Graph Coloring with Transformers

Under the supervision of Dr. Martin

Injamam Ul Karim & Pratik Dhameliya

January 29, 2025

Outline

Introduction to Graph Coloring

- **Definition of Graph Coloring Problem:**
  - Given a graph $G = (V, E)$ and k colors, assign a color to each vertex such that no two adjacent vertices share the same color.
- **Challenges:**
  - Graph coloring is *NP-hard*.
  - Traditional methods are computationally expensive for large graphs.
- **Our Task:**
  - Solve the graph coloring problem using a transformer-based model with an unsupervised learning technique.

Graph Coloring as a Constraint Satisfaction Problem (CSP)

- **Formulation:**

$$\min \sum_{(u,v)\in E} \mathbf{1}[\text{color}(u) = \text{color}(v)]$$

- **Key Details:**
  - Each node has a discrete domain of k colors.
  - **Constraint:** Adjacent nodes cannot share the same color.
  - **Objective:** Minimize conflicts (unsatisfied edges).

our approach and loss function

Probability of Different Colors

$$p_{\text{diff}}(u, v) = \sum_{i=1}^{C} \sum_{j=1}^{C} p_u(i) \cdot A_{\text{neq}}(i, j) \cdot p_v(j)$$

- Score of assigning u and v different color.
- **Matrix** $A_{\mathbf{neq}_{(CXC)}}$:
  - $A_{\text{neq}}(i, j) = 1$ if $i \neq j$.
  - $A_{\text{neq}}(i, j) = 0$ if $i = j$.

Negative Log-Likelihood

$$\text{loss}(u, v) = -\log(p_{\text{diff}}(u, v))$$

Mathematical Derivation

Total Loss

$$\text{total\_loss} = \frac{1}{|\text{edges}|} \sum_{(u,v)\in\text{edges}} -\log(p_{\text{diff}}(u,v))$$

- Therefore, our model minimizes this loss function, learning that u and v are adjacent.

What is the Distance-Based Bias Term?

- The **distance-based bias term** incorporates graph structure into the Transformer.
- It modifies attention scores to focus on local neighborhoods and mask out distant nodes.
- **Shortest-Path Distance Computation**:
  - Computes pairwise distances using BFS.
  - Ensures global graph structure awareness.
- **DistanceEncoder**:
  - Encodes distances into embeddings.
  - Projects embeddings into scalar biases.
- **DistanceAwareTransformerLayer**:
  - Combines self-attention and feedforward layers.
  - Uses distance-based attention masks.

Attention Mask

Attention Mask Construction:

$$M_{ij} = \begin{cases} f(D_{ij}), & \text{if } D_{ij} \leq d_{\text{local}} \\ -\infty, & \text{otherwise} \end{cases}$$

- d_{local}: Predefined threshold distance.
- $f(D_{ij})$: Learned distance encoding via the distance encoder.

Effect:

- Nodes within the local distance threshold retain meaningful attention values.
- Distant nodes are masked out (assigned $-\infty$).

How Does the Bias Term Influence the Model?

- **Attention Mechanism**:
  - Modified attention scores:

$$\text{Modified Attention Score}(u, v) = \text{softmax}\left(\frac{Q_u \cdot K_v}{\sqrt{d_k}} + \text{Bias}(u, v)\right)$$

