GeoTree: A Dynamic Tree-based Geometry Problem Solver through LLM-Symbolic Reasoning

Yaxian Wang, Bifan Wei, Yinghong Ma, Lingling Zhang, Xudong Jiang, *Fellow, IEEE*, Henghui Ding, and Jun Liu, *Senior Member, IEEE*

***Abstract*—Geometry problem solving (GPS) requires high-level symbolic and logical reasoning based on geometry theorem knowledge to arrive at the answer. Despite the remarkable advances achieved by Large Language Models (LLMs) in various problem-solving tasks, they still struggle to perform rigorous multi-step geometry reasoning, which is essential for GPS. In this paper, we propose a dynamic tree-based geometry problem solver named GeoTree, which combines a knowledgeable LLM with a rigorous symbolic solver to perform geometry reasoning cooperatively. Specifically, an iterative multi-step geometry reasoning process is performed dynamically based on a tree-like structure, thereby emulating divergent and deliberate human problem-solving thinking. Each geometry reasoning step is completed collaboratively through four components, consisting of *Theorem Seeker*, *Symbolic Solver*, *Evaluator*, and *Controller*. First, *Theorem Seeker* prompts LLMs to seek out candidate theorems with their inherent geometry theorem knowledge. Subsequently, *Symbolic Solver* applies the theorems on the known conditions to obtain new additional conditions. Then, *Evaluator* assesses the availability of the theorems and prompts LLMs to judge the usefulness of these new conditions for the problem target, which serves as the heuristic guidance for subsequent reasoning. Finally, *Controller* determines the termination state, which decides whether to continue invoking the other three components for further attempts. Extensive experiments on Geometry3K demonstrate the superiority of GeoTree in accuracy, efficiency, and explainability.**

***Index Terms*—Geometry problem solving, large language model, symbolic reasoning.**

I. Introduction

GEOMETRY problem solving (GPS) is an interesting but challenging task, which has received increased attention due to its great application potential in intelligent mathematical education [1]–[5]. Given a geometry problem, the machine is expected to provide a solution accompanied by an answer, as shown in Fig. 1. A geometry problem typically consists of an illustrative geometry diagram and a textual problem, where the text problem provides several additional conditions as a supplement for diagram information and explicitly states a problem target. Solving such a geometry problem necessitates the machine to possess the comprehensive capabilities of symbolic abstraction, logical reasoning, and numerical computation. The machine must engage in deliberate multi-step geometry reasoning based on geometry theorem knowledge, thereby providing a solution that arrives at a numerical answer to the problem target. Therefore, GPS has emerged as a pivotal benchmark for evaluating the advanced multimodal reasoning ability of AI machines.

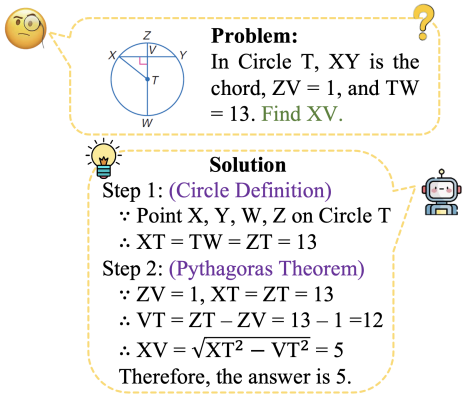

Fig. 1. An example of geometry problem solving. The geometry problem consists of a diagram and a textual problem. The machine is expected to provide a step-wise solution, further deriving the numerical answer of the problem target.

Existing works have explored solving geometry problems along two main lines. One line of neural-based work [6]–[9] formulates GPS as a data-driven text sequence generation task. Based on the multimodal representation of the visual diagram and textual problem, these models are trained in an end-to-end manner to generate a solution program sequence about theorems, arithmetic operators and operands, ultimately obtaining the final answer through program execution. Although these methods can generate intermediate solution programs, they necessitate expensive and labor-intensive manual program annotation for training. It often skips some of the necessary reasoning steps to arrive at the correct answer, resulting in a lack of reliability in terms of mathematical rigor. Moreover, the feature-based diagram representation method cannot exploit the structural and semantic information of geometry diagrams efficiently, leading to inferior performance. The abstract and sparse nature of geometry diagrams makes it challenging to comprehend them accurately with this kind of neural-based

Yaxian Wang is with School of Information Engineering, Chang'an University, Xi'an, Shaanxi 710064, China (email: wyx1566@gmail.com)

Bifan Wei and Jun Liu are with Ministry of Education Key Laboratory of Intelligent Networks and Network Security, and School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China (email: weibifan@xjtu.edu.cn; liukeen@xjtu.edu.cn).

Yinghong Ma and Lingling Zhang are with Shaanxi Province Key Laboratory of Big Data Knowledge Engineering, and School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China (email: myh8888@stu.xjtu.edu.cn; zhanglling@xjtu.edu.cn).

Xudong Jiang is with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore 639798 (email: exdjiang@ntu.edu.sg).

Henghui Ding is with Institute of Big Data, Fudan University, Shanghai 200438, China (email: henghui.ding@gmail.com).

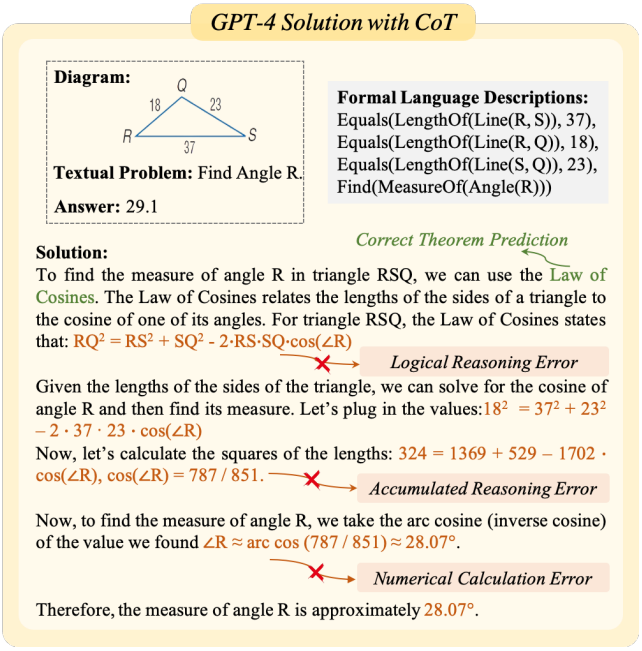

Fig. 2. A case of geometry problem solved by GPT-4 with Chain-of-Thought (CoT). Although the correct geometry theorem is provided, it is prone to different forms of errors in theorem application, such as logical reasoning errors and numerical calculation errors. The incorrect logical reasoning can lead to the accumulated reasoning error further.

methods, even for advanced large language models like GPT-4o [10]. Contrastively, the other line of work [3], [5] parses the geometry diagram and the textual problem into unified formal symbolic language descriptions to convey the information about the geometry problem better. Geometry reasoning is described as two crucial steps, including theorem prediction and theorem application. Based on the unified formal symbolic language descriptions, a theorem sequence is first inferred from a constructed, predefined geometry theorem set. A symbolic solver is then utilized to perform explicit logical and numerical reasoning based on the predicted theorems. The proposed symbolic solver for theorem application has significant advantages in terms of rigor and explainability. Nevertheless, without the prior theorem knowledge, the heuristic sequential theorem prediction strategy is still far away from satisfactory with puzzling redundant steps or costly reinforcement learning required.

Notably, the recent emerging capability of Large Language Models (LLMs) [11]–[14] offers an opportunity to engage in problem-solving with an extensive knowledge space. LLMs trained on a large-scale corpus contain a vast amount of geometry theorem knowledge, so it is promising to take LLMs into account to advance the GPS task. To obtain deeper insights into the LLMs' capabilities to solve challenging geometry problems, we tested CoT-based GPT-4 [15] on the Geometry3k test dataset [3]. Here, we primarily evaluate its geometry reasoning ability. The geometry diagram and the textual problem are translated as the unified formal symbolic language descriptions following [3], [5] to serve as the input of GPT-4, ensuring compatibility with LLMs. CoT-based GPT-4 achieves just 28.6% accuracy, revealing that LLMs still struggle with this complex task. Interestingly, we find that LLMs can perform outstanding theorem prediction due to the rich geometry theorem knowledge possessed by them. Furthermore, we analyze that the reason for the frustrating overall performance on GPS is that LLMs still fall short of rigorous logical reasoning ability about theorem application and accurate numerical calculation ability, which is essential for GPS. Even if a correct theorem can be given, it is prone to making errors when applying it to the corresponding geometric primitives, such as substituting the non-matched sides and angles into the theorem formula shown in Fig. 2, leading to an accumulation of errors in subsequent reasoning.

