1 Proof of NP-complete

To prove that the "Node-Labelling Decision Problem" is NP-complete, we need to show two things:

1. The problem is in NP:

   Given a certificate, which is a labeling of the nodes in the graph, we can easily verify in polynomial time whether the labeling is valid or not.

   Specifically, given an instance of the Node-Labelling Decision Problem with graph $G = (V, E)$, label set $K = 0, 1, \ldots, k - 1$, and non-negative integer R, suppose we have a labeling function $c : V \to K$. To verify that this labeling is valid and satisfies the requirement that $|C(v, R)| = k$ for every vertex $v \in V$, we can perform the following steps:

   (a) Check if every label in K is assigned to at least one node in G. This can be done by iterating through all vertices $v \in V$ and maintaining a set of encountered labels. This step takes $O(|V|)$ time.

   (b) For each vertex $v \in V$, calculate the set $N(v, R)$, which contains all nodes within R hops from v. This can be done using a Breadth-First Search (BFS) starting from v, and stopping once we reach depth R. Since BFS has a time complexity of $O(|V| + |E|)$, this step takes $O(|V|(|V| + |E|))$ time in total for all vertices.

   (c) For each vertex $v \in V$, calculate the set $C(v, R)$, which contains the distinct labels within R hops from v. This can be done by iterating through $N(v, R)$ and maintaining a set of encountered labels. This step takes $O(|V|^2)$ time in total for all vertices.

   (d) Check if $|C(v, R)| = k$ for every vertex $v \in V$. This step takes $O(|V|)$ time.

   The overall time complexity for verifying the solution is $O(|V|^2 + |V|(|V| + |E|))$. Since this is polynomial in the size of the input, we conclude that the Node-Labelling Decision Problem is in NP.

2. The problem is NP-hard:

   We will show that the problem is NP-hard by reducing the 3-coloring problem (which is known to be NP-complete) to the Node-Labelling Decision Problem. Suppose we have an instance of the 3-coloring problem, which is an undirected graph $G = (V, E)$, and we want to determine if there exists a valid 3-coloring of the graph.

   Now we will construct a new instance $G' = (V', E')$ of the Node-Labelling Decision Problem as follows:

   (a) Initially, let $V' = V$. (This step takes $O(|V|)$ time to create the set V'.)

(b) For each edge $(u, v) \in E$, add a new vertex $w_{(u,v)}$ to V' and edges $(u, w_{(u,v)}), (v, w_{(u,v)})$ to E'. (This step takes $O(|E|)$ time to process all edges and create the new vertices and edges.)

(c) For each vertex u in V, add two new vertices u_1, u_2 to V', and add edges (u, u_1), (u, u_2), (u_1, u_2) to E'. (This step takes $O(|V|)$ time to process all vertices and create the new vertices and edges.)

(d) Set the set of labels $K = 0, 1, 2$, and set $R = 1$. (This step takes $O(1)$ time.)

The total time complexity of the reduction is $O(|V| + |E|)$, which is polynomial with respect to the size of the input graph $G = (V, E)$. Therefore, the reduction from the 3-coloring problem to the Node-Labelling Decision Problem is in polynomial time.

Now we will prove that G has a valid 3-coloring if and only if there exists a valid labeling for the Node-Labelling Decision Problem with the constructed G', K, and R.

($\Rightarrow$) Let $c : V \to K$ be a valid 3-coloring of G. We will extend this coloring to the vertices of G' as follows:

- For each vertex $v \in V$, we assign $c'(v) = c(v)$.
- For each edge $(u, v) \in E$, we assign $c'(w_{(u,v)}) = 3 - c(u) - c(v)$, which is the remaining color not used by u or v.
- For each vertex $u \in V$, we assign $c'(u_1) = (c(u) + 1) \mod 3$ and $c'(u_2) = (c(u) + 2) \mod 3$.

Now, we will show that this labeling c' is a valid solution for the Node-Labelling Decision Problem with G', K, and R.

