Machine Learning with Graphs: Homework I - Graph Algorithms Due: 01/29

Spring 2026

Reminder: You can choose to solve problem 2 or 3.

Problem 1 (conceptual)

For each of the following abstract problems, (1) formalize its objective, (2) describe a poly-time algorithm (you don't need to invent it) with its complexity, (3) state whether the algorithm is exact or approximate based on the objective, and (4) illustrate the execution of your algorithm on small graph as an example.

a. Drawing a graph so that node positions reflect structural proximity.

b. Comparing two graphs based on their structural similarity.

c. Comparing two trees based on their structural proximity

d. Grouping the nodes in the graph based on their structural proximity.

Problem 2 (optional, analytical)

The binary graph classification problem consists of learning a function $f(G_t) \approx y_t$, where $y_t \in \{0,1\}$ and $G_t = (V_t, E_t, L_t)$ is a labeled graph with $L_t : V \to \{1, \ldots K\}$ being a node labelling function. Let f be a simple linear classifier that applies a given set of labeled subgraphs $H_1, H_2, \ldots H_R$ as features. More specifically, let $g(G_t) = \sum_{i=1}^{R} \alpha_i \times count(G_t, H_i)$, where $\alpha_1, \ldots \alpha_R$ are learnable weights and $count(G_t, H_i)$ returns the number of subgraphs in G_t that are (label-preserving) isomorphic to H_i. We predict $f(G_t) = 1$ if $g(G_t) > 0.5$ and $f(G_t) = 0$, otherwise. For the following settings, either provide (in high-level) a poly-time algorithm for computing g (for any given/fixed weights α) or show that the problem is computationally hard—i.e. as hard as an NP-complete problem.

a. Graphs (G_t) and subgraphs (H_i) are constrained to be path graphs.

b. Graphs (G_t) and subgraphs (H_i) are constrained to be trees.

c. Only subgraphs (H_i) are constrained to be paths.

d. Only subgraphs (H_i) are constrained to be trees.

e. Subgraphs (H_i) have a fixed maximum number of vertices (e.g. up to 4 vertices).

Problem 3 (optional, coding)

Your goal is to test whether a Large Language Model (LLM) of your choice can solve some of the graph problems discussed in class. Select two problems and a set of small graphs for which you are able to compute the optimal (ground-truth) solution. Propose a method to encode the graph and problem as text (or image) to provide them as inputs to the LLM and compare the solutions obtained against the optimal. Report all the prompts and raw LLM outputs and provide a brief discussion about the ability of LLMs to solve graph problems.