
        $\mu \leftarrow w_{min} + \alpha \cdot (w^{max} - w_{min})$;
        $\mathrm{RCL}_v \leftarrow \{v \in V' : \max\{\sigma_X(v), \sigma_Y(v)\} \geq \mu\}$;
        Select vertex $v^*$ at random from $\mathrm{RCL}_v$;
        **if** $\sigma_X(v^*) > \sigma_Y(v^*)$ **then**
            $X \leftarrow X \cup \{v^*\}$;
        **else**
            $Y \leftarrow Y \cup \{v^*\}$;
        **end-if**;
    **end-while**;
    $S \leftarrow X$;
    $\bar{S} \leftarrow Y$;
    **return** $(S, \bar{S})$, $w(S, \bar{S})$;
**end** SEMI-GREEDY-MAXCUT.
```

Input DataSet : The input data set which consists of the graph. As an unweighted graph , we have two endpoints of an edge and the associated weight of the edge .

Output :

The Output data can be found here :

https://docs.google.com/spreadsheets/d/1UQhRhThVElzJNqNZ_bM4ds8rFFTc3VLsErAa-quyqw8/edit?usp=sharing

Discussion:

As the alpha value increases, the Restricted Candidate List (RCL) becomes smaller.
A smaller RCL leads to a higher initial cut value.
With a higher alpha, the number of steps taken to reach a local maximum reduces.
Increasing the number of iterations brings the approximate value closer to the actual value.
Approximate values are almost always slightly different from the true values.

Conclusion:

In this assignment, we implemented various algorithms including randomized, semi-greedy, greedy, and GRASP, and then compared their outcomes. Through this comparison, we observed that the GRASP algorithm consistently produced the most optimized results, although these results were still below the upper limit. Since the Max-cut problem belongs to the NP-Hard category, no polynomial algorithm can provide an optimized result matching the upper limit. Consequently, the solution provided by the GRASP algorithm is acceptable.