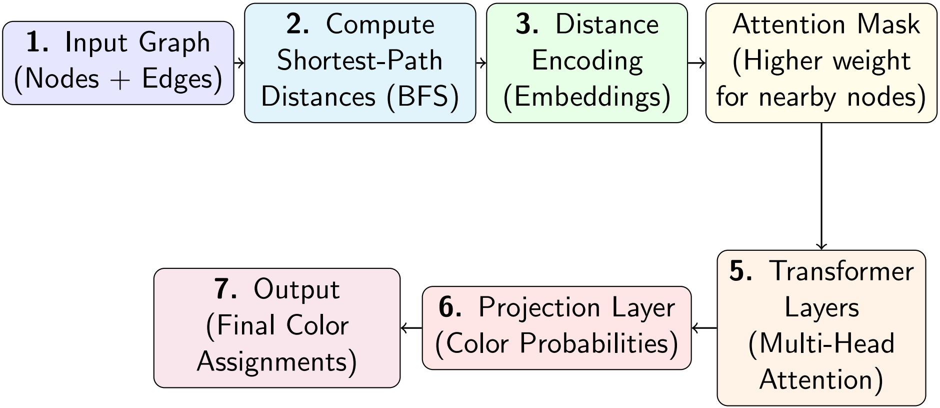
  - Encourages attention to **local neighborhoods**.
  - Masks out **distant nodes** (e.g., $-\infty$).
- **Efficiency**:
  - Reduces computational complexity by focusing on local neighborhoods.
  - Thus it learns the connectedness of two nodes

Model Architecture: Distance-Aware Transformer

Key Features:

- **Distance-Aware Attention Mask:** Encodes graph structure via BFS distances.
- **Multi-Head Attention Mechanism:** Captures local and global interactions.
- **Layer Normalization and Skip Connections:** Ensures stable training and gradient flow.

Graph Coloring Process: Step-by-Step



Edge Conflict Evaluation in Graph Coloring

Purpose: The function `evaluate_unsatisfied_percentage` computes the percentage of edges in a graph where both endpoints have been assigned the same color.

Mathematical Formulation:

- **Compute Node Color Assignments:**

$$A_u = \arg\max_i P_u(i), \quad \forall u \in \{1, 2, ..., N\}, \quad i \in \{1, 2, ..., K\}$$

  where A_u is the most probable color for node u.

- **Compute Unsatisfied Edge Percentage:**

$$\text{Unsatisfied\%} = \frac{\sum_{(u,v)\in E} \mathbf{1}(A_u = A_v)}{|E|} \times 100$$

Generating Dataset

Graph Types:

- **GNM Random Graphs:** Random n-node, m-edge graphs.
- **Geometric Graphs:** Nodes connected by geometric proximity.
- **Power-Law Cluster Graphs:** Scale-free graphs with communities.
- **Caveman Graphs:** Dense clusters with sparse interconnections.
- **Mix Graphs:** Mixing all data evenly.

Training and Validation

Dataset Highlights:

- We generate 5000 evenly mixed graphs.
- Train the model on 4000 graphs.
- Validate on 1000 graphs.

Testing

Testing on Different Graph Types:

- We test on 1000 graphs of each type: GNM, Powerlaw, Geometric, and Caveman.

Experiment

Setup:

- Train the best model with the following configuration:
- **Embedding Dim:** 128.
- **Num Colors:** 3.
- **Num Transformer Blocks:** 5.
- **Optimizer:** Adam optimizer (6×10^{-5}).
- **Epochs:** 25.
- **Scheduler:** Reduce on plateau with patience $= 4$, factor $= 0.8$.

Optimization Goals:

- Minimize graph coloring loss.
- Efficiently reduce unsatisfied constraints in validation sets.

Comparison of Unsatisfied Constraints (%):

Table: Comparison of Unsatisfied Constraints (%) Across Different Graph Types

Graph Type	**Literature (%)**	**Achieved (%)**
GNM (gnm_graphs)	4.73	3.06
Powerlaw (pwl_graphs)	1.89	2.11
Geometric (geo_graphs)	10.18	11.80
Caveman (cc_graphs)	2.33	18.53

Issues

- **Lack of Experiments:** Due to frequent cluster disconnections.
- **Bottleneck Problem:** Calculating the distance matrix every epoch is computationally expensive.

Future Directions

- Solve (Tried) the bottleneck problem for better performance.
- Perform more experiments with hyperparameter tuning (e.g., different techniques for node embedding initialization, layer, and embedding dimensions).
- More comparison literature for additional insights.

Conclusion

Summary:

- Introduced a distance-aware Transformer for graph coloring.
- Used an unsupervised technique for constraint satisfaction.
- Achieved promising results with low unsatisfied constraints.

Thank you

Thank you!