Technical Summary: Deterministic Online Bipartite Edge Coloring

Team Name / Your Name

April 12, 2025

1 Problem and Contribution

Problem Statement: The paper addresses the *online edge coloring* problem in bipartite graphs. In this setting, the graph's edges are revealed sequentially, and an algorithm must assign a color to each edge immediately upon arrival. The coloring must be proper (no two adjacent edges share the same color). While Kőnig's theorem guarantees that an optimal offline coloring of a bipartite graph requires at most Δ colors (where Δ is the maximum degree), the online constraint makes it challenging to match this bound.

Main Contribution: The authors present a *deterministic* online algorithm that achieves a competitive ratio close to the optimal number of colors. Unlike randomized approaches, which may only guarantee good performance in expectation, the deterministic algorithm provides worst-case guarantees. The novelty lies in how the algorithm adapts to the online arrival of edges, ensuring that each decision is made with provable bounds on the total number of colors used.

2 Algorithmic Description

Input and Output:

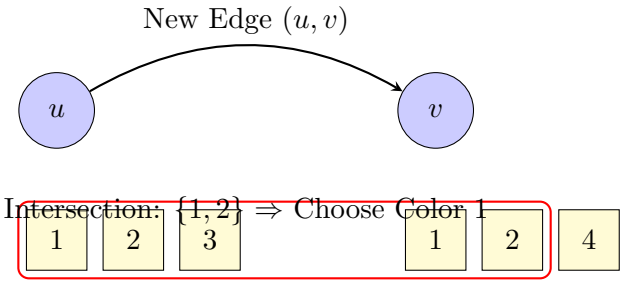
- **Input:** A bipartite graph $G = (U, V, E)$ with edges arriving one-by-one in an online fashion.
- **Output:** An assignment of colors to each edge such that no two edges sharing a vertex receive the same color.

High-Level Idea: The algorithm maintains a set of *available colors* for each vertex, updating these sets as new edges arrive. When an edge (u, v) is revealed, the algorithm:

1. Determines the intersection of the available color sets for vertices u and v.
2. Assigns the smallest available color (according to a fixed order) from this intersection.
3. Updates the available color sets for both u and v, ensuring future edges are assigned colors that do not conflict.

Edge Assignment Diagram

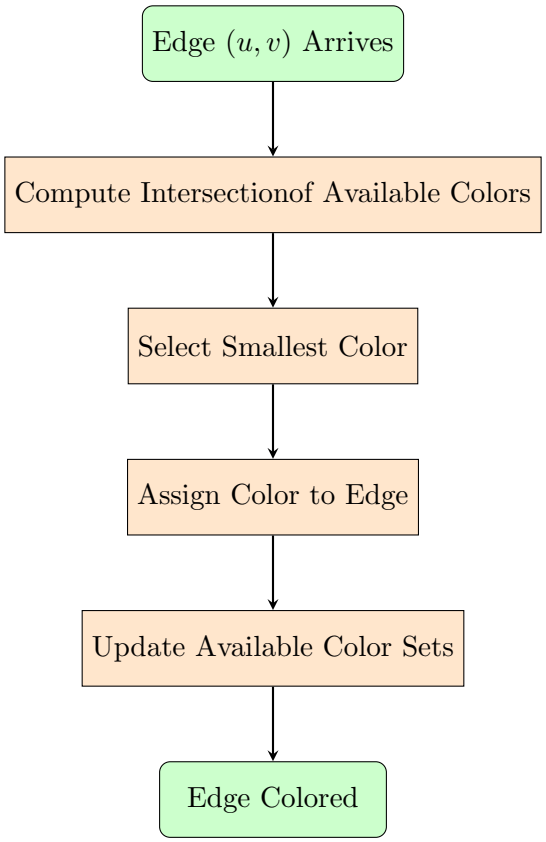
The diagram below illustrates two vertices u and v with their available color sets. When an edge arrives between them, the intersection of these sets is highlighted (in red), and the smallest color from the intersection is chosen and assigned.



Algorithm Flowchart

The flowchart below outlines the step-by-step process of the algorithm:

1. **Edge Arrival:** An edge (u, v) arrives.
2. **Compute Intersection:** Compute the intersection of available colors for u and v.
3. **Select Color:** Choose the smallest color from the intersection.
4. **Assign & Update:** Assign this color to the edge and update both vertices' available color sets.



3 Comparison with Existing Approaches

Existing Work: Previous algorithms for online edge coloring include both randomized methods and simpler deterministic greedy approaches. Randomized algorithms can sometimes achieve near-optimal performance in expectation but lack strong worst-case guarantees. Greedy methods, on

the other hand, tend to be simple but may use significantly more than Δ colors in adversarial scenarios.

Novelty and Advantages: The proposed deterministic algorithm is novel because it:

- Provides worst-case performance guarantees without relying on randomness.
- Uses a more sophisticated approach to manage available color sets and updates, reducing the risk of "color exhaustion" at any vertex.
- Balances the trade-off between immediate decision-making and long-term performance, which is critical in online settings.

4 Data Structures and Techniques

Data Structures:

- **Color Availability Lists:** Arrays or bit vectors for each vertex that track which colors are still available.
- **Adjacency Lists:** To store the graph structure and quickly access neighbors when updating available colors.

Techniques:

- **Greedy Selection:** Choosing the smallest available color from the intersection of the two vertices' available color sets.
- **Potential Function Analysis:** Used in the theoretical analysis to guarantee that the number of colors used does not exceed the bound by too large a factor relative to Δ.
- **Online Algorithm Design:** Strategies for making irrevocable decisions based on the current state and limited future knowledge.

5 Implementation Outlook and Anticipated Challenges

Technical Challenges:

- **Dynamic Updates:** Efficiently updating the available color sets in real time as new edges are added, especially in graphs with high vertex degrees.
- **Scalability:** Ensuring that the algorithm remains efficient with large input sizes. This includes managing memory usage for the color availability data structures.
- **Edge Cases:** Handling adversarial edge sequences that may try to force the algorithm into using suboptimal color choices.
- **Numerical Precision:** While less of a concern in combinatorial problems, careful bookkeeping is needed when indexing and managing color sets.

Implementation Strategy: To address these challenges, we plan to:

- Optimize the data structures using arrays or bit masks for fast bitwise operations.
- Use rigorous unit testing to cover worst-case scenarios and adversarial inputs.
- Profile the implementation on synthetic datasets that mimic worst-case inputs to ensure the algorithm's performance aligns with theoretical guarantees.