In summary, exploiting the geometric knowledge within LLMs holds significant potential for advancing theorem prediction in GPS. In contrast, theorem application entails more rigorous logical reasoning over geometry primitives and demands precise arithmetic calculation. Compared with neural-based LLMs, a symbolic reasoning system with mathematical rigor and logical faithfulness has the advantage of reliable logical reasoning and numerical calculation when applying geometry theorems, making symbolic solvers a potential complement to LLMs. Solving a complex geometry problem requires multi-step geometry reasoning about theorem prediction and theorem application. In this reasoning process, humans typically engage in a trial-and-error procedure and perform elaborate strategic exploration through lookahead or backtracking to refine the solution. Therefore, the tree-like structure with the ability to backtrack and iterate is closer to deliberate and divergent problem-solving thinking, making it highly suitable for tasks that necessitate sophisticated reasoning.

In light of the above, we propose a dynamic tree-based geometry problem solver named **GeoTree**, which combines a knowledgeable LLM with a rigorous symbolic solver to perform multi-step geometry reasoning based on a tree-like structure. Our framework involves four main components: *Theorem Seeker*, *Symbolic Solver*, *Evaluator*, and *Controller*, which collaboratively complete one-step geometry reasoning. An iterative process of the four components is performed to construct a tree for step-by-step geometry reasoning, thereby emulating deliberate and divergent human problem-solving thinking. More specifically, *Theorem Seeker* prompts an LLM to seek out candidate theorems with its inherent knowledge. Subsequently, *Symbolic Solver* applies the theorem on the known conditions to obtain intermediate results as new additional conditions. Then, *Evaluator* assesses the availability of the theorem and the usefulness of these conditions for the final problem target, which can serve as the heuristic guidance for subsequent reasoning. Finally, *Controller* plays a role in determining the termination state, which decides whether to continue invoking the other three components. It is worth noting that by decomposing theorem prediction and application into separate components, we take full advantage of the enriched theorem knowledge of LLMs and the logical rigor of the symbolic solver to accomplish this challenging GPS task.

Our contributions can be concluded as follows:

- We propose a novel dynamic tree-based geometry problem solver named GeoTree that emulates human deliber-

ate and divergent thinking to perform multi-step geometry reasoning based on a tree-like structure.
- We leverage a knowledgeable LLM for theorem prediction in conjunction with a rigorous symbolic solver for theorem application to solve geometry problems for the first time. This offers a fresh perspective for the GPS task by combining the complementary strengths of LLMs and symbolic systems.
- Extensive experiments have been conducted on the Geometry3K dataset to demonstrate the superiority of our proposed GeoTree in accuracy, efficiency, and explainability.

The rest of the paper is organized as follows. Section II briefly introduces existing related work. Section III describes the details of GeoTree. Section IV presents the experimental details and analysis of the results for our proposed approach. Section V concludes our work and puts forward future work.

II. Related Work

A. Geometry Problem Solving

Automatic geometry problem solving has attracted much attention in recent years. There has been extensive research focused on diagram parsing and representation [8], [9], [16], problem formalization [17], [18], and geometry reasoning [2], [3], [5]–[7], [19]–[22]. Here, we mainly focus on the research about geometry reasoning. Existing methods of geometry reasoning can be classified into two classes: neural-based and symbolic-based solvers. The neural-based methods [6], [7], [9], [21], [22] adopt a sequence-to-sequence framework to predict a solution program sequence with the multimodal representation as the input. GeoQA [6], GeoQA+ [8] and PGPS9K [16] datasets are proposed with the exhaustive dense annotation of solution programs. With the annotated solution programs at hand, GPS is formulated as a data-driven text sequence generation task based on the multimodal representation of geometry diagram and textural problem. Zhang et al. [23] applied a self-limited decoder to generate solution program sequences in an autoregressive manner, which introduced structural-semantic pre-training to enhance geometry problem representation and a multi-level theorem verifier to select the solution programs with the highest confidence as the final solution. Zhang et al. [24] converted geometry diagrams into natural language descriptions and then directly integrated LLMs to generate solution programs. Nevertheless, it is essentially a neural-based solver that generates a sequence of solution programs. Contrastively, the symbolic-based methods have made efforts to incorporate theorem knowledge as the form of horn clause rules [2], [19] or theorem conditional rules [3], [5]. A prolog-style probabilistic solver [2] is proposed to perform logical reasoning based on horn-clause rules, but the parameter optimization of the solver makes theorem search difficult. Later, Lu et al. [3] and Peng et al. [5] proposed to perform explicit symbolic reasoning via theorem sequence search and condition matching based on the theorem conditional rules to derive the final answer. Inter-GPS [3] parses the diagram and text problem into formal language descriptions first and then performs explicit symbolic reasoning, including path search and condition matching. GeoDRL [5] enhances the search strategy of Inter-GPS by incorporating logical graph deduction and deep reinforcement learning. HGR [25] parses the problem text and diagrams into a unified global hologram, thereby achieving direct reasoning on the hologram. It uses deep reinforcement learning to perform pattern matching between the global hologram and a set of predefined graph models, each representing a geometric theorem, further facilitating theorem selection and application. Latest AlphaGeometry [26] mainly adds new auxiliary construction to open new paths for the symbolic deduction until a solution is found, which aims to find proofs for complex geometry theorems like the Olympiad problems.

In this work, we focus on geometry problem solving to derive a numerical value of the problem target by finding a reasonable theorem sequence and meanwhile applying it correctly. Different from these existing methods, our method tries to incorporate theorem knowledge to promote theorem prediction. We take full advantage of an LLM with rich geometry theorem knowledge to focus solely on theorem prediction, at which it excels, while delegating the task of theorem application to a rigorous symbolic solver. The individual advantages of neural LLMs and symbolic systems are fully exploited to accomplish the theorem prediction and application, respectively. This division of labor allows each component to play to its strengths.

B. Math Problem Solving

Recent studies have explored various prompting methods based on LLMs to solve math word problems [27], [28], [28]–[31]. Addressing a complex problem often involves breaking down the overall target into multiple intermediate steps, where the solution associated with each step is a thought. There are different forms to organize the thoughts, including linear chains (CoT) [15], hierarchical trees (ToT) [32], and intricate graphs (GoT) [33]. However, LLMs still struggle to perform arithmetic operations [34], [35]. Existing methods [36]–[39] have introduced an external calculator to perform the operations. PAL [37] prompts LLMs to express the reasoning steps as Python programs and utilizes a Python interpreter to perform the calculations. Program-of-thoughts (PoT) [38] generates accurate programs to express complex reasoning procedures, which delegates the computation steps to an external program interpreter. Joy He-Yueya et al. [39] used an LLM to formalize the problem as a set of variables and equations, and then employed SymPy [40], a Python library for symbolic computation. Later, Zhao et al. [41] combined CoT and PAL by utilizing an LLM to make a selection dynamically. Furthermore, Liu et al. [42] proposed XoT as an integrated problem-solving framework to prompt LLMs with diverse reasoning thoughts, including CoT, PoT, and Equation-of-Thought (EoT), which allows the model to rethink and switch to a different method via verification.