- First, all labels in K are used in G', so the labeling is valid.
- Second, we need to show that for each vertex $v \in V'$, we have $|C(v, R)| = 3$. Since $R = 1$, we only need to consider the 1-hop neighborhood of each vertex.
  - For each vertex $v \in V$, its 1-hop neighbors are v_1, v_2, and all the $w_{(u,v)}$ where $(u, v) \in E$. Since $c'(v_1) \neq c'(v_2)$ and $c'(v) \neq c'(v_1), c'(v_2)$, we have $|C(v, 1)| = 3$.
  - For each vertex $w_{(u,v)}$, its 1-hop neighbors are u and v. Since $c'(u) \neq c'(v)$ and $c'(w_{(u,v)}) \neq c'(u), c'(v)$, we have $|C(w_{(u,v)}, 1)| = 3$.
  - For each vertex u_1, its 1-hop neighbors are u and u_2. Since $c'(u) \neq c'(u_2)$ and $c'(u_1) \neq c'(u), c'(u_2)$, we have $|C(u_1, 1)| = 3$.
  - Similarly, for each vertex u_2, we have $|C(u_2, 1)| = 3$.

Thus, if G has a valid 3-coloring, there exists a valid labeling for the Node-Labelling Decision Problem with the constructed G', K, and R.

($\Leftarrow$) Now, let $c' : V' \to K$ be a valid labeling for the Node-Labelling Decision Problem with G', K, and R. We will use this labeling to construct a valid 3-coloring for G.

Define a coloring $c : V \to K$ such that for each vertex $v \in V$, we assign $c(v) = c'(v)$. We will show that c is a valid 3-coloring for G.

We need to show that for each edge $(u, v) \in E$, we have $c(u) \neq c(v)$. Recall that we added a vertex $w_{(u,v)}$ in G', and edges $(u, w_{(u,v)})$, $(v, w_{(u,v)})$ in E'. Since c' is a valid labeling for the Node-Labelling Decision Problem with $R = 1$, we have $|C(w_{(u,v)}, 1)| = 3$. Therefore, $c'(w_{(u,v)})$, $c'(u)$ and $c'(v)$ must be the three distinct labels. This implies that $c(u) = c'(u) \neq c'(v) = c(v)$.

Thus, if there exists a valid labeling for the Node-Labelling Decision Problem with the constructed G', K, and R, then G has a valid 3-coloring.

In summary, we have shown that G has a valid 3-coloring if and only if there exists a valid labeling for the Node-Labelling Decision Problem with the constructed G', K, and R. This reduction can be done in polynomial time. Hence, the Node-Labelling Decision Problem is NP-hard.

Since we have shown that the Node-Labelling Decision Problem is both in NP and NP-hard, we conclude that the problem is NP-complete.

2 Main idea of algorithm

1. Select a root node with the minimum m value using *select_root_with_minimum_m* function.
2. Perform BFS on the tree, starting from the root node.
3. At each node visited:
   (a) Compute the impact of each available label using *label_impact* function.
   (b) Select the label with the minimum impact.
   (c) Assign the selected label to the node.
4. Continue until all nodes are visited and labeled.

3 Pseudo code

Algorithm 1 Tree Node-Labelling Algorithm

```
procedure ModifiedTreeNodeLabelling($(G, k)$)
    $root \leftarrow select\_root\_with\_minimum\_m(G, k)$
    Initialize empty dictionary $labels\_assigned$
    $BFS\_queue \leftarrow [(root, [])]$
    while BFS_queue is not empty do
        $(v, visited) \leftarrow$ dequeue $BFS\_queue$
        Compute available labels for node $v$
        for each label in available labels do
            Compute $label\_impact(v, label)$
        $best\_label \leftarrow$ label with minimum impact
        $labels\_assigned[v] \leftarrow best\_label$
        for each child in $children\_of(G, v, visited)$ do
            Enqueue $(child, visited.copy())$ in $BFS\_queue$
    return $labels\_assigned$
```

4 Proof of proximity ratio

Lemma 1. *For any node v in the tree, if the label selected by the algorithm results in the smallest $r(v)/m(v)$ ratio for the node, then the overall maximum ratio in the tree is minimized.*

Proof. Assume there exists another label assignment that would result in a smaller maximum ratio in the tree. Let L be the set of labels assigned by our algorithm, and L' be the set of labels assigned by the alternative assignment.

