Learning from Teaching Assistants to Formulate Subgoals for Programming Tasks: Exploring the Potential for AI Teaching Assistants

Changyoon Lee[1,†], Junho Myung[1,*], Jieun Han[1,*], Jiho Jin[1,*] and Alice Oh[1]

[1]Korea Advanced Institute of Science Technology (KAIST), 291 Daehak-ro, Yuseong District, Daejeon, South Korea

Abstract

Active formulation of subgoals in problem-solving is an effective learning strategy for programming learners, allowing the transfer of knowledge across similar problems. Although proper guidance and feedback for learners are crucial to correct mistakes and misconceptions during subgoal formulation, providing them at scale is challenging and costly. With recent advances in generative AI, we investigate the practicality of using generative AI as TAs in programming education by examining their effectiveness in a subgoal learning environment. We explore whether programming learners can distinguish AI TAs from humans. In a long-term study, we explore whether the subgoal learning workflow with AI TAs yields learning gains and assess their capability to assist in coding the subgoals into executable programs. Our study shows that learners can distinguish AI TAs from human TAs based on response length and accuracy. Learners show learning gains over learning sessions with AI TAs in formulating subgoals and can produce code solutions faster with comparable satisfaction scores with AI TAs as human TAs.

Keywords

Generative AI, CS Education, Human-AI Interaction, Subgoal Learning

1. Introduction

Students taking introductory programming courses are expected to learn various programming concepts such as debugging, designing algorithms, techniques in programming, and computational thinking [1, 2]. Having to learn these new concepts in a single course presents difficulties to the learners [3], and if they are not appropriately alleviated, learners may lose motivation and even drop out of the course [4, 5]. Teaching assistants (TAs) play a crucial role in alleviating these difficulties by correcting learners' misconceptions and fixing errors in their code, enhancing their overall learning gain [6, 7, 8]. With a sufficient number of TAs, learners can receive individual care by getting help in solving programming tasks and clarifying programming concepts [9, 10], but TAs are costly.

With recent advances in generative AI and Large Language Models (LLMs), the educational field has discovered some exciting opportunities for assisting learners. In the context of programming education, recent large generative models such as ChatGPT[1], LLaMA [11], and Bard[2] show a remarkable ability to understand, generate, and explain code, making them strong candidates for TAs in programming courses [12]. They can fix and explain errors present in the code and discuss possible approaches to solve various programming tasks. AI coding assistants have been shown to relieve the cognitive load and struggles of learners, allowing them to perform better and faster in solving programming tasks [13].

We introduce the concept of subgoal learning to novice programming learners with the aid of AI TAs. Subgoal learning is well known to be an effective learning strategy in the STEM domain by helping students break down complex problems into smaller counterparts [14], which remains important for a programmer even with the aid of generative AI to code. Through a series of experiments, we observe the effect of our learning workflow and investigate how learners perceive and interact with the AI TAs.

We first determine how learners embrace AI TAs when they divide the task into subgoals, through which learners are expected to develop computational thinking skills. We observe learners' expectations of AI and human TAs and how learners differentiate them. In a month-long study where learners formulate subgoals for programming tasks with the help of AI TAs, we investigate the learning effect of AI TAs during the subgoal formulation exercise. Finally, we compare the AI and human TAs in a between-subject study with 20 novice programming learners. Learners solve 4 programming tasks with the aid of either an AI TA or a human TA. We assess AI's ability to help plan an algorithm and write the code for it within our learning workflow. We examine learners' perceptions of conversation satisfaction with the TA.

Although learners' expectations of the TAs' response time vary, they correctly anticipate that the AI may be occasionally inaccurate and produce lengthy responses. This allows nearly all learners to differentiate between AI and human TAs accurately. AI TAs show the capability to help learners produce more accurate and detailed subgoals over time and to review learner-generated subgoals in comparison to the correct subgoals. Learners assisted by the AI TA demonstrated faster problem-solving and attempted more tasks, achieving comparable scores for the assigned tasks. In the survey, learners reported that AI TA's replies were prompt, sufficiently detailed, and helpful throughout the workflow. Moreover, learners were satisfied with the conversation with the AI and perceived that it was generally uncomplicated and helpful for learning programming. However, the AI's tendency to offer answers and occasionally break down calls for careful consideration before its deployment.

Educational Datamining '24 Human-Centric eXplainable AI in Education and Leveraging Large Language Models for Next-Generation Educational Technologies Workshop Joint Proceedings, July 13, 2024, Atlanta, GA

*Corresponding author.

†These authors contributed equally.

changyoon.lee@kaist.ac.kr (C. Lee); i.tiddi@vu.nl (J. Myung); jieun\protect\TU_han@kaist.ac.kr (J. Han); jinjh0123@kaist.ac.kr (J. Jin); alice.oh@kaist.edu (A. Oh)

[1]https://chat.openai.com/

[2]https://bard.google.com/

2. Related Works

2.1. Subgoal Learning

Subgoal learning is a method designed to assist students in breaking down complex problem-solving procedures into smaller structural components within the STEM domain [14]. In the context of programming education, subgoal learning is known to help reduce the extraneous cognitive load of the learners, thereby enhancing their problem-solving performance [15, 16].