Unlike math problem solving, geometry problem solving focuses more on the complex theorem application, which involves not only arithmetic computation but also logical reasoning over geometry primitives. Although ToT [32] has

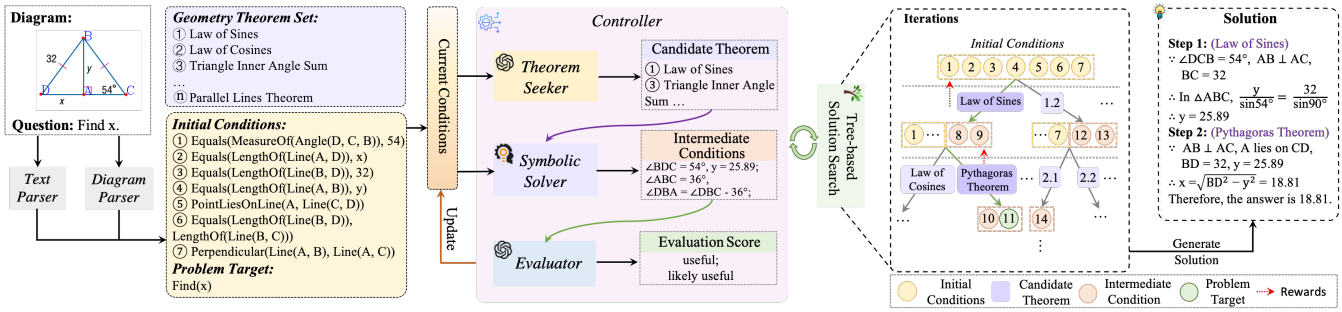

Fig. 3. The overall framework of our proposed GeoTree. **Left:** The diagram and the question are parsed with a diagram parser and a text parser to obtain the initial conditions as the initial reasoning state. Four components are designed to complete one-step reasoning, including *Theorem Seeker*, *Symbolic Solver*, *Evaluator*, and *Controller*. **Right:** An iterative process of these four components is performed for step-by-step geometry reasoning, enabling the tree-based solution search. The overall reasoning process is accompanied by a geometry tree construction process. The bold green path in the tree indicates the final reasoning path to the problem target, which can be processed further to generate a readable solution.

demonstrated that tree-based reasoning possesses the ability to iterate and backtrack, scientific geometry reasoning is not discussed. It relies solely on the knowledge of LLMs and lacks the interaction between the intermediate steps in the tree with the real-time environmental feedback. In this work, our proposed GeoTree incorporates four components to perform dynamic and interactive geometry reasoning with their coordinated efforts.

III. Methods

In this section, we first formalize the task of Geometry Problem Solving (GPS) in Section III-A. Then, we describe our proposed GeoTree in Section III-B. Next, we introduce the four main components, including *Theorem Seeker*, *Symbolic Solver*, *Evaluator*, and *Controller*, respectively. Finally, in Section III-G, the tree-based solution search algorithm is elaborated in detail.

A. Problem Definition

A geometry problem P consists of an illustrative geometry diagram d, a text problem p, where the text problem p includes several textual additional conditions C as the complement to the diagram information, and a problem target g as the reasoning goal. The objective is to find an eligible theorem sequence $T = [t_1, t_1, ..., t_k]$ from a geometry theorem set $\mathcal{KB}$, and apply the theorem sequence reasonably to derive a numerical value a of the problem target g. In essence, geometry reasoning can be described as two crucial steps, including theorem prediction and theorem application, wherein theorem prediction can be described as an exploring process to determine which theorems and in what order to apply them to solve a geometry problem.

B. Overview of GeoTree

When solving a complex geometry problem, humans typically engage in an elaborate trial-and-error process and iterative exploration to refine a solution by utilizing the geometry knowledge inherent in the brain. Taking inspiration from deliberate and divergent human problem-solving thinking, we incorporate a tree structure into the geometry reasoning process. The overview of our GeoTree framework is presented in Fig. 3. Specifically, the geometry diagram and the textual problem can be parsed into a unified formal symbolic language with a diagram parser and a text parser following [3], [5]. The unified formal language descriptions act as the initial conditions for geometry reasoning. In each reasoning step, four main components are incorporated to accomplish a sub-task about a theorem collaboratively: *Theorem Seeker* for theorem prediction, *Symbolic Solver* for theorem application, *Evaluator* for theorem evaluation, and *Controller* for termination decision. An iterative process of the abovementioned four components is performed based on a tree to complete step-by-step geometry reasoning, enabling the tree-based solution search. When the problem target is hit, we can extract the reasoning path to the problem target from the tree, which can be post-processed to generate a readable solution.

Particularly, as shown in the right part of Fig. 3, in a geometry tree, the root node with a yellow dotted box denotes the initial conditions derived from the parsed question text and the diagram, and the non-root nodes denote the current conditions. Each edge denotes the state transition from current conditions to new conditions after applying the theorem. For clarity, each candidate theorem from *Theorem Seeker* is denoted as a square purple node and inserted into the edge. Based on the current given conditions, a theorem can be applied via *Symbolic Solver* to introduce new additional conditions, combining with existing conditions as the child node of the current conditions node. Then, *Evaluator* is designed to assess the contribution of the newly generated conditions, which serves as the heuristic guidance for the subsequent reasoning. Finally, *Controller* plays a role in determining the termination state, which decides whether to continue invoking the other three components for next-step reasoning. The iterative geometry reasoning process is accompanied by a geometry tree construction process driven by a tree search algorithm.

Next, we first introduce the four components in detail and then describe the tree-based solution search algorithm.

C. Theorem Seeker

The theorem-driven nature of geometry problems enables us to address this complex GPS task by breaking it down

Theorem Seeker Prompt

(Instruction) Below, you are an expert in solving plane geometry problems. Please follow the four examples given to you as a paradigm, and based on the known conditions and the problem target, provide the most likely theorems to be chosen for the next step to solve the problem. The selected theorems must strictly come from the following 17 theorems, and please select {n_propose} theorems alternatives. You had better not choose used theorems. (note: you only need to provide the sequence numbers of theorems you think are most likely to use next, separated by spaces, without additional explanation):

1. Triangle inner angle sum theorem (1-2 unknown angles out of three angles).
2. Triangle inner angles sum theorem (all three angles are unknown).
3. Isosceles triangle theorem (angle).
4. Isosceles triangle theorem (side).
…
14. Similar triangle proving and applying.
15. Angular bisector theorem for triangles.
16. Law of Cosines.
17. Law of Sines.

[In-context examples]

Your task:
Current Conditions: {conditions}
Used Theorems: {used theorems}
Problem Target: {target}
Candidate Theorems:

Fig. 4. The prompt for *Theorem Seeker* including instruction, in-context examples and a test case.

into a series of manageable atomic steps about theorems. Following [3], [5], we have a predefined theorem set $\mathcal{KB}$ about different graphics such as triangles, circles, and polygons to be our search space. *Theorem Seeker* plays an important role in generating possible candidate theorems at each reasoning step, which prompts an LLM to exploit its prior theorem knowledge and unleash its potential for seeking out relevant theorems. A prompt template is designed for the *Theorem Seeker*, which consists of three parts: instruction, in-context examples, and a test problem including current conditions, used theorems followed by the final problem target. As shown in Fig. 4, the prompt is formulated as follows: "Instruction: [I]. In-context Examples: [E]. Current Conditions: [C]. Used Theorems: [U]. Problem Target: [T]. Candidate Theorems:". Specifically, the instruction provides the LLM with the background of the task and an available predefined theorem set (e.g., "*Triangle Angle-Sum Theorem*", "*Parallel Lines Theorem*"). The few-shot in-context examples serve as a demonstration to instruct the LLM to propose candidate theorems for a geometry problem in a similar form. The current conditions provide the LLM with the initial reasoning state (note that the conditions are updated after each reasoning step). The used theorems in the previous reasoning step are updated to serve as an implicit hint for the subsequent theorem prediction. The problem target acts as the final reasoning goal. The output is a sequence of promising candidate theorems that may contribute to the problem target,

Evaluator Prompt

(Instruction) As a geometry expert, based on the known conditions and the problem target, please determine how helpful the newly added intermediate conditions are in solving the problem target, considering the following three factors including direct relevance with the final target, necessity, sufficiency. The degree of help is formulated as a classification task including useless, likely useful, and useful, denoted by the numbers [1,2,3], where 3 represents great help, 2 represents a little help, and 1 represents almost no help.

[In-context examples]

Your Task:
Current Conditions: {conditions}
Problem Target: {target}
New Conditions: {new conditions}
Usefulness:

Fig. 5. The prompt for *Evaluator* including instruction, in-context examples, and a test case.

i.e., $\mathcal{T} = \{t_{d,j} | j = 1, ..., n\}$, where $t_{d,j}$ denotes the j-th theorem at d-th reasoning step.

