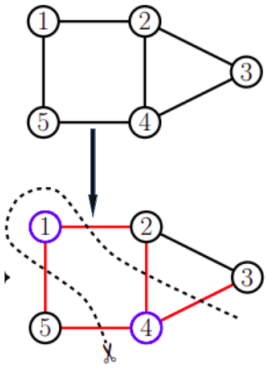
Efficient GRASP Heuristics for Max-Cut Problem

Asif Azad
1905004
Computer Science Engineering
Bangladesh University of Engineering and Technology

I. Problem Overview

Maximum Cut Problem (Max Cut), a quintessential example of a combinatorial optimization problem falls into the NP-hard complexity class. The Max Cut problem is fundamental in graph theory and finds applications in diverse domains, including network design, clustering, image segmentation, and even quantum computing.At its core, the Max Cut problem revolves around partitioning the vertices of a given graph into two disjoint sets—usually referred to as ”cuts”—such that the number of edges crossing the partition is maximized.



The Max Cut problem can be formulated mathematically as follows: Given an undirected graph $G = (V, E)$ with a set of vertices V and with a set of edges E where each edge e is associated with a weight and the goal is to partition the vertex set V into two disjoint subsets A and B such that the sum of the weights of the edges crossing the partition is maximized.

$$\max \sum_{e \in E(A,B)} w_e,$$

where $E(A, B)$ denotes the set of edges with one vertex in subset A and the other vertex in subset B.

II. GRASP

The Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic designed to tackle combinatorial optimization problems, such as the Max Cut problem, by combining the strengths of greedy construction and local search.

III. Semi-Greedy Construction Phase

The construction phase of GRASP aims to create an initial solution that serves as a starting point for the subsequent local search phase. However, instead of purely following a deterministic greedy approach, GRASP employs a semi-greedy strategy that introduces an element of randomness. In this work, we experimented on a novel semi-greedy heuristic along with the classical semi-greedy approach for max-cut.

A. Novel Semi-Greedy Heuristic

The main intuition of this semi-greedy heuristic is to maximize the cut taking the maximum weighted edge from the current position.

Algorithm 1: Novel Semi-Greedy Heuristic for Max Cut

Input : Graph $G = (V, E)$, Parameter k
Output: Partition of vertices V

```
Sort edges in non-ascending order based on weights;
foreach edge e in sorted order do
    Form a restricted list of candidate edges from the top k edges;
    Randomly select one edge e_selected from the restricted list;
    if both endpoints of e_selected are not included in any partition then
        Assign the endpoints randomly to different partitions;
    end
    else if one endpoint of e_selected is already in one partition then
        Assign the other endpoint to the opposite partition;
    end
end
```

1) Algorithm: Code

2) Time Complexity: $O(m * log(m))$

Where,
$m \rightarrow$ Number of Edges

B. Classical Semi-Greedy Heuristic

We also optimized the classical semi-greedy heuristic to achieve a time complexity of $O(n * log(n) + m * log(m))$ from a cubic time complexity.

1) Naive Approach:

a) Time Complexity: $O(n^3)$

Where,
$n \rightarrow$ Number of Vertices

2) Efficient Approach:

a) Algorithm: Code

b) Time Complexity: $O(n * log(n) + m * log(m))$

Where,
$n \rightarrow$ Number of Vertices
$m \rightarrow$ Number of Edges

IV. Local Search Phase

After generating an initial solution, the local search phase of GRASP aims to refine and improve the solution's quality through iterative neighborhood exploration. In the context of the Max Cut problem, the local search phase involves iteratively swapping vertices between the two subsets to improve the cut value. We optimized the naive local search approach to achieve a better complexity of $O(m*k)$ from a complexity $O(n * m * k)$.

A. Naive Approach

1) Algorithm: Code

2) Time Complexity: $O(n * m * k)$

Where,
$n \rightarrow$ Number of Vertices
$m \rightarrow$ Number of Edges
$k \rightarrow$ Expected Number of Iterations for Convergence

B. Efficient Approach

The key insight driving the optimization strategy revolved around the realization that altering the partition assignment of a singular vertex exclusively impacts the edges stemming from the adjacency list of that particular vertex, subsequently influencing the overall max-cut value.

Algorithm 2: Local Search Efficient Variant

Input : Initial semi-greedy solution construction
Output: Improved solution with better max-cut

$S_{\text{current}} \leftarrow$ Initial solution construction;
$max\_cut_{\text{current}} \leftarrow$ Calculate max-cut of S_{current};
while *true* **do**
  $S_{\text{neighbour}} \leftarrow$ Neighbour construction of S_{current};
  $max\_cut_{\text{neighbour}} \leftarrow max\_cut_{\text{current}}$;
  foreach *vertex* v *in* $S_{neighbour}$ **do**
    Change partition assignment of vertex v;
    foreach *edge* e *in adjacency list of* v **do**
      if *other endpoint of* e *is in opposite partition* **then**
        $max\_cut_{\text{neighbour}} \leftarrow max\_cut_{\text{neighbour}} + w_e$;
      end
      else
        $max\_cut_{\text{neighbour}} \leftarrow max\_cut_{\text{neighbour}} - w_e$;
      end
    end
    if $max\_cut_{neighbour} > max\_cut_{current}$ **then**
      $S_{\text{current}} \leftarrow S_{\text{neighbour}}$;
      $max\_cut_{\text{current}} \leftarrow max\_cut_{\text{neighbour}}$;
      continue;
    end
    Change partition assignment of vertex v back;
  end
  if *no neighbour construction with better max-cut found* **then**
    break;
  end
end

1) Algorithm: Code

2) Time Complexity: $O(m * k)$

Where,
$m \rightarrow$ Number of Edges
$k \rightarrow$ Expected Number of Iterations for Convergence

V. Statistics and Insights

In this section, we explore the valuable findings resulting from our experiments involving the novel semi-greedy constructive heuristic, the optimized local search process, and the overall performance of the GRASP algorithm across various instances.

A. Detailed Statistics

Statistics sheet

B. Optimized Running Time

One of the most compelling revelations emerged in the realm of computational efficiency. Through meticulous optimization of both the construction phase and local search phase, the GRASP algorithm exhibited a substantial enhancement in running time. Notably, the algorithm's performance became notably swift, as evidenced by the "Running Time" column in

the statistics sheet. The time for running the GRASP on all the benchmark tests reduced to 31.6 minutes from hours. The optimization efforts streamlined the algorithm's complexity, rendering it blazingly fast and capable of tackling instances of varying complexities.

C. Performance of Novel Semi-Greedy Constructive Heuristic

When confronted with uniformly distributed edge weights, the heuristic exhibited competitive results in comparison to the classical semi-greedy approach. However, a divergence in performance surfaced when edge weights were nearly identical. This outcome aligns with the heuristic's inclination to prioritize edges with higher weights, potentially neglecting other crucial structural attributes of the graph.

D. Local Search Convergence and Iterations

Following the initial construction phase, we applied a local search algorithm to both the classical semi-greedy approach and the novel heuristic. Intriguingly, the local search yielded comparable results for both methods.Nevertheless, the novel heuristic necessitated a greater number of iterations to achieve convergence.