The effectiveness of subgoal learning is further amplified when implemented as an active learning strategy. The passive learning approach was found to be less effective compared to self-directed learning methods, which involve self-reflection and explanation of the hierarchical structure of the solutions [17, 18]. Yet, proper guidance or feedback is necessary to correct learners' misconceptions of the concept when creating subgoals by themselves [18, 19]. With the advances of AI models, guiding learners in the process of self-labeling subgoal tasks with AI has become possible. However, no previous work has explored such applications.

2.2. Generative AI for Programming Education

Generative AI exhibits remarkable performance in various programming tasks, such as code summarization [20], code generation [21, 22], and even code explanation [23]. This recent advancement in generative AI opens up numerous opportunities to support programming education. Novice learners can gain a deeper understanding of basic programming concepts with line-by-line code explanations generated by LLMs [24]. They can also receive feedback and detect bugs before they submit their assignments for grading [25]. The nearly instantaneous provision of feedback and explanations makes generative AIs more accessible and convenient for learners than human instructors. Yet, to the best of our knowledge, examining the performance of LLM compared to human TA has been underexplored, and existing literature exploring how to optimally leverage LLMs as TAs is only emerging, especially in programming education.

3. Methods

While generative AI can nearly produce perfect code for a program given its descriptions [21, 22], learning computational thinking, designing the solution, and understanding and debugging code remain important for a programmer [1]. Thus, we design a learning workflow that focuses on helping learners practice computational thinking and planning out the solution to a programming task with the help of an AI TA.

In our learning workflow, novice learners with little to no programming experience break down a programming task into smaller and more manageable subgoals with the aid of a TA via a chatting interface. Learners are provided with the task description that includes the task requirements, sample inputs, and sample outputs. Learners converse freely with the TA to develop subgoals for the task. After they formulate the subgoals, they review their subgoals by comparing them with the model answers for the subgoals. Within this learning workflow, we conduct three sets of experiments to answer three research questions:

1. How do learners embrace the AI TA and what are their expectations of AI and human TAs?
2. Can the AI TA help learners achieve learning gains in subgoal learning tasks?
3. How does the AI TA compare to the human TA in helping to formulate subgoals and solve programming tasks?

3.1. Participants

We recruited participants in their 20s and 30s who reside in Korea with little to no experience in programming by posting advertisements in online university communities. Participants self-reported their proficiency in programming on a 5-point Likert scale. We selected participants with reported proficiency levels of 3 and below for our experiments for research questions 2 and 3, which measure learners' abilities in solving programming tasks. All participants used Korean for communication.

3.2. Generative AI-powered TA

We used the latest ChatGPT at the time of the experiment (gpt-3.5-turbo-1106 for experiments 1 and 2, and gpt-3.5-turbo-0613 for experiment 3) as the model behind the AI TA since ChatGPT can handle conversational text data and performs better on programming when prompted in non-English language compared to other models such as Codex [26]. The default temperature of 1.0 was used. In all the experiments, the task description was provided to the model, and the model was prompted not to directly provide answers, i.e. the correct subgoals or code, as it can interfere with learning by removing the opportunity for the learners to practice producing the answers independently. The full prompt texts can be found in Appendix **??**

3.3. Experiment 1: Learners' Expectations of AI and Human TAs

It is important to understand if learners can distinguish AI TAs and how they respond differently to TAs when using AI TAs for education. We explore if learners can distinguish between AI and human TA in the subgoal formulation activity based on their expectations of each TA's behavior, and if they can, what characteristics separate the two. We recruit 12 learners who formulate subgoals for six programming problems with the help of either a human or AI TA, unaware of which TA is aiding them. Half of the learners are assigned to AI TA, while the other half are assigned to human TA. The recruited human TA has four years of experience as TAs in computer science courses. The TAs are instructed to help the learners create subgoals by providing hints, reiterating the subgoals, and giving feedback on the subgoals. To ensure the distinction between the AI and human TA isn't too obvious, we instruct the human TA to respond with only one message at a time, and we add a 500-millisecond delay per word for the AI TA's response. The one-message restriction is designed to impose as little restriction as possible on the human TA while preventing the students from distinguishing the two types of TAs simply by the number of messages. After the subgoal formulation activity, learners participate in a survey that asks which TA they think helped them, which of the speed, length, accuracy, style of the response, or another factor leads them to their choice, and the explanation for their choice.

Category (Score)	Definition
Prestructural (1)	Subgoals are copies of the task description or represent incorrect interpretations of the task.
Unistructural (2)	Subgoals represent the correct approach to the task, but include significant errors or major details are missing.
Multistructural (3)	Subgoals can form correct solution to the task, but includes minor errors or unnecessary steps that disturbs the program structure.
Relational (4)	Subgoals form nearly correct solution without error and the structure is solid, but small details are missing.
Extended (5)	Subgoals are perfect and detailed, and represents programming concepts such as loops and conditionals.

Table 1
Adaptation of SOLO taxonomy to computing education. Definitions are adapted to fit subgoal formulation for problem solving.

