Check-Point 2 Report

Analysis and Implementation Outlook of Edge-Connectivity Augmentation

Roshaan Khan - rk08103
Hamza Ansari - ha08033

April 8, 2025

Problem and Contribution

The paper addresses the problem of **increasing the edge-connectivity of a simple k-edge-connected graph G by one**, using the minimum number of edges from the complement of the graph $\bar{G}$. The key challenge basically is finding an efficient and minimal set of edges from the complement graph, $\bar{G}$, such that the resulting graph G' becomes $(k+1)$-edge-connected.
When coming to the paper's main contribution, it is the identification of a dichotomy: depending on whether $\bar{G}$ contains a matching that covers all vertices of degree k in G, the augmentation process follows two different algorithmic strategies, both achievable in **polynomial time**. In other words, we are dealing with a NP-hard problem and trying to come up with a solution in polynomial time.

Algorithmic Description

The core idea of the algorithm is to decide between two augmentation strategies based on the structure of $\bar{G}$:

- **Case 1:** If $\bar{G}$ contains a matching covering all vertices of degree k in G, then this matching can be used directly as the augmenting edge set. The new graph $G \cup M$ becomes $(k+1)$-edge-connected.
- **Case 2:** If such a matching does not exist, the algorithm constructs a path-based augmenting system where paths of length 1 or 2 (max) are used to connect unmatched vertices. This system is initially found as a minimum-degree augmenting path system and then converted into an edge-connectivity augmenting one using the same number of edges.

The inputs to the algorithm are:

- A simple graph G that is k-edge-connected.
- Its complement $\bar{G}$.

The output is:

- A set of edges from $\bar{G}$ that, when added to G, makes it $(k+1)$-edge-connected.

Comparison with Existing Approaches

The traditional edge-connectivity augmentation problems make use of the addition of arbitrary edges. However, there are more efficient algorithmic-based approaches which we as Algorithms and Design students aim to look at. What makes this paper novel is the restriction to edges only from the **complement graph**. This variation has not been studied as extensively and presents unique challenges due to its structural constraints. The algorithm leverages known techniques (like Edmonds' matching algorithm) but applies them in a constrained and direct way, which gives a solution that is efficient in theory and precise in application.

Data Structures and Techniques

- Coming to the dataset itself, our idea is to work with more synthetic data sets, making graphs ourselves to make comparisions.
- **Matching Theory:** Specifically, Edmonds' Blossom Algorithm for finding maximum matching in general graphs.
- **Graph Complements:** Using $\bar{G}$ to select feasible augmenting edges.
- **Subgraph Induction:** Constructing G_k, the subgraph of vertices with degree k.
- **Path Systems:** Using paths of length 1 or at max 2 to construct efficient augmentation sets.

Implementation Outlook

- **A Matching Algorithms:** Implementing Edmonds' Blossom Algorithm might be a bit complex and requires careful data structure design. It makes use of some new concepts (e.g., alternating trees, union-find structures).
- **Graph Representation:** Maintaining both the original graph and its complement efficiently is essential, especially for large graphs.
- **Scalability:** While the algorithm is polynomial, its runtime may still be high for graphs with thousands of nodes. The paper itself is dealing with specific types of simple graphs.

- **Path Transformation:** The transformation from minimum-degree path systems to edge-connectivity path systems must preserve correctness and may involve intricate logic.

Implementation into Algorithm: A Basic Idea

1. **Input Representation**
   Accept a simple undirected graph $G = (V, E)$ as input. The graph can be represented using either an adjacency list or an adjacency matrix. Compute the edge-connectivity k of the graph using standard graph algorithms.

2. **Identify Critical Vertices**
   Identify the set V_k of vertices in G that have degree exactly k.

3. **Construct Complement Graph**
   Construct the complement graph $\bar{G} = (V, \bar{E})$ such that $\bar{E} = \{(u, v) \mid u, v \in V, u \neq v, (u, v) \notin E\}$. Focus on the induced subgraph $\bar{G}_k$ formed by the vertices in V_k.

4. **Apply Edmonds' Matching Algorithm**
   Use Edmonds' Blossom Algorithm to compute a maximum matching M in $\bar{G}_k$. This step is for identifying the largest set of disjoint pairs of vertices in V_k.

5. **Case Analysis**

   - **Case 1:** If the matching M covers all vertices in V_k, add all edges from M to G. The resulting graph becomes $(k + 1)$-edge-connected.
   - **Case 2:** If the matching does *not* cover all vertices in V_k, construct a path-based augmentation system using paths of length 1 or 2 to connect the unmatched vertices. Choose the minimal number of edges required to ensure increased edge-connectivity.

6. **Updating the graph and check**
   Add the selected edges (from the matching or path-based augmentation) to the original graph G. Optionally, recompute the edge-connectivity to verify the correctness of the augmentation.