Graph Coloring with Transformers

Step-by-Step Workflow

Injamam Ul Karim & Pratik Dhameliya

January 29, 2025

Outline

1. Graph Coloring Transformer Workflow
2. Step-by-Step Execution

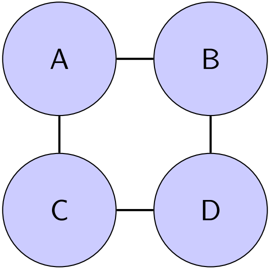
Graph Coloring Transformer Workflow

Step-by-Step Flow of the Model:

1. **Input:** Graph with nodes and edges.
2. **BFS Calculation:** Compute shortest-path distances.
3. **Distance Encoding:** Map distances to embeddings.
4. **Attention Masking:** Assign higher attention to close nodes.
5. **Initialize Node Embeddings:** Assign random initial vectors.
6. **Transformer Layers:** Multi-head attention refines embeddings.
7. **Projection Layer:** Compute color probabilities.
8. **Final Output:** Assign discrete colors.

Step 1: Input Graph Example

Consider a simple graph:



Edge List: {(A,B), (B,D), (C,D), (A,C)}

Goal: Assign colors to nodes such that no two connected nodes share the same color.

Step 2: Compute Shortest-Path Distances (BFS)

Distance Matrix for Graph

$$\begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}$$

Key Idea:

- Nodes with **shorter distances** should have **higher attention weights**.
- Nodes **far apart** should have **low influence** on each other.

Step 3: Distance Encoding

Encoding Process:

- Distances are **clamped and mapped** to embeddings.
- The model assigns **biases** based on distances.

Example Mapping:

- **Distance 1** $\rightarrow$ Small bias (nodes highly connected).
- **Distance 2** $\rightarrow$ Medium bias (nodes slightly connected).
- **Distance Greater 2** $\rightarrow$ Large bias or ignored.

Mathematical Representation:

$$\text{dist\_bias}[u, v] = \text{Embedding}(\text{clamp}(d(u, v), 0, \text{max\_distance} + 1))$$

Step 4: Generate Attention Mask

Purpose of Attention Mask:

- Determines which nodes can attend to each other.
- Nearby nodes get **higher attention scores**.
- Distant nodes receive **a large penalty weight** (10^9).

Example Attention Mask:

$$\begin{bmatrix} 0 & 0.8 & 0.8 & 10^9 \\ 0.8 & 0 & 10^9 & 0.8 \\ 0.8 & 10^9 & 0 & 0.8 \\ 10^9 & 0.8 & 0.8 & 0 \end{bmatrix}$$

Step 5: Initialize Node Embeddings (128-D)

Initial Embeddings:

- Each node starts with a **vector of ones**:

$$\mathbf{E}_i = [1, 1, ..., 1] \quad \text{(Size: 128)}$$

- No meaningful structure yet; embeddings will be refined in later steps.

Embedding Representation:

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}_{n \times 128}$$

Step 6: Multi-Head Attention (128-D)

How Multi-Head Attention Works:

1. **Normalize input embeddings** (LayerNorm).
2. **Compute attention scores** using self-attention.
3. **Apply attention mask**
4. **skip connection**.
5. **Normalize output** before passing to the next layer.
6. **Apply feed-forward network**.
7. **another skip**

Mathematical Representation:

$$\text{Updated Embeddings} = \text{LayerNorm}(\text{Embeddings} + \text{Attention}(\text{Embeddings}))$$

Example: Multi-Head Attention Transformation

Initial Embeddings (Before Attention):

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

After Attention (Refined by Context):

$$\begin{bmatrix} 0.6 & 0.3 & \dots & 0.7 \\ 0.2 & 0.7 & \dots & 0.5 \\ 0.7 & 0.4 & \dots & 0.8 \\ 0.3 & 0.9 & \dots & 0.5 \end{bmatrix}$$

Observation: - Initial values were **all ones**. - After attention, values are influenced by connected nodes.

Step 7: Projection Layer

What Happens in the Projection Layer?

- Outputs **softmax probabilities** over possible colors.
- Converts learned embeddings into meaningful **color predictions**.

Example Soft Assignments:

$$\begin{bmatrix} \text{A: } [0.7, 0.2, 0.1] \\ \text{B: } [0.1, 0.8, 0.1] \\ \text{C: } [0.3, 0.2, 0.5] \\ \text{D: } [0.2, 0.7, 0.1] \end{bmatrix}$$

Step 8: Assign Final Colors

Hard Assignments from Soft Probabilities:

- **Node A $\rightarrow$ Red**
- **Node B $\rightarrow$ Blue**
- **Node C $\rightarrow$ Green**
- **Node D $\rightarrow$ Blue**

No adjacent nodes share the same color!

Conclusion

Key Takeaways:

- **Distance-aware attention** efficiently encodes graph structure.
- **Transformer updates embeddings** for optimal coloring.
- **Final assignments minimize unsatisfied constraints**.

Thank You!