3.4. Experiment 2: Learning Gains from Subgoal Learning Workflow with AI TA

We conduct a four-week experiment to assess the AI TA's efficacy in assisting learners to generate a more comprehensive and accurate set of subgoals within our learning workflow. We recruit 20 novice learners who are given three programming tasks per session over 8 sessions. In each session, learners formulate subgoals for the given task with the aid of the AI TA. We design the experiment such that the programming tasks extracted from a crowdsourcing platform[3] increase in difficulty from the first session to the subsequent sessions to present challenges to the learners over time.

The AI TA is directed to offer hints to the learner, compile learner-generated subgoals in each response, and correct inaccuracies until the subgoals are deemed sufficient to solve the task and meet the learner's satisfaction. We create model answers for the subgoals by instructing ChatGPT to formulate the subgoals for the programming tasks. The authors review and revise the generated subgoals to ensure accuracy and consistent formatting. ChatGPT's performance in generating the subgoals is decent, and the authors mainly only had to split long subgoals into smaller ones and add subgoals that will benefit learning for 5 out of 24 problems. These subgoals are provided to the AI TA in the second step, where the AI TA reviews the learner-generated subgoals by describing why subgoals in the answer are necessary and comparing them to the learner-generated subgoals. The second step shows the correct answers to the learners and revise their own answers.

Four authors grade all learner-generated subgoals on a scale of 1 to 5 based on our adaptation of the SOLO taxonomy for subgoal labels in CS1 which allows for a deeper evaluation of the subgoals in terms of completeness and understanding of the relevant concepts [27]. The rubrics for grading the subgoals are provided in Table 1. The scores for the subgoals are compared as the learners progress through the sessions.

3.5. Experiment 3: Comparison between AI and Human TA for Solving Programming Tasks

We conduct a between-subject experiment with novice programming learners to compare the learning effects gained with the AI TA and human TA. We recruit 20 learners of which 10 participants have a self-reported proficiency of 1 and have not taken any computer science course (Group 1). The remaining 10 participants have a self-reported proficiency of 2 and have taken only introductory courses in computer science (Group 2).

We randomly assign 5 participants in each proficiency level to solve the programming tasks with the help of the AI TA and assign the remaining 5 participants to solve with the help of a human TA. Four human TAs participate in the experiment. Learners solve each programming task in three steps: 1) **Subgoal Formulation step**, where learners break down the task into smaller and more manageable goals, 2) **Subsolution Generation step**, where learners tackle each of the subgoals they have formulated and implement a solution for them, and 3) **Solution Generation step** where learners combine their subsolutions into a single code that solves the programming task. The subgoal formulation step drives learners to understand and organize the task and devise a plan for the final solution. The subsolution generation step helps learners focus on a subgoal at a time and progressively write the program. The solution generation step allows learners to review their subsolutions and debug them.

Out of all participants, 19 participants completed both sessions. One participant who is assigned to the AI TA and has a self-reported proficiency of 2 dropped out of the study and only participated in the first session. We collect the time learners take to complete each task, the number of tasks they attempted, and the scores for the tasks. The scores are calculated by counting the number of test cases the learner's code successfully passes.

In an online survey, learners are asked to rate their experience with the TA and the helpfulness of the learning workflow and the system. All survey questions are rated using a 7-point Likert scale (1: strongly disagree, 7: strongly agree) followed by an open-ended question asking the reason behind the choice.

4. Results

4.1. Experiment 1

Of the 12 learners, 11 correctly identified which TA helped them in the subgoal formulation task. One learner mistook the human TA for the AI TA and suspected a system where the human TA and the AI TA take turns answering the questions. All learners who correctly identified the human TA picked the accuracy of the response as the deciding factor of the TA's identity. Learners expected the human TA to always answer the learners' questions accurately, even when the question does not directly address the task. This shows that learners expect human TAs to be able to understand the learner's status and respond in context. The learner who mistook human TA as AI TA also thought so because the learner thought the TA gave wrong feedback to the learner's subgoals. There was no significant difference in the interaction and the conversation style for this learner, possibly because the learner thought a human TA was taking turns to reply.

The main reason learners thought they were talking with an AI TA was the tone of the TA's responses. The AI TA's responses were generally longer and carefully formatted,

