PSC CSC19 : Knowledge tracing assistant for learning programming

Alan André Alexandre Ittah Abdelkayoum Kaddouri
Mounir Lbath Alexandre Paresy

2025 - 2026

Tuteur : Jill-Jênn Vie (INRIA, X)

1 Introduction

Given that knowledge tracing is a rapidly evolving area in computer science, we aim to develop and evaluate methods for an intelligent assistant for learning programming. Our focus will be on teaching competitive programming.

In competitive programming, participants tackle algorithmic problems by designing and implementing efficient solutions that produce correct outputs for given inputs, all while complying to strict time and memory constraints. Submissions are evaluated automatically through a series of tests, allowing both errors and performance to be tracked with precision.

For example, such a problem could be: given a set of points given in the input, output a simple polygon with the initial points as vertices (that is, output an ordering of the points such that the resulting path does not cross itself):

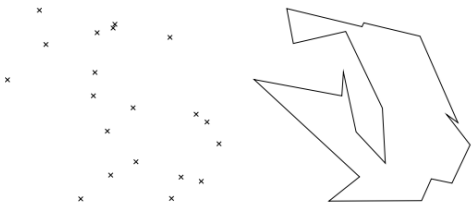

Figure 1: An example of a competitive programming problem.

Our approach to this project is guided by three complementary and multifaceted objectives:

- Generate a domain representation of competitive programming
- Experiment on different ways to store the knowledge and abilities of each student

- Create and evaluate methods to help student progress while taking into account their current abilities

This project offers the opportunity to explore the intersection of learning science, computer science and AI from a theoretical and practical perspective under the guidance of an INRIA researcher.

2 Purpose and Motivation for this Project

Preparing for programming competitions can help learners develop abilities such as algorithmic thinking, problem decomposition, debugging, and managing constraints under time pressure. These skills often extend beyond competitions and prove useful in academic and professional settings.

Although human tutoring plays an important role, it is difficult to scale for the growing number of learners. As a result, extensive efforts have been directed toward the development of intelligent tutoring systems, notably for teaching programming [1]. Our goal is to design automated support that prevents students from remaining blocked on minor or recurring errors and offers a clearer structure to the learning process.

We focus on programming competitions because they offer extensive collections of problems across different levels and topics. Such problems are well suited to automated learning environments, since online judges can evaluate solutions quickly and reliably. This setting also creates opportunities to experiment with Artificial Intelligence tools, including LLMs and classifiers, in multiple ways.

Learning in programming is often reflected as much in the mistakes as in the solutions. Errors can reveal valuable information for predicting future performance and guiding personalized feedback. Traditional knowledge-tracing methods, which represent domains through structures like prerequisite graphs of knowledge components, may also support adaptive recommendations to strengthen learning outcomes.

In this sense, competitive programming is not only engaging for learners but also a promising testbed for educational AI.

3 State of the art

Educational research and intelligent tutoring systems cover a broad range of methods and models aimed at understanding how people learn and how technology can best support them. We have exposed here a brief summary of the founding and most common methods in this field that will help us throughout our project.

Knowledge Tracing (KT) is the process of modeling and tracking a learner's progression over time.

This can be assessed through the learner's mastery of certain knowledge components. A **Knowledge Component** (KC) is a specific unit of knowledge or skill that a learner needs to master in order to solve a problem or perform a task. In educational research

and intelligent tutoring systems, KCs are used to break complex domains into smaller, measurable pieces that can be taught, practiced, and assessed.

There are 3 main types of knowledge tracing models:

- Bayesian models
- Logistic models
- Deep learning models

3.1 Bayesian models

One of the earliest and most widely used KT model is called **Bayesian Knowledge Tracing** (BKT) [2]. In BKT, we focus on one specific KC and estimate the probability $\mathbb{P}_n(\textbf{KC learned})$ for it to be mastered at time n given the sequence $(c_i, \delta_i)_{1 \leq i \leq n} \in (\{0,1\}^2)^n$ of the learner's answers c_i paired with the correct answers δ_i. It relies on these 3 rules:

1. The Knowledge Component is modeled as either learned or not learned.
2. A Markov model updates the probability a learner has mastered a KC based on their sequence of correct/incorrect answers.
3. Four key parameters (initial knowledge, learning rate, guess, slip) govern how mastery is estimated over time.

This model, despite being very powerful, offers some limitations. It models knowledge in a binary way—either mastered or not—without accounting for partial understanding. Its parameters for guessing, slipping, and learning are fixed and do not adapt to individual learners or changing contexts. Moreover, BKT treats each knowledge component independently, overlooking relationships and dependencies among skills. Finally, it relies only on sequences of correct or incorrect answers, leaving out richer information such as time spent, problem difficulty, or learning strategies.

3.2 Logistic models

An alternative to Bayesian approaches is offered by logistic models, which view knowledge tracing as a regression problem. Instead of representing a learner's knowledge as a hidden binary state, logistic models estimate the probability of a correct response directly from observable features.

3.2.1 Performance Factor Analysis

One of the most influential methods in this family is **Performance Factor Analysis** (PFA) [3]. In PFA, the likelihood of answering correctly the item for student i is modeled as a logistic function of the number of previous successes and failures a learner has accumulated on a given KC:

$$\mathbb{P}(\textbf{correct}_i) = \sigma(\beta + \sum_{j \in KCs} (\gamma_j s_{ij} + \rho_j f_{ij}))$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic function and

- β is the global bias term (the higher it is, the easier the problem is)
- KCs is the set of KCs involved in the exercise
- γ_j is the learning gain weight per success for skill j
- ρ_j is the learning gain (or penalty if negative) weight per failure for skill j
- s_{ij} number of prior successes student i had on knowledge component j
- f_{ij} number of prior failures student i had on knowledge component j

This formulation captures the intuition that practice through both correct and incorrect attempts contributes to learning, with different weights assigned to successes and failures.