D. Symbolic Solver

Instead of relying on LLMs to give the solution about the theorem application, we adopt a rigorous symbolic solver to perform logical and numerical reasoning based on the given conditions and the given promising theorems output by *Theorem Seeker*. By decomposing theorem prediction and theorem application into two separate components, we take full advantage of the abundant theorem knowledge of LLMs and the logical rigor of the symbolic solver for geometry reasoning. Specifically, each theorem t can be defined as a condition statement with a premise P and a conclusion O. The premise P consists of a set of known conditions, which encompass the semantic and structural information about geometry primitives, such as quantitative information and geometry relationships. A successful application of a theorem can lead to the conclusion O. Here, the application for each theorem is implemented by a rigorous symbolic module, which leads to a state transition, such as the application of "*Triangle Inner Angle Sum Theorem*" ($\triangle$ABC, $\angle$BAC $= 90°$, $\angle$BCA $= 54° \Rightarrow \angle$ABC $= 36°$). Specifically, if the current conditions C (the conditions given by the original problem and the intermediate conditions obtained from applied theorems in previous steps) can match the premise P of a theorem, the theorem is applicable. Then, the conclusions are generated to act as the new intermediate conditions C', facilitating the subsequent reasoning. Theorems that pertain to triangles are not suitable for problems involving circles. Therefore, it is necessary for *Theorem Seeker* to combine the conditions at hand and geometry theorem knowledge to ensure the applicability of given theorems as much as possible.

E. Evaluator

Evaluating the generated theorems is helpful to assist the model in selecting the most promising nodes to continue

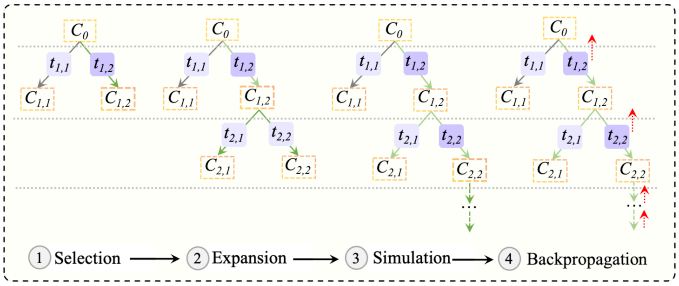

Fig. 6. The illustration of the four steps for an iteration in the Monte Carlo Tree Search.

reasoning. The accuracy of individual steps in a sequence of reasoning steps cannot guarantee the overall accuracy of the solution. The reason is that the correctness of the final solution for a geometry problem often relies on the accuracy and validity of the entire sequence of steps. There is also no ground-truth intermediate solution to be provided. Therefore, it is challenging to evaluate the accuracy of each step. Here, we evaluate from two aspects: theorem applicability and theorem usefulness. Firstly, the candidate theorems are retained if they are available. Otherwise, they are deleted, thus narrowing the search space for further exploration. If all candidate theorems are not applicable, the *Theorem Seeker* is prompted to re-generate the candidate theorems. Secondly, the usefulness of the theorem is equivalent to the usefulness of the intermediate conditions. By evaluating the intermediate results after applying theorems, we can judge their contribution to solving the final problem target. We use an LLM as the *Evaluator* to generate a score r as an assessment for the current reasoning state s, which is represented as $R(f_\theta, s) \sim f_\theta(r|s)$. Specifically, we employ a prompt-based approach and consider it as a classification problem. As shown in Fig. 5, the prompt is formulated as follows: "Instruction: [I]. In-context Examples: [E]. Current Conditions: [C]. Problem Target: [T]. New Conditions: [C_n]. Usefulness:". The LLM is prompted to output one of three classes including "useful", "likely useful", or "useless". Then, the classification can be heuristically transformed into a numerical value r, which acts as a reward to motivate the reasoning to advance toward the problem target.

F. Controller

The controller is designed to determine whether the termination state has been hit and whether to continue to invoke the other three components. Since GeoTree is a recurrent algorithm, each iteration is terminated when the final problem target is solved, or the maximum depth is reached. Given the initial state, the symbolic solver is first called to check if it can succeed in reaching the problem target directly. If it is not successful, the controller recurrently calls the above three components to perform geometry reasoning until it can reach a termination state.

G. Tree-based Solution Search

To determine the solution search logic, our GeoTree is equipped with the Monte Carlo Tree Search (MCTS) [43]–[47] algorithm. MCTS serves as an efficient search method to optimize the prediction of a theorem sequence from a given theorem space. Compared with brute-force tree search methods like breadth-first search (BFS) and depth-first search (DFS), MCTS is superior due to the balance between exploration and exploitation. It is also flexible to incorporate other tree search algorithms into our GeoTree. Specifically, the geometry tree is built by performing four steps iteratively, namely selection, expansion, simulation, and backpropagation, as shown in Fig. 6. Algorithm 1 illustrates the overall iterative geometry reasoning process based on MCTS algorithm in detail.

Algorithm 1 Tree-based Solution Search

Require: Initial Conditions C_0, *Theorem Seeker*, *Symbolic Solver*, *Evaluator* and *Controller*, number of generated candidates theorems n, number of iteration K, depth limit D. Initialize theorem set $\mathcal{KB}$: $t \mapsto \mathcal{KB}$. Define stored values $V(C,t)$, the selection policy $\mathcal{P}()$, and visit counter $N()$.
$root(T) \leftarrow C_0$
for $i = 0$ to $K-1$ **do**
  for $d = 0$ to $D-1$ **do**
    // Expansion & Simulation
    $\{t_1, ..., t_n\} \in \mathcal{KB} \leftarrow$ *Theorem Seeker*$(C_d, \mathcal{KB})$
    for $j \leftarrow 1, 2, ..., n$ **do**
      // Apply theorem
      $C'_{d+1,j} \leftarrow$ *Symbolic Solver*$(C_{d,j}, t_j)$
      // Evaluate conditions-theorem pair
      flag, $r_{d,j} \leftarrow$ *Evaluator*$(C'_{d+1,j}, t_j)$
      if flag **then** Update conditions with $C'_{d+1,j}$ to get $C_{d+1,j}$ and add $C_{d+1,j}$ to children
    end for
    // Selection
    $t_d \leftarrow \mathcal{P}(C_d, t, V_d, N)$
    $C_{d+1} \leftarrow$ *Symbolic Solver*(C_d, t_d)
    $r_d \leftarrow$ *Evaluator*(C_{d+1}, t_d)
    Increment $N(C_{d+1}, t_d)$
    Termination State $S \leftarrow$ *Controller*(C_{d+1})
    if S **then** break
  end for
  $M \leftarrow$ the actual number of steps
  // Backpropagation
  for $d \leftarrow M-1, ..., 0$ **do**
    Update $V(C_d, t_d)$ with $\{r_d, r_{d+1}, ..., r_l\}$
  end for
end for

First, each iteration begins by selecting an expandable node to plan a promising solution. Starting from the root node, the selection step ends with a leaf node. Note that, in the first iteration, only the root node with the initial conditions can be selected. Following the Upper Confidence bounds applied to Trees (UCT) [44] algorithm, the node selection policy $\mathcal{P}$ is defined as:

$$t^* = \underset{t \in \mathcal{T}(C)}{\arg\max} \left(\frac{V(C,t)}{N(C,t)} + w \sqrt{\frac{lnN(C)}{N(C,t)}} \right), \quad (1)$$

where $V(C,t)$ denotes the value of the conditions-theorem

pair, $N(C,t)$ denotes the number of applying theorem t to conditions node C, $N(C)$ denotes the total visited count of the node C in previous iterations, and w is the exploration weight for the exploration and exploitation trade-off.

Subsequently, the expansion step adds new child nodes to the leaf node selected above, further expanding the tree. Next, in the simulation step, the node with the largest local reward r is expanded until the problem target is solved or the predefined limit is reached. The simulation step consists of multiple expansion steps. In each expansion step, the above-mentioned four components for geometry reasoning are performed. When the termination state is hit, a top-down reasoning path can be extracted from the root node to the leaf node. The rewards of nodes output by *Evaluator* are back-propagated from bottom to top to update the value $V(C,t)$ of each conditions-theorem pair along the path. By updating the values in this manner, the algorithm can make more informed decisions during the selection step of the search algorithm.