[3]https://solved.ac/

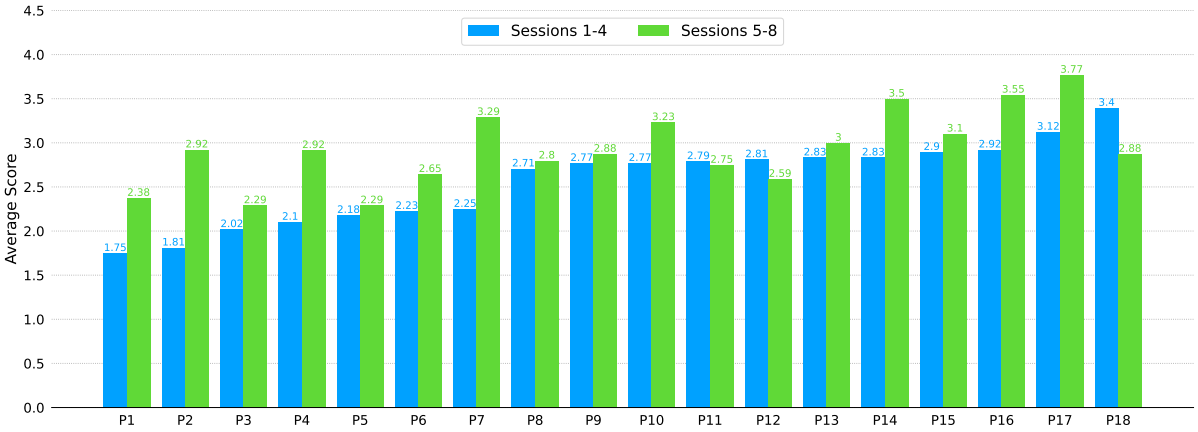

Figure 1: Mean scores of subgoals for each learner in sessions 1 to 4 and sessions 5 to 8. Learners are numbered from P1 to P18. All learners except P11, P12, and P18 formulated higher-scoring subgoals during the second half of the sessions.

which learners expected as the characteristics of generative AI. Additionally, the AI TA sometimes failed to catch the intention of the learner's question and responded out of context. Learners expect AI TA to be imperfect and have a formal tone in its response. Learners, on the other hand, had a divided opinion on the response speed. One learner assigned to the human TA and three learners assigned to the AI TA identified the slow response time as a reason for their choice. One other learner assigned to the AI TA identified the AI TA based on its fast response time. This shows that learners have varying expectations of the human and AI TAs' response speed.

4.2. Experiment 2

Among 20 learners, 18 participated in all eight sessions in the experiment. We only report the scores for those learners in the results. The authors underwent a grade norming session for the subgoals learners generated in the first session to increase consistency in grading the subgoals. The authors then independently graded all subgoals based on our adaptation of the SOLO taxonomy. The mean value of the scores authors gave for each set of subgoals was taken as the final score for the task. Figure 1 shows the mean scores of the subgoals each learner formulated for sessions 1 to 4 and sessions 5 to 8. The results show that all but three learners had higher scores for the second half of the sessions compared to the first half. The learners learned to formulate more structured and detailed subgoals over time with the AI TA, even when the difficulties of the programming tasks increased across the session.

In a post-experiment survey, learners also indicated an average score of 5.7 on a 7-point Likert scale in response to a question assessing the perceived helpfulness of the AI TA in learning programming. Learners reported that the AI TA's hints and thorough explanations allowed them to get used to the subgoal learning process and learn about new approaches to solving the task.

4.3. Experiment 3

4.3.1. Performance Measures

We first compare the task completion rate, which we define as the percentage of tasks a learner attempted and produced a code solution within 3 hours. The completion rates for each task for Groups 1 and 2 are shown in Figure 2.

Learners in both groups produced code solutions for most of the programming tasks. Learners who solved the tasks with the AI TA showed higher or equal completion rates for all the tasks across the two groups, although there was no statistical significance from an independent samples t-test with an alpha level of 0.05. Learners who solved the tasks with a human TA showed a sharp decrease in completion rates for the later tasks in a session.

We report the average time taken to finish each task in Figure 3 for learners in Groups 1 and 2. Results show that the learners completed the tasks faster with the AI TA than with the human TA. The difference is more evident with Group 1 learners who had no experience in programming before. The average time taken to finish solving a task for Group 1 learners was 31.5 minutes for those with AI TA and 59.8 minutes for those with human TA. The time taken for Group 2 learners to finish solving the tasks was 21.5 minutes for those with AI TA and 38.9 minutes for those with human TA. The difference in the average time taken between the two types of TAs in both Group 1 and Group 2 is statistically significant from a t-test, with P-values of 0.000096 and 0.0044, respectively.

We test the correctness of the learners' solution code for each programming task by comparing its output for 10 test cases with the correct answer. Figure 4 shows the average score for each task in the percentage of test cases passed.

Learners who solved the task with AI TA showed higher or equal scores than those with human TA in the first session of the user study for both proficiency groups, although the differences did not reach statistical significance. The scores in the second session show mixed results. The average score for all tasks for learners was 59.5 with the AI TA and 50.25 with the human TA for Group 1. The average score for all tasks for learners was 71 with the AI TA and 53 with the human TA for Group 2. The differences did not reach statistical significance. Both the AI and the human TAs helped the learners achieve similar scores for the programming tasks over the two sessions of user study.

4.3.2. Perception of the TAs

The learners' responses to the survey questions are summarized in Figure 5 in the Appendix.