The model can then be fit using standard logistic regression using a batch of students' learning history.

Compared to BKT, logistic models avoid some of BKT's limitations by moving away from binary knowledge states and allowing for a continuous representation of mastery (we predict the probability of getting a correct answer rather than the proability of the KC being learned or not). However, like BKT, they often treat knowledge components as independent and may fail to capture complex temporal dynamics in student learning. Moreover the parameters, once learned with a pre-determined training set, are fixed.

3.2.2 Elo-based Knowledge Tracing

Another logistic approach is based on Elo [4]. Elo ratings are very popular in chess and competitive programming (notably on *Codeforces*, a website hosting competitive programming contests).

Elo also uses a logistic link, but the parameters are updated online instead of estimated via regression. This time, the probability of student i mastering the KC j is calculated by:

$$\mathbb{P}(\mathbf{correct}_{i,j}) = \sigma(\theta_i - \beta_j)$$

where

- θ_i is the student skill parameter (his "Elo rating")
- β_j is the KC's difficulty
- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic function

after each response, the parameters are updated much like Elo ratings:

$$\theta_i \leftarrow \theta_i + K \cdot (r - \mathbb{P}(\mathbf{correct}_{i,j}))$$

$$\beta_j \leftarrow \beta_j - K \cdot (r - \mathbb{P}(\mathbf{correct}_{i,j}))$$

where

- $r \in \{0, 1\}$ is the observed response (incorrect/correct)
- $K > 0$ is the learning rate

Elo knowledge tracing is fast and adaptive, updating after each response, but it's simplistic and ignores practice history. PFA models prior successes and failures for smoother learning curves, but usually needs batch training and isn't as real-time as Elo.

3.3 Deep Learning models

Cognitive processes are influenced by numerous factors operating at both macro and micro levels. Traditional approaches, such as Bayesian or logistic models, often struggle to capture the complexity of these processes. In contrast, deep learning, with its strong capacity for non-linear representation, offers a powerful alternative, particularly when large-scale data on learner interactions is available.

3.3.1 Deep Knowledge Tracing

One of the first approaches to apply deep learning to knowledge tracing is **Deep Knowledge Tracing (DKT)** [5]. It models a learner's sequence of interactions using a Recurrent Neural Network (RNN), which maintains a hidden state summarizing past knowledge. At each time step, the hidden state is updated from the previous state and the current input, which gives the output prediction.

DKT provides a high-dimensional, continuous representation of student knowledge and outperforms traditional Bayesian or logistic models. However, it has some limitations: the hidden state is difficult to interpret, it cannot always reconstruct observed inputs, and predictions may be inconsistent across time steps. Despite these drawbacks, DKT remains a foundational model for applying deep learning to knowledge tracing.

There are also more advanced deep learning models, such as Memory-Aware KT, Attentive KT, and Graph-Based KT, that are applied to various extensions depending on the different types of student learning behaviors, interaction patterns, or knowledge structures being modeled.

4 Intermediary objectives

Our project is ambitious, so splitting it into subtasks is essential to keep track of our progression.

First, we aim to explore AI models and datasets (Codeforces dataset from Hugging Face, Google DeepMind's AlphaCode contest dataset, ...) to get a sense of what is currently possible and what may be limited.

Once a domain representation of competitive programming is drafted, we may use it to develop methods to estimate a student's knowledge and to suggest problems or courses that fit a student's knowledge level. This would likely involve estimating what has already been learned and adapting assessments accordingly. While something like Elo scoring

might provide a rough baseline, we intend to test different approaches and see which ones seem most promising for our case.

5 Task distribution and our working organization

To develop and evaluate methods to build an assistant for students in competitive programming, we aim to balance theory and practice. While our precise working organization is not entirely defined yet, here is the general line we will try to stick to.

On the theoretical side, we have to look at problem formats, knowledge tracing, and related models, which shape our dataset choices and experimental setup.

At the same time, implementation challenges will help adjust or question our starting assumptions, so ideas are regularly confronted with practice.

During our working sessions, we should combine reading scientific articles, brainstorming ideas, implementing them and evaluating them.

For task distribution, we first plan to work collectively to establish a common understanding of the project, before forming smaller groups, especially for practical tasks. Each group should regularly present its progress to the rest of the team in order to generate new ideas and receive detailed feedback. In addition, regular check-ins with our tutor will help us maintain consistency and ensure alignment with project goals.

References

[1] Kanav Mittal, Abigail O'Neill, Hanna Schlegel, Gireeja Ranade, and Narges Norouzi. Modeling student knowledge progression across concepts in intelligent tutoring interactions. In *Artificial Intelligence in Education (AIED)*, 2025.

[2] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 1994.

[3] Philip I Pavlik Jr, Hao Cen, and Kenneth R Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online submission*, 2009.

[4] Margit Antal. On the use of elo rating for adaptive assessment. *Studia Universitatis Babes-Bolyai, Informatica*, 2013.

[5] Chris Piech, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. *Online submission*, 2015.