After completing the predefined K iterations, the search is terminated. The multiple iterations allow comprehensive exploration to refine the solution, which may produce multiple solution alternatives that can lead to the answer. Then, the optimal reasoning path with the highest reward and the predicted answer of the problem target can be extracted from the constructed tree. To provide a more readable solution for humans, we post-processed the reasoning process according to the selected theorem, and intermediate conditions contributed to the final answer.

IV. Experiments

In this section, we first introduce the two datasets and the evaluation metric in Section IV-A. Then, we introduce the implementation details in Section IV-B. In Section IV-C, we present the experimental results. Subsequently, we conduct extensive experiments to perform further analysis for GeoTree in Section IV-D. Finally, we make analyses for some cases in Section IV-E.

A. Dataset and Evaluation

Dataset. We conduct experiments on a public geometry problem dataset Geometry3K [3], which is collected from the geometry textbooks across 6-12 grades. We evaluate the performance of all models on the test set containing 601 problems. The geometry diagram and the problem text of the geometry problem are annotated with dense formal language descriptions, which bridge the multimodal semantic gap and facilitate geometry reasoning. These problems can be classified into two classes according to geometric shapes (e.g., line, triangle, quadrilateral, and circle) and goal types (e.g., angle, length, area, and ratio).

Evaluation Metric. We use accuracy as the evaluation metric. However, the previous methods in the form of multiple choices [3], [5] select the choice closest to the generated answer from four candidates. In this way, if a generated answer is incorrect but happens to be closest to the correct choice, the correct option is also selected, or the randomly guessed answer happens to be the correct choice, causing the evaluation results to be falsely high. To provide more robust evaluation results for the model performance, we define that the problem is solved correctly only when the difference between the generated answer and the ground truth is within 0.1. Moreover, if the solver fails to output a numerical value of the problem target, we regard it as a wrong prediction with "no result" rather than randomly choosing one from the four candidates. Additionally, the new setting demands generating open-ended answers based on the problem without accompanying choices, posing a greater challenge for the pure LLM-based methods. We also compare the model performance using the two evaluation settings in subsequent experiments.

B. Imeplementation Details

We use the OpenAI GPT-3.5 (gpt-3.5-turbo) model as an LLM backbone for all evaluated methods. Our experiments are conducted in the in-context learning setting. We manually designed 4-shot exemplars for *Theorem Seeker* and 2-shot exemplars for *Evaluator*. These demonstration examples are selected from the training set of Geometry3K. The score value is sampled 3 times for *Evaluator* in each step and the average value is taken as the final score. During theorem prediction, we set the temperature of the LLM to 0.7. The number of candidate theorems in each step n is limited to 7. Our symbolic solver is built on GeoDRL [5], and improves its equation solver. The number of iterations is defined as 5. We conduct experiments on a geometry theorem set $\mathcal{KB}_1$ with 17 theorems [3] and an expanded theorem set $\mathcal{KB}_2$ with 24 theorems [5]. To test the performance upper bound that GeoTree can reach without interference from other noise information, we use the ground truth formal language descriptions for the diagram and the textual problem, which can be regarded as an independent evaluation for the geometry reasoning. For a fair comparison, the same goes for all other baselines. Additionally, we also conduct the experiments using the parsed formal language descriptions from parsers.

C. Experimental Results

Baselines. We compare our method with two classes of baselines: two symbolic methods requiring training including Inter-GPS [3] and GeoDRL [5] and four LLM-based methods including IO prompting, chain-of-thought (CoT) [15], self-consistency with CoT (CoT-SC) [48], and tree-of-thought (ToT) [32]. The details are introduced as follows:

- Inter-GPS [3] utilizes a transformer-based theorem predictor with a low-first search strategy to solve the geometry problem.
- GeoDRL [5] enhances the search strategy of Inter-GPS by introducing logical graph deduction and reinforcement learning.
- IO prompting method receives the problem and directly outputs an answer via an LLM.
- Chain-of-thought (CoT) [15] promotes LLMs to generate the intermediate process step-by-step to arrive at the final answer. We use GPT-3.5 and GPT-4 with CoT as our baselines, respectively.

TABLE I
COMPARISON OF MODEL PERFORMANCE ON THE TEST SET OF GEOMETRY3K. * DENOTES OUR REPRODUCED RESULTS IN THE NEW EVALUATION SETTING WITH THE OPEN-SOURCE CODE. $^\diamond$ DENOTES OUR MODIFIED BASELINES. † DENOTES USING THE THEOREM SET $\mathcal{KB}_1$, AND ‡ DENOTES USING THE THEOREM SET $\mathcal{KB}_2$.

Method	All	Angle	Length	Area	Ratio	Line	Triangle	Quad	Circle	Other
Human	56.9	53.7	59.3	57.7	42.9	46.7	53.8	68.7	61.7	58.3
Human Expert	90.9	89.9	92.0	93.9	66.7	95.9	92.2	90.5	89.9	92.3
Inter-GPS*† [3]	69.7	77.6	68.2	47.2	37.5	**81.5**	79.4	72.3	47.4	47.8
GeoDRL*‡ [5]	75.9	**81.0**	73.9	66.0	62.5	71.6	78.1	81.5	**68.2**	82.6
IO	12.6	8.4	16.8	9.4	0.0	9.8	12.4	19.2	8.1	13.0
CoT [15]	15.9	16.5	15.5	17.0	12.5	17.3	15.0	21.5	11.8	13.0
CoT-SC [48]	18.3	16.9	18.2	22.6	25.0	17.3	18.5	26.9	11.1	17.4
CoT (GPT-4) [15]	28.6	30.4	27.1	24.5	50.0	28.4	31.3	34.6	20.0	21.7
o1-mini [49]	60.7	48.5	71.5	54.7	62.5	45.6	72.1	63.9	46.7	60.9
ToT (SQ)‡ [32]	38.6	56.1	22.5	45.3	75.0	51.9	25.8	58.5	35.5	30.4
ToT$^\diamond$† [32]	59.7	69.2	57.9	30.2	37.5	66.7	66.5	67.7	40.0	39.1
ToT$^\diamond$‡ [32]	67.4	68.8	67.6	60.4	62.5	67.9	77.3	80.8	38.8	60.9
GeoTree†	67.6	71.7	67.6	52.8	37.5	69.1	78.5	73.1	46.0	47.8
GeoTree ‡	**76.5**	75.9	**78.8**	**66.0**	**75.0**	75.3	**82.4**	**81.5**	61.4	**82.6**

- Self-consistency with CoT (CoT-SC) [48] improves CoT by sampling k different chains of thought and employing majority voting to output the frequent answer.
- GPT-o1 [49] is a next-generation language model focused on complex reasoning, with two versions: o1-preview and o1-mini. The o1-preview is ideal for tasks requiring broad general knowledge, while the o1-mini is smaller, faster, and cost-effective, suitable for coding, math, and science tasks. Despite its smaller size, o1-mini retains much of o1-preview's reasoning power, making it a practical choice for cost-effective and faster processing. In summary, we select o1-mini as the baseline.
- Tree-of-thought (ToT) [32] extends CoT to explore reasoning paths via searching based on a tree, which is modified to adapt to the GPS task. ToT (SQ) decomposes the whole geometry problem into sub-problems as the thoughts that contribute to the problem target, similar to what is done in math problem solving.
- ToT$^\diamond$ [32] generates the theorems and the results of the theorem application as the thought. The baseline is implemented based on the breadth-first-search (BFS) algorithm. The same symbolic solver as ours is utilized to perform logical reasoning and numerical computation for a fair comparison.

These LLM-based baselines use 3-shot exemplars, and k is set to 5 for CoT-SC. Additionally, the performance of human and human experts is also included. Human denotes that the annotators have obtained a high school or higher degree. Human expert denotes graduates majoring in science or engineering. They are asked to answer all problems in the test set to obtain the performance.