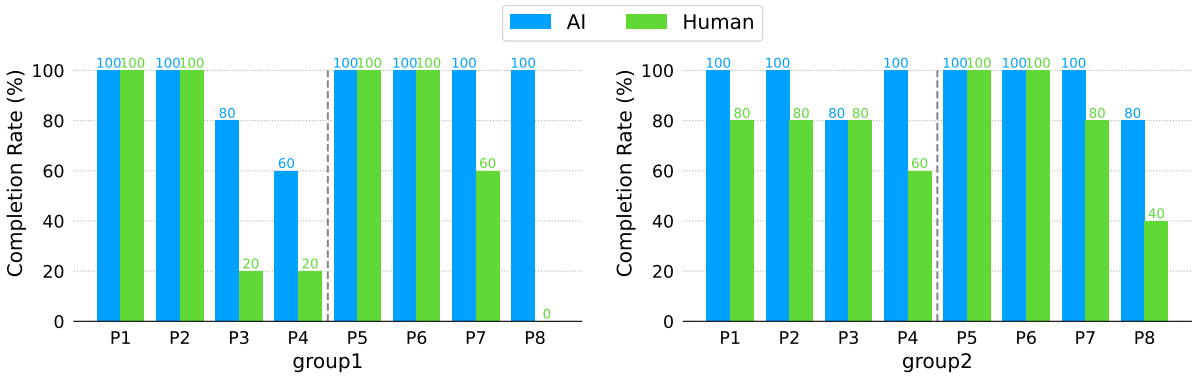

Figure 2: Mean completion rate for programming tasks. P1 to P4 are tasks given in the first session and P5 to P8 are tasks given in the second session.

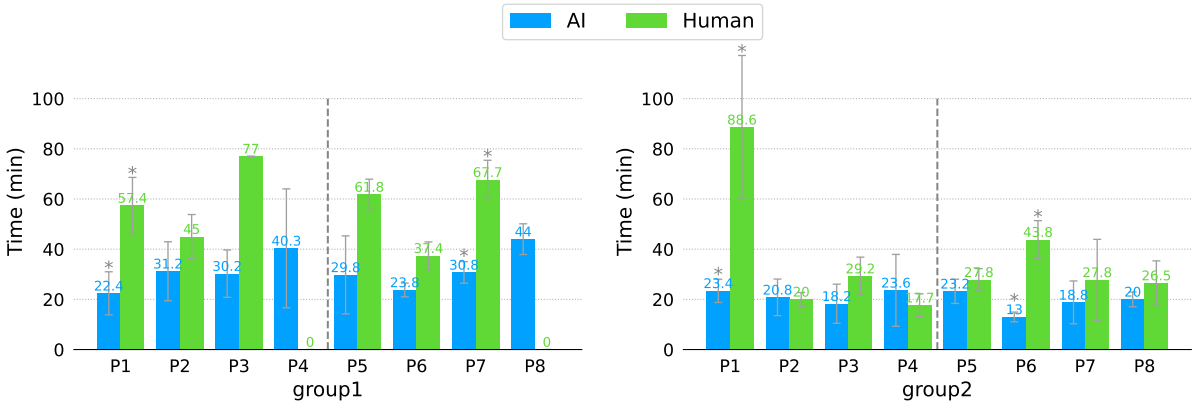

Figure 3: Average time taken to complete solving the programming tasks for the two groups of learners. Tasks with significant differences between the TA types are indicated with the asterisk (*). Group 1 learners took 31.5 minutes with AI TA and 59.8 minutes with human TA on average. Group 2 learners took 21.5 minutes with AI TA and 38.9 minutes with human TA on average.

Learners in both groups were generally satisfied with the promptness of the TA's replies regardless of the TA type. There was no statistically significant difference in the learners' perception of TA's promptness between the TA types. However, one learner mentioned that the human TA's replies were too slow.

Learners generally rated the usefulness of both TAs' replies positively in all of the three steps of the user study. However, some negative remarks in the subgoal formulation stage mentioned that the AI TA did not provide detailed explanations for the difference between the subgoals and dismissed a subgoal that the learner felt was appropriate as unnecessary.

Learners perceived that the TAs effectively assisted them

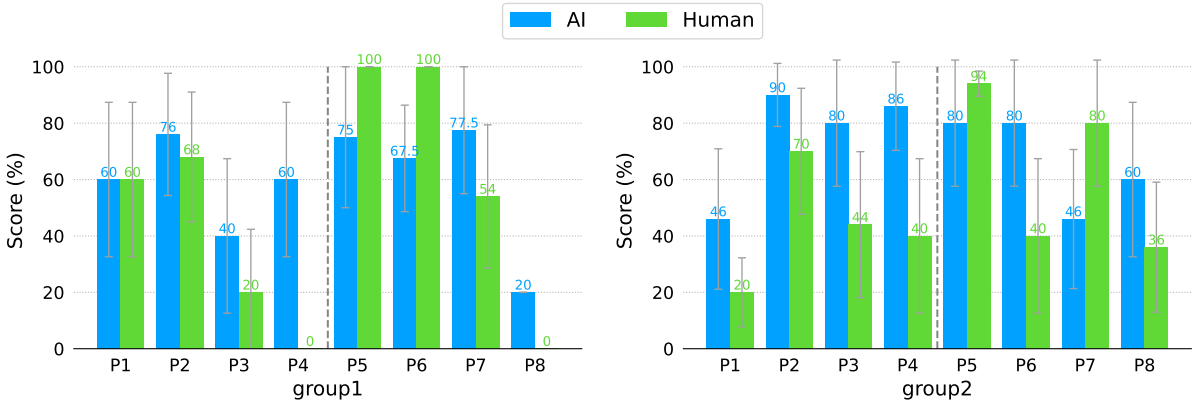

Figure 4: Scores for the programming tasks for the two groups of learners. Group 1 learners had a score of 59.5 with AI TA and 50.25 with human TA on average. Group 2 learners had a score of 71 with AI TA and 53 with human TA on average.