For any node v, let $r(v, L)$ and $m(v, L)$ be the $r(v)$ and $m(v)$ values when labels are assigned according to L, and $r(v, L')$ and $m(v, L')$ be the $r(v)$ and $m(v)$ values when labels are assigned according to L'.

Let v^* be the node with the highest $r(v, L)/m(v, L)$ ratio when using our algorithm. Since our algorithm selects the label that results in the smallest $r(v)/m(v)$ ratio for each node, we have:

$r(v^*, L)/m(v^*, L) \leq r(v^*, L')/m(v^*, L')$

Now, let v' be the node with the highest $r(v, L')/m(v, L')$ ratio when using the alternative assignment. Since the maximum ratio in the tree is minimized by our algorithm, we have:

$r(v^*, L)/m(v^*, L) \leq r(v', L')/m(v', L')$

From the two inequalities above, we can conclude that:

$r(v^*, L)/m(v^*, L) \leq r(v', L')/m(v', L') \leq r(v^*, L')/m(v^*, L')$

This contradicts our assumption that there exists another label assignment that would result in a smaller maximum ratio in the tree. Thus, the lemma is proven. □

Now we can analyze the proximity ratio of the algorithm. Let ρ be the maximum ratio achieved by the algorithm for any input instance, and ρ^* be the optimal proximity ratio for that instance.

By Lemma 1, our algorithm minimizes the maximum ratio in the tree. Therefore, ρ must be less than or equal to the maximum ratio of any other possible labeling. Since ρ^* is the optimal proximity ratio for that instance, we know that ρ^* is the smallest possible maximum ratio that can be achieved. Therefore, we have:

$\rho^* \leq \rho$

Now, let's analyze the relationship between $r(v)$ and $m(v)$. Since $r(v)$ must be at least $m(v)$ for any node v (as $m(v)$ is a lower bound of $r(v)$), we know that $r(v)/m(v) \geq 1$. Also, note that the maximum possible $r(v)$ value for any node v is $|V| - 1$, as in the worst case, we would need to visit all nodes in the tree to find all k labels. Therefore, the maximum possible value for $r(v)/m(v)$ is $(|V| - 1)/1 = |V| - 1$. Thus, we have:

$1 \leq \rho^* \leq \rho \leq |V| - 1$

This inequality establishes that our algorithm has a proximity ratio ρ that is bounded by a constant value for any input instance. While the proximity ratio is not constant across all input instances (as it depends on the size of the graph), it is constant for each individual input instance.

5 Analysis of Time Complexity

The time complexity of the algorithm can be analyzed as follows:

- The *select_root_with_minimum_m* function computes $m(v)$ values for all nodes in the tree. In the worst case, this takes $O(|V|^2)$ time.
- The BFS traversal of the tree takes $O(|V| + |E|)$ time, but since the input graph is a tree, $|E| = |V| - 1$, so the BFS traversal takes $O(|V|)$ time.
- When visiting each node during BFS, the algorithm computes the impact of each available label using the *label_impact* function. Since there are k labels, this step takes $O(k)$ time per node. For all nodes in the tree, this step takes $O(k|V|)$ time.
- Selecting the label with the minimum impact and assigning it to the node is an $O(1)$ operation.
- Continuing until all nodes are visited and labeled results in a total time complexity of $O(|V|^2) + O(|V|) + O(k|V|) = O(|V|^2 + k|V|)$.

As k is a constant, the time complexity of the algorithm is polynomial in the number of nodes, which is $O(|V|^2)$. Thus, the algorithm has polynomial-time complexity.