Main Results. The experimental results are presented in Table I. We make the following three observations:

- First, compared with the IO prompting method, the utilization of improved prompting strategies achieves advantages over vanilla LLMs in the GPS task. Among these methods relying solely on LLMs, CoT-based GPT-4 achieves a performance of only 28.6%. We analyze that the vanilla LLMs often rely on surface-level language pattern matching and still struggle to perform deep reasoning and rigorous logical derivation. Compared to CoT (GPT-4), the latest reasoning LLM o1-mini achieves a substantial performance improvement of 32.1%, which highlights its strong reasoning capabilities. However, despite these improvements, o1-mini remains substantially inferior to our proposed GeoTree, exhibiting a performance gap of 15.8%. We believe that the root cause of this gap lies in the complex nature of the geometry problem itself. It not only requires a deep understanding of geometric primitives and their relationships, but also relies on the precise prediction and application of multiple theorems to perform rigorous logical reasoning and numerical computation. Although large language models like GPT-o1 have made significant progress in the GPS task, relying solely on them for rigorous geometric logic reasoning still has obvious limitations, resulting in suboptimal performance. Our method combines LLMs with symbolic systems to balance language understanding, numerical computation, and geometric logic reasoning, thereby achieving more powerful automatic solving capabilities for the GPS task.
- Second, ToT (SQ) based on the sub-problems performs worse than ToT based on the theorems, which proves that it is difficult to decompose a geometry problem due to

TABLE II
THE IMPACT OF THE SIZE OF THE THEOREM SET ON THE GPS PERFORMANCE OF COT PROMPTING-BASED GPT-3.5.

Setting	Accuracy (%)
with $\mathcal{KB}_1$	10.0
with $\mathcal{KB}_2$	12.5
No Limitation	15.9

TABLE III
PERFORMANCE COMPARISON OF GPS USING DIFFERENT PARSING RESULTS OF THE DIAGRAM AND TEXTUAL PROBLEM.

Models	GT (%)	Predicted (%)
Inter-GPS [3]	69.7	$43.3_{(-26.4)}$
GeoDRL [5]	75.9	$50.4_{(-25.5)}$
GeoTree	76.5	$42.3_{(-34.2)}$

TABLE IV
PERFORMANCE COMPARISON UNDER DIFFERENT EVALUATION SETTINGS. FIB DENOTES FILL-IN-THE-BLANK, AND MC DENOTES MULTIPLE-CHOICE.

Models	FIB (%)	MC (%)
Inter-GPS [3]	69.7	78.3
GeoDRL [5]	75.9	84.9
CoT (GPT-3.5) [15]	15.9	29.0
GeoTree	76.5	83.5

the weak connection between sub-problems and the main problem. Compared with the modified ToT, our GeoTree achieves a performance gain using both theorem sets, which demonstrates the superiority of MCTS combined with our framework.

- Third, compared with Inter-GPS using $\mathcal{KB}_1$ and GeoDRL using $\mathcal{KB}_2$, our few-shot GeoTree achieves comparable performance with the same theorem set. However, they require re-training when introducing new theorems, especially GeoDRL, which uses reinforcement learning and involves time-consuming training to adapt to a new theorem set. Our GeoTree using LLMs can integrate new theorems effortlessly and search for optimal theorem sequences effectively for a new specific geometry problem, which provides flexibility in problem solving.

D. Ablation Study and Further Analysis

Effect of Theorem Set Size. To explore the impact of the size of the theorem set, we conduct experiments with different sizes of geometry theorem sets. Experimental results in Table I demonstrate that introducing a more complete theorem set $\mathcal{KB}_2$ compared to $\mathcal{KB}_1$ can improve model performance effectively. Note that the symbolic solver is designed only for each theorem in the theorem set with a fixed size. Here, we take CoT-based GPT 3.5 as an example to further analyze the impact of the theorem set. As shown in Table II, when there is no limited theorem set given, CoT-based GPT 3.5 achieves better performance with its inherent knowledge. We analyze that LLM itself encompasses comprehensive theorems, which can cover more geometry problems. Therefore, how to develop the symbolic solver to accommodate more theorems automatically is a potential research direction.

Effect of the Parsing Results. In the above main experiments, we utilize the ground truth parsing results of the textual problem and the diagram, which can be seen as an independent performance evaluation of geometry reasoning. To further explore the effect of the parsing results on the textual problem and the diagram, we use textual and diagram parsers [3] to get the formal symbolic language descriptions as the input of the subsequent geometry reasoning. The experimental results in Table III demonstrate that the inaccurate parsing results for the diagram and textual problem affect the subsequent reasoning, resulting in cascade errors. Moreover, we can find that the negative impact of inaccurate parsing results is amplified in our few-shot model. Without being trained on a large amount of specific annotated data, the quality of parsing results becomes more crucial for the overall performance and reliability of the model. Therefore, designing improved parser techniques can be valuable to deploy our framework in user-facing applications, further better dealing with any geometry problem from a user.

TABLE V
COMPARISON OF EFFICIENCY FOR DIFFERENT MODELS.

Method	Accuracy (%) ↑	No Result (%) ↓	Steps (Avg)↓
Inter-GPS [3]	69.7	25.6	7.03
GeoDRL [5]	75.9	16.6	2.31
GeoTree	76.5	15.5	2.18

TABLE VI
COMPARISON OF EXPLAINABILITY FOR DIFFERENT MODELS.

Method	Explainability (%)↑
Inter-GPS [3]	49.3
GeoDRL [5]	88.9
GeoTree	87.0

Effect of the Evaluation Methods. When geometry problem solving (GPS) is formulated as a multiple-choice task with four candidate answers, one choice must be given as the final answer. However, unlike traditional visual question answering as a classification task, GPS is a natural regression task to obtain a numerical answer about the problem target by finding a solution. If a numerical value of the problem target is generated, the choice closest to the value is selected. If the problem target is not solved, one choice from the four candidates is randomly chosen. Although the models generated wrong solutions in this kind of multiple-choice setting, the correct choice may be selected. Therefore, we tend to evaluate GPS by formulating it as a fill-in-the-blank task. In this setting, the ground-truth answer is defined as a numerical value. The experimental results in Table IV demonstrate that the multiple-choice setting leads to higher evaluation results compared with the fill-in-the-blank setting using the same model. There are two types of problems in practical math exams, and we do not claim which is better. We use the fill-in-the-blank setting in all our main experiments to evaluate the real performance of the models on GPS.

Analysis of Model Efficiency. To further validate the effectiveness of GeoTree, we compare the efficiency of Inter-GPS, GeoDRL, and GeoTree. On the one hand, compared with the two training-required methods, our few-shot GeoTree can solve a geometry problem more efficiently and flexibly. On the

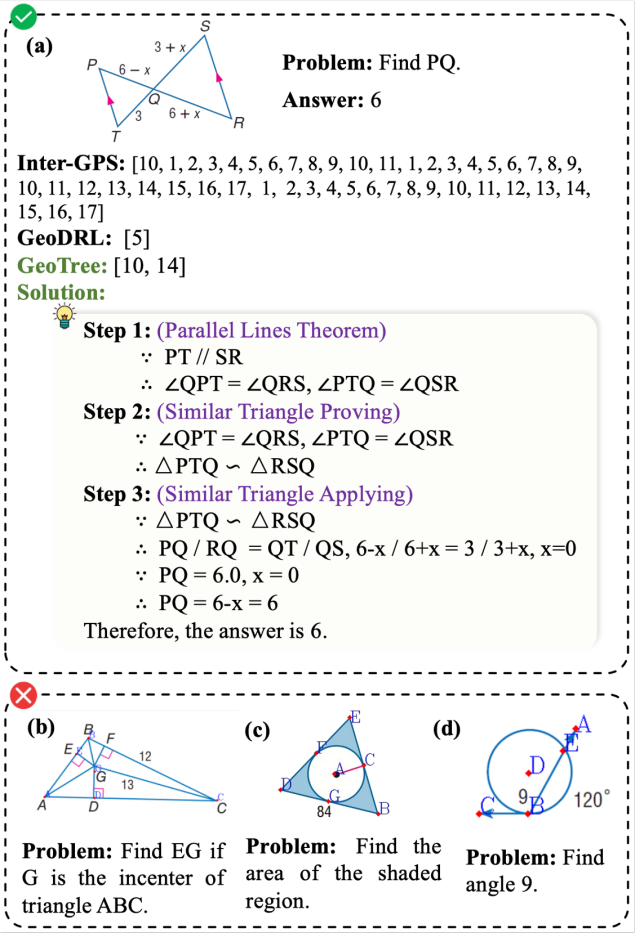

Fig. 7. Several typical successful and failure cases of GeoTree.

other hand, when solving a geometry problem, it is crucial to strive for simplicity to ensure the clarity of the solution. As demonstrated in Table V, compared with Inter-GPS and GeoDRL, our GeoTree archives a simpler solution with fewer steps. Meanwhile, GeoTree can obtain higher accuracy and a lower proportion of "no result".