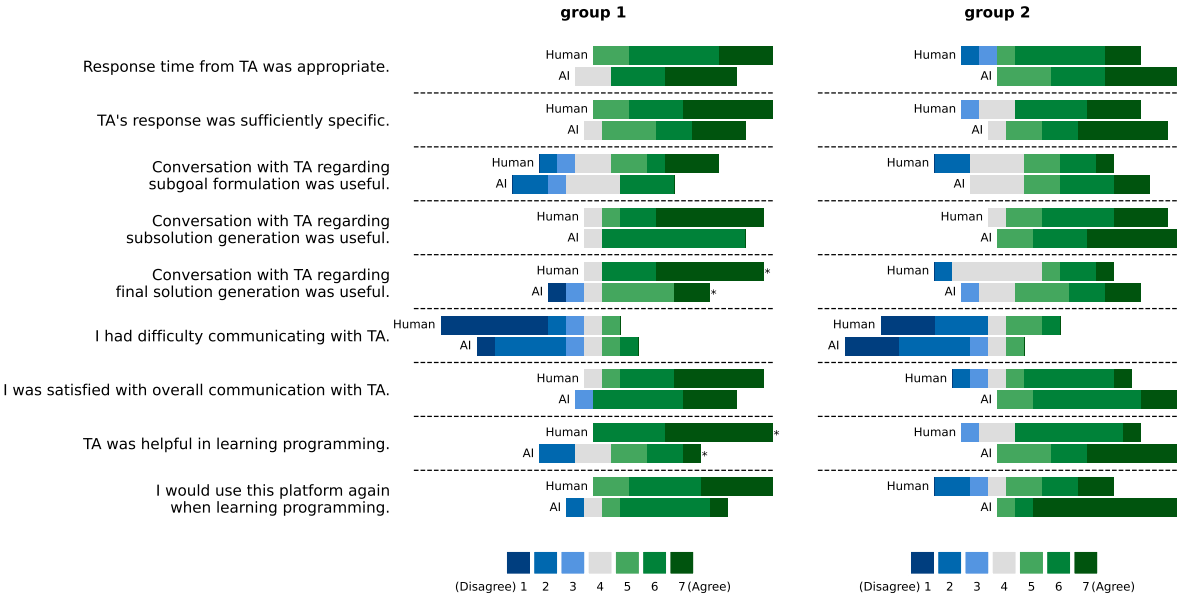

Figure 5: Summary of the survey results on the perception of the TAs. Scores for questions that show a statistically significant difference between the two types of TAs are marked with the asterisk (*).

in translating subgoals into subsolutions. Learners mentioned that the AI TA explained the necessary functions with examples thoroughly, provided detailed feedback, and debugged the code even when the question was vague. The human TA was also helpful for debugging.

Interacting with TAs, whether AI or human, was generally effortless for learners. However, some learners felt that the conversation could be improved. Learners encountered difficulty communicating with the human TA due to a lack of familiarity with basic programming syntax. Some felt intimidated, fearing they were posing what they perceived as 'stupid' questions. On the other hand, when talking to an AI TA, the AI TA often produced unexpected responses unrelated to the question, demonstrating a lack of understanding of the learners' inquiries.

Group 1 learners with the AI TA reported that without basic programming knowledge, the AI TA allowed the learners to solve the tasks, but they were unsure whether they had picked up knowledge in the process. Group 2 learners felt that the AI TA was helpful, as the TA taught the learners new ways to solve the problem and how to write concise code. Some learners appreciated the freedom to learn without time constraints when using AI. Others reported that having a TA enhanced efficiency and made the learning experience enjoyable.

4.4. Chat Log Analysis

There were slightly more learner utterances with the human TA than with the AI TA. In experiment 3, learners had, on average, 2.3 and 12.60 utterances with the human TA in the subgoal formulation and subsolution generation steps, respectively. Learners had, on average, 2.71 and 8.60 utterances with the AI TA in the same steps. Learners mostly asked questions in the subsolution generation step where they had to write working programs. In the subgoal formulation step, learners mostly only clarified their subgoals with the TAs.

The ideal role of the TA in subgoal formulation involves offering feedback on the learner's subgoals, including refining abstract subgoals into more concrete and executable ones. We observed that the AI TA provides more feedback for the subgoals with more detail to the learners. In contrast, human TAs preferred allowing learners to formulate subgoals independently, opting to provide feedback during the subsolution generation step. The AI TA sometimes provided the full set of subgoals voluntarily, while no such case was observed for the human TAs.

The AI TA also proactively offers the answer code more frequently than the human TA. In experiment 3, human TAs provided answer codes on only 10 occasions, whereas the AI TA provided answer codes on 40 occasions. Human TAs provided the code voluntarily only when the learner was stuck at a step for an extended duration. Also, they offered the code when the programming concept was difficult to explain only in words, such as when explaining the formatted printing statement.

Learners asked for the code more often with the AI TA (135 occasions) than with the human TA (46 occasions). Human TAs often refused to provide the answer code directly when the learner asked for help; they tried to explain the syntax or algorithm in words first, allowing the learner to develop the code independently. The AI TA, on the other hand, provided the code nearly always on the learner's request. Therefore, learners with the AI TA might have been more inclined to ask for the answer code to solve the tasks faster.