Analysis of Solution Explainability. To obtain deeper insights into how effectively GeoTree provides explainable solutions for humans, we sample 100 problems from the test set and manually evaluate the explainability of correct cases for each model. The metric is computed as *S1 / S2 = the number of explainable cases / the number of correct cases*. We argue that an incorrect case is certainly unexplainable. All three methods perform logical reasoning based on an explainable symbolic solver to derive the problem target. Therefore, we primarily assess explainability in terms of solution step relevance, which involves evaluating the ability of the model to predict theorems accurately. An irrelevant step in the solution also makes the model unexplainable. We argue that a good theorem sequence should be reasonable and succinct. As shown in Table VI, Inter-GPS exhibits poor explainability due to redundant steps introduced by blindly multiple attempts, which results in a puzzling process for humans. Furthermore, our GeoTree achieves comparable explainability to GeoDRL without costly training, demonstrating the potential of GeoTree.

E. Case Study

We conduct several case studies to discuss the strengths and weaknesses of GeoTree, as shown in Fig. 7. In case(a), Inter-GPS and GeoDRL all fail to solve this problem, while GeoTree can generate a reasonable and explainable theorem sequence, which can be applied to arrive at the correct answer with our improved equation solver. We also provide an interpretable and human-readable problem-solving process. Cases (b), (c), and (d) are three failure cases. Specifically, case (b) with a sophisticated diagram layout involves rich geometric primitives and their complex geometric relationships, making primitive-level reasoning insufficient and prone to getting lost in the low-level details. Integrating primitives with higher-level geometric shapes (e.g., triangles, quadrilaterals) facilitates both detailed and holistic reasoning. A promising direction for future research is to enhance parsers for automatic shape detection and develop efficient symbolic solvers across both primitive and shape levels. Case (c) is a kind of shadow area problem. The main reason for the error in this type of problem is the unclear problem target, making it difficult for the parser to determine the specific region intended for calculation. Clearly defining this kind of region by converting it into calculations involving regular shapes (e.g., $S_{\triangle EDB} - S_{\odot A}$) or introducing the intuitive visual diagram as the input can help reduce ambiguity and improve overall accuracy. Furthermore, due to the limitations of the predefined geometry theorem set, case (d) cannot utilize the necessary "*Radius-Tangent theorem*" to get "$\angle \mathrm{CBD} = 90°$" to solve the problem. The analysis of failure cases reveals the limitations of GeoTree while also shedding light on potential directions for our future research.

V. Conclusion

In this paper, we propose a novel dynamic tree-based geometry problem solver named GeoTree, which integrates a knowledgeable LLM and a rigorous symbolic solver for theorem-driven geometry problem solving. Benefiting from the abundant geometry theorem knowledge of LLMs, GeoTree can perform effective theorem prediction with the guidance of prior knowledge. For theorem application, a symbolic solver with mathematical rigor is utilized to perform logical reasoning and numerical computation. Inspired by human-like deliberate and divergent thinking, GeoTree performs an iterative tree construction process dynamically via MCTS, which simulates a trial-and-error process to complete step-by-step geometry reasoning until the termination state is reached. Moreover, four components consisting of *Theorem Seeker*, *Symbolic Solver*, *Evaluator*, and *Controller* are designed to finish one-step reasoning collaboratively. Extensive experiments demonstrate the superiority of GeoTree in accuracy, efficiency, and explainability. In the future, we are going to explore the following directions: (1) We are going to improve the symbolic solver to accommodate more theorems automatically, further promoting the application of GeoTree in real-world scenarios. (2) The theorem seeker and evaluator rely on the powerful capabilities of frozen LLMs that are pretrained on general domains and only accessible with APIs. It is also promising to explore how to fine-tune LLMs tailored to the specific

scientific geometry domain, enabling GeoTree to be applied to geometry problem-solving more flexibly and effectively.

ACKNOWLEDGEMENT

This work was supported by National Key Research and Development Program of China (2022YFC3303600), National Natural Science Foundation of China (62137002, 62293550, 62293553, 62293554, 62450005, 62437002, 62477036, 62477037, 62176209, 62192781, and 62306229), "LENOVO-XJTU" Intelligent Industry Joint Laboratory Project, the Shaanxi Provincial Social Science Foundation Project (2024P041), the Natural Science Basic Research Program of Shaanxi (2023-JC-YB-593), and the Youth Innovation Team of Shaanxi Universities "Multi-modal Data Mining and Fusion".

REFERENCES

[1] M. Seo, H. Hajishirzi, A. Farhadi, O. Etzioni, and C. Malcolm, "Solving geometry problems: Combining text and diagram interpretation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1466–1476.

[2] M. Sachan, A. Dubey, E. H. Hovy, T. M. Mitchell, D. Roth, and E. P. Xing, "Discourse in multimedia: A case study in extracting geometry knowledge from textbooks," *Computational Linguistics*, vol. 45, no. 4, pp. 627–665, 2020.

[3] P. Lu, R. Gong, S. Jiang, L. Qiu, S. Huang, X. Liang, and S. Zhu, "InterGPS: Interpretable geometry problem solving with formal language and symbolic reasoning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021, pp. 6774–6786.

[4] P. Lu, L. Qiu, W. Yu, S. Welleck, and K. Chang, "A survey of deep learning for mathematical reasoning," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023, pp. 14 605–14 631.

[5] S. Peng, D. Fu, Y. Liang, L. Gao, and Z. Tang, "GeoDRL: A self-learning framework for geometry problem solving using reinforcement learning in deductive reasoning," in *Proceedings of Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 13 468–13 480.

[6] J. Chen, J. Tang, J. Qin, X. Liang, L. Liu, E. P. Xing, and L. Lin, "GeoQA: A geometric question answering benchmark towards multi-modal numerical reasoning," in *Proceedings of Findings of the Association for Computational Linguistics: ACL/IJCNLP*, 2021, pp. 513–523.

[7] J. Chen, T. Li, J. Qin, P. Lu, L. Lin, C. Chen, and X. Liang, "UniGeo: Unifying geometry logical reasoning via reformulating mathematical expression," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 3313–3323.

[8] J. Cao and J. Xiao, "An augmented benchmark dataset for geometric question answering through dual parallel text encoding," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 1511–1520.

[9] M. Ning, Q.-F. Wang, K. Huang, and X. Huang, "A symbolic characters aware model for solving geometry problems," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 7767–7775.

[10] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford *et al.*, "GPT-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[11] J. Huang and K. C. Chang, "Towards reasoning in large language models: A survey," in *Proceedings of Findings of the Association for Computational Linguistics: ACL*, 2023, pp. 1049–1065.

[12] G. Mialon, R. Dessi, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Roziere, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz *et al.*, "Augmented language models: a survey," *Transactions on Machine Learning Research*, 2023.

[13] X. Wen, W. Nie, J. Liu, Y. Su, Y. Zhang, and A.-A. Liu, "CDCM: Chatgpt-aided diversity-aware causal model for interactive recommendation," *IEEE Transactions on Multimedia*, 2024.

[14] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg *et al.*, "Sparks of artificial general intelligence: Early experiments with GPT-4," *arXiv preprint arXiv:2303.12712*, 2023.

[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Proceedings of Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[16] M. Zhang, F. Yin, Y. Hao, and C. Liu, "Plane geometry diagram parsing," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, L. D. Raedt, Ed., 2022, pp. 1636–1643.

[17] W. Gan, X. Yu, T. Zhang, and M. Wang, "Automatically proving plane geometry theorems stated by text and diagram," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 07, p. 1940003, 2019.

[18] W. Gan, X. Yu, and M. Wang, "Automatic understanding and formalization of plane geometry proving problems in natural language: A supervised approach," *International Journal on Artificial Intelligence Tools*, vol. 28, no. 04, p. 1940003, 2019.

[19] M. Sachan, K. Dubey, and E. Xing, "From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 773–784.

[20] B. Song, G. Xiong, Z. Shen, F. Zhu, Y. Lv, and P. Ye, "Geometry problem solving based on counter-factual evolutionary reasoning," in *Proceedings of the 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023, pp. 1–6.

[21] M. Zhang, F. Yin, and C. Liu, "A multi-modal neural geometric solver with textual clauses parsed from diagram," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 3374–3382.

[22] M. L. Zhang, Z. Z. Li, F. Yin, and C. L. Liu, "LANS: A layout-aware neural solver for plane geometry problem," in *Proceedings of Findings of the Association for Computational Linguistics ACL 2024*, 2023, pp. 2596–2608.

[23] M.-L. Zhang, Z.-Z. Li, F. Yin, L. Lin, and C.-L. Liu, "Fuse, reason and verify: Geometry problem solving with parsed clauses from diagram," *arXiv preprint arXiv:2407.07327*, 2024.

[24] J. Zhang and Y. Moshfeghi, "GOLD: Geometry problem solver with natural language description," in *Proceedings of Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 263–278.

[25] L. Huang, X. Yu, F. Xiong, B. He, S. Tang, and J. Fu, "Hologram reasoning for solving algebra problems with geometry diagrams," *arXiv preprint arXiv:2408.10592*, 2024.

[26] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, "Solving olympiad geometry without human demonstrations," *Nature*, vol. 625, no. 7995, pp. 476–482, 2024.

[27] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le *et al.*, "Least-to-most prompting enables complex reasoning in large language models," in *Proceedings of the Eleventh International Conference on Learning Representations*, 2022.

[28] S. Imani, L. Du, and H. Shrivastava, "MathPrompter: Mathematical reasoning using large language models," in *Proceedings of ICLR 2023 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models*, 2023.

[29] V. Gaur and N. Saunshi, "Reasoning in large language models through symbolic math word problems," in *Proceedings of Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 5889–5903.

[30] A. Zhou, K. Wang, Z. Lu, W. Shi, S. Luo, Z. Qin, S. Lu, A. Jia, L. Song, M. Zhan *et al.*, "Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification," in *Proceedings of the Twelfth International Conference on Learning Representations*, 2023.

[31] M. Zong and B. Krishnamachari, "Solving math word problems concerning systems of equations with GPT-3," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 13, 2023, pp. 15 972–15 979.

[32] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Proceedings of Advances in Neural Information Processing Systems*, vol. 36, 2024.

[33] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 682–17 690.

[34] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," in *Proceedings of Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.
[35] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo *et al.*, "Solving quantitative reasoning problems with language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3843–3857, 2022.
[36] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.
[37] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "PAL: Program-aided language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 10 764–10 799.
[38] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *Transactions on Machine Learning Research*, 2023.
[39] J. He-Yueya, G. Poesia, R. Wang, and N. Goodman, "Solving math word problems by combining language models with symbolic solvers," in *Proceedings of the 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023.
[40] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh *et al.*, "SymPy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, 2017.
[41] J. X. Zhao, Y. Xie, K. Kawaguchi, J. He, and M. Q. Xie, "Automatic model selection with large language models for reasoning," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
[42] T. Liu, Q. Guo, Y. Yang, X. Hu, Y. Zhang, X. Qiu, and Z. Zhang, "Plan, Verify and Switch: Integrated reasoning with diverse x-of-thoughts," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
[43] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*. Springer, 2006, pp. 72–83.
[44] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European Conference on Machine Learning*. Springer, 2006, pp. 282–293.
[45] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.
[46] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[47] X. Fu, H. Deng, X. Yuan, and J. Hu, "Generating high coherence monophonic music using monte-carlo tree search," *IEEE Transactions on Multimedia*, 2022.
[48] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations*, 2022.
[49] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney *et al.*, "Openai o1 system card," *arXiv preprint arXiv:2412.16720*, 2024.

Yaxian Wang received her B.E. degree from Hunan Normal University, Changsha, China, in 2019. She received her Ph.D. degree in School of Computer Science and Technology from Xi'an Jiaotong University, Xi'an, China, in 2025. She is currently a lecturer in the School of Information Engineering at Chang'an University, Xi'an, China. Her research interests include visual question answering, cross-modal learning and interpretable reasoning.

Bifan Wei received his Ph.D. degree in computer science and technology from Xi'an Jiaotong University in 2014. He is currently a research fellow with the computer science and technology at Xi'an Jiaotong University. His research interests include web data mining, taxonomy learning and distance education.

Yinghong Ma received his B.E. and M.E. degrees from Xi'an Jiaotong University, Xi'an, China, in 2022 and 2025, respectively. His research interests include visual question answering and large language models.

Lingling Zhang is currently an associate professor in computer science at Xi'an Jiaotong University. She received her PhD degree in Computing Science from Xi'an Jiaotong University in 2020. She was a visiting student with the School of Computer Science, Carnegie Mellon University, working with Prof. A. Hauptmann. Her research interests include cross-media information mining, computer vision, zero-shot learning, and few-shot learning.

Xudong Jiang (Fellow, IEEE) received the B.E. and M.E. degrees from the University of Electronic Science and Technology of China (UESTC), and the Ph.D. degree from Helmut Schmidt University, Hamburg, Germany. From 1986 to 1993, he was a Lecturer with UESTC. From 1998 to 2004, he was with the Institute for Infocomm Research, A*STAR, Singapore, as a Lead Scientist, and the Head of the Biometrics Laboratory. He joined Nanyang Technological University (NTU), Singapore as a Faculty Member in 2004, where he served as the Director of the Centre for Information Security from 2005 to 2011. He is currently a professor with the School of EEE, NTU and serves as the Director of the Centre for Information Sciences and Systems. He has authored over 200 papers with over 50 papers in IEEE journals, including 9 papers in T-PAMI and 18 papers in T-IP. He served as IFS TC Member of the IEEE Signal Processing Society and Associate Editors for IEEE SPL and IEEE T-IP. Currently, Dr. Jiang is an IEEE Fellow. He serves as a Senior Area Editor for IEEE T-IP and Editor-in-Chief for IET Biometrics. His current research interests include image processing, pattern recognition, computer vision, machine learning and biometrics.

Henghui Ding received the B.E. degree from Xi'an Jiaotong University, Xi'an, China, in 2016. He received the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2020. From 2020 to 2022, he was a Research Scientist at ByteDance AI Lab in Singapore and a Postdoctoral Researcher at Computer Vision Lab of ETH Zurich in Switzerland. He was a Presidential Postdoctoral Fellow (Principal Investigator) at Nanyang Technological University (NTU) in Singapore. He is currently a tenure-track Professor at Fudan University and a member of Fudan Vision and Learning Lab. He has published over 60 papers in top conferences and top journals in the past 5 years since 2018. He serves as an Associate Editor of IET Computer Vision and a Guest Editor of Electronics. He also serves/served as an Area Chair of CVPR'24, a Senior Program Committee member of AAAI'(22-24) and IJCAI'23. His research interests include computer vision, language, and machine learning, with special emphasis on high-level image/video recognition and pixel-level scene understanding.

Jun Liu received his B.S. and Ph.D. degrees in computer science and technology from Xi'an Jiaotong University in 1995 and 2004, respectively. He is currently a professor of the School of Computer Science and Technology at Xi'an Jiaotong University in China. His current research focuses on data mining and text mining, and he has authored more than seventy research papers in various journals and conference proceedings. He serves as an Associate Editor of IEEE Transactions on Neural Networks and Learning Systems (TNNLS), an Area Editor of Information Fusion, and a Guest Editor for many technical journals, such as Information Fusion, IEEE Systems Journal, and Future Generation Computer Systems. He also acted as a conference/workshop/track chair at numerous conferences.