5. Discussion

5.1. Feasibility of Using Generative AI as a TA

Our results show that generative AI is capable of assisting as TAs in teaching introductory programming with subgoals. Learners can discern when assisted by an AI TA, yet the interaction with the AI TA is generally satisfactory for the majority of learners. Learners demonstrate the ability to

formulate subgoals and write code for programming tasks with comparable proficiency when assisted by the AI and human TAs. Learners exhibit learning gains in subgoal formulation when working with the AI TA, demonstrating an improvement in the quality of their subgoals over a 4-week period. Nearly all learners formulated higher-quality subgoals in the second half of the experiment, even though the tasks were more difficult.

Learners' perception of the AI TA is generally positive and on par with that of the human TA in several aspects. Learners feel that the AI TA's responses are fast and detailed enough to help them solve the programming tasks. Group 2 learners show a more positive perception of the AI TA compared to Group 1 learners, even exceeding the perception of the human TAs by the learners in the same group. This shows that the AI TA is better suited for programming learners who already possess some prior programming knowledge, while absolute beginners find it more challenging to communicate and learn with an AI TA.

5.2. Strengths and Weaknesses of AI TA and Usage Guidelines

An evident strength of the AI TA lies in its capacity to furnish detailed responses to learners' questions, offering a substantial amount of information when compared to the human TA. The AI TA leaves thorough feedback on individual subgoals, providing the reason why the subgoal is essential within the context of the task. The AI TA's replies are more structured, reiterating the learner's question and providing the answer with an explanation for the answer. Such structured replies can be beneficial for learning as the learner is reminded of the full context of the problem and how to solve it.

However, the AI frequently fixates on a single mistake in a conversation, persistently highlighting it even when the learner addresses and corrects the error, leading to repeated responses. This often made the learners frustrated and the learners had to complete the task on their own.

The main weakness of AI TA in the educational field lies in providing excessive information to the learners. The AI TA seems to be oriented to help the learner solve a task rather than focus on the educational benefits of the learners. While assistance is valuable, providing too many hints or code can impede the learning process by depriving learners of the opportunity to solve the problem on their own. This aligns with the perception of Group 1 learners who find the TA less helpful in learning programming; the AI TA's tendency to offer the answer code too frequently may hinder independent coding engagement for learners.

This behavior could have influenced the learners' interaction patterns with the TAs. With the human TA, learners asked questions about the code indirectly by describing issues in their code and stating their intentions. In contrast, learners ask for code assistance from the AI TA directly by asking for explanations about their code issues and how to address them. As the AI TA provides more direct assistance close to the answer, learners with the AI TA might have completed the tasks faster, consequently resulting in higher completion rates compared to the learners with the human TA.

One way to prevent this is to append an additional text at the end of **every** learner's prompt that explicitly requests the AI not to provide the answer unless it is absolutely necessary. However, completely prohibiting the AI TA from providing some form of solution might lead to learner's frustration.

For complete beginners in programming, human TAs can be better suited to guide them in solving programming tasks. The human TA is better able to understand the learner's struggles and is more attentive to the small details in programming that beginners have to pay attention to. When testing the code, human TAs are better at catching the edge cases and removing rare errors in the code. In our experiment, human TAs ensured that learners passed all the test cases by guiding them attentively to produce the correct solution and debugging the code. Although AI TAs could help learners achieve high scores for most tasks, they were insufficient to help learners achieve perfect scores.

5.3. Ethical Considerations and Limitations

All experiments involving human subjects were reviewed and approved by the institute's Institutional Review Board. All participants in the experiment were paid a reasonable amount meeting the minimum wage requirements as compensation.

Despite the authors' best attempts to measure learning gains with the AI TA by analyzing scores and survey responses, the positive results might be a direct result of the explicit help of the TAs and do not necessarily represent long-term learning gains. Future work could explore whether learners are able to solve new tasks without the help of the TAs after the learning sessions.

6. Conclusion

The advances in generative AI have opened the opportunity for AIs to take the role of teaching assistants in programming. We explore the potential for AI teaching assistants in a subgoal-learning environment to teach computational thinking and writing code to a programming novice and how the learners interact and perceive the AI as teaching assistants. Our findings indicate positive learning gains in subgoal formulation when learners engage with AI TAs. AI TAs can assist learners in achieving comparable performance in scores and task completion time as human TAs by offering exceptionally detailed explanations. Learner's perception of the AI TA is positive, especially for learners with some previous experience in programming. Nevertheless, the AI TA's tendency to readily provide answers may lead to reduced educational benefits for learners. The AI also occasionally becomes fixated on a specific point in the conversation, entering an irrecoverable state and consequently leaving the learner to struggle with the task in isolation. While the AI TA demonstrates its value in programming education, integrating it into a real-world educational setting demands thoughtful consideration and control, extending beyond its out-of-the-box application.

Acknowledgments

This work was supported by Elice.

References

[1] B. A. Becker, T. Fitzpatrick, What do cs1 syllabi reveal about our expectations of introductory programming students?, in: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1011–1017.

[2] J. M. Wing, Computational thinking, Communications of the ACM 49 (2006) 33–35.

[3] M. Konecki, Problems in programming education and means of their improvement, DAAAM international scientific book 2014 (2014) 459–470.

[4] N. Rountree, J. Rountree, A. Robins, Predictors of success and failure in a cs1 course, SIGCSE Bull. 34 (2002) 121–124.

[5] P. Kinnunen, L. Malmi, Why students drop out cs1 course?, ICER '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 97–108.

[6] A. T. Corbett, J. R. Anderson, Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '01, Association for Computing Machinery, New York, NY, USA, 2001, p. 245–252.

[7] L. Gusukuma, A. C. Bart, D. Kafura, J. Ernst, Misconception-driven feedback: Results from an experimental study, in: Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 160–168.

[8] S. Marwan, G. Gao, S. Fisk, T. W. Price, T. Barnes, Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science, in: Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 194–203.

[9] D. Mirza, P. T. Conrad, C. Lloyd, Z. Matni, A. Gatin, Undergraduate teaching assistants in computer science: A systematic literature review, in: Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 31–40.

[10] E. Riese, M. Lorås, M. Ukrop, T. Effenberger, Challenges faced by teaching assistants in computer science education across europe, in: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 547–553.

[11] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, 2023. arXiv:2302.13971.

[12] J. Savelka, A. Agarwal, M. An, C. Bogart, M. Sakr, Thrilled by your progress! large language models (gpt-4) no longer struggle to pass assessments in higher education programming courses, arXiv preprint arXiv:2306.10073 (2023).

[13] M. Kazemitabaar, J. Chow, C. K. T. Ma, B. J. Ericson, D. Weintrop, T. Grossman, Studying the effect of ai code generators on supporting novice learners in introductory programming, in: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23, Association for Computing Machinery, New York, NY, USA, 2023. doi:10.1145/3544548.3580919.

[14] R. Catrambone, K. Holyoak, Learning subgoals and methods for solving probability problems, Memory & cognition 18 (1990) 593–603. doi:10.3758/BF03197102.

[15] L. E. Margulieux, M. Guzdial, R. Catrambone, Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications, ICER '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 71–78. doi:10.1145/2361276.2361291.

[16] R. K. Atkinson, S. J. Derry, A. Renkl, D. Wortham, Learning from examples: Instructional principles from the worked examples research, Review of educational research 70 (2000) 181–214.

[17] B. B. Morrison, L. E. Margulieux, M. Guzdial, Subgoals, context, and worked examples in learning computing problem solving, in: Proceedings of the Eleventh Annual International Conference on International Computing Education Research, ICER '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 21–29. doi:10.1145/2787622.2787733.

[18] L. E. Margulieux, R. Catrambone, Finding the best types of guidance for constructing self-explanations of subgoals in programming, Journal of the Learning Sciences 28 (2019) 108–151. doi:10.1080/10508406.2018.1491852. arXiv:https://doi.org/10.1080/10508406.2018.1491852.

[19] H. Jin, M. Chang, J. Kim, Solvedeep: A system for supporting subgoal learning in online math problem solving, in: Extended abstracts of the 2019 CHI conference on human factors in computing systems, 2019, pp. 1–6.

[20] T. Ahmed, P. Devanbu, Few-shot training llms for project-specific code-summarization, in: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, 2022, pp. 1–5.

[21] S. Chakraborty, T. Ahmed, Y. Ding, P. T. Devanbu, B. Ray, Natgen: generative pre-training by "naturalizing" source code, in: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2022, pp. 18–30.

[22] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, et al., Codexglue: A machine learning benchmark dataset for code understanding and generation, arXiv preprint arXiv:2102.04664 (2021).

[23] E. Chen, R. Huang, H.-S. Chen, Y.-H. Tseng, L.-Y. Li, Gptutor: a chatgpt-powered programming tool for code explanation, arXiv preprint arXiv:2305.01863 (2023).

[24] S. MacNeil, A. Tran, A. Hellas, J. Kim, S. Sarsa, P. Denny, S. Bernstein, J. Leinonen, Experiences from using code explanations generated by large language models in a web software development e-book, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2023, Association for Computing Machinery, New York, NY, USA, 2023, p. 931–937. doi:10.1145/3545945.3569785.

[25] S. Sarsa, P. Denny, A. Hellas, J. Leinonen, Automatic

generation of programming exercises and code explanations using large language models, in: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1, ACM, 2022. URL: https://doi.org/10.1145%2F3501385.3543957. doi:10.1145/3501385.3543957.

[26] A. Hellas, J. Leinonen, S. Sarsa, C. Koutcheme, L. Kujanpää, J. Sorva, Exploring the responses of large language models to beginner programmers' help requests, in: Proceedings of the 2023 ACM Conference on International Computing Education Research V.1, ACM, 2023. URL: https://doi.org/10.1145%2F3568813.3600139. doi:10.1145/3568813.3600139.

[27] A. Decker, L. E. Margulieux, B. B. Morrison, Using the solo taxonomy to understand subgoal labels effect in cs1, in: Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 209–217. doi:10.1145/3291279.3339405.