GEOMETRY PROBLEM SOLVING BASED ON COUNTERFACTUAL EVOLUTIONARY REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

As a representative topic in natural language processing and automated theorem proving, geometry problem solving requires an abstract problem understanding and symbolic reasoning. A major challenge here is to find a feasible reasoning sequence that is consistent with given axioms and the theorems already proved. Most recent methods have exploited neural network-based techniques to automatically discover eligible solving steps. Such a kind of methods, however, is greatly impacted by the expert solutions for training. To improve the accuracy, this paper proposes a new method called counterfactual evolutionary reasoning, which uses a generative adversarial network to generate initial reasoning sequences and then introduces counterfactual reasoning to explore potential solutions. By directly exploring theorem candidates rather than the neural network selection, the new method can sufficiently extend the searching space to get a more appropriate reasoning step. Through comparative experiments on the recent proposed Geometry3k, the largest geometry problem solving dataset, our method generally achieves a higher accuracy than most previous methods, bringing an overall improvement about 4.4% compared with the transformer models.

1 INTRODUCTION

As an essential subject in the secondary mathematical education, geometry problem solving is beneficial for the development of students' abstract thinking. Psychologists and educators believe that to achieve successful solutions for geometric problems, one needs high-level thinking abilities of symbolic abstraction and logical reasoning (Chinnappan, 1998; Nur & Nurvitasari, 2017). These abilities can partially reflect the human's a mental activity and thus the level of his intelligence. Therefore, as a long-standing challenge in natural language processing (NLP) and theorem proving (TP), the automated geometry problem solving is viewed as an ideal scenario to test whether (or to what extent) an algorithm achieves the human's intelligence (Hopkins et al., 2017).

Generally, a geometry problem is typically given by an illustrative figure with a paragraph of natural language description. And the people's acquisition of its solution may roughly experience a problem understanding and a symbolic reasoning. The former stage usually exploits formal representations with explicit semantics, which characterizes the human's cognitive (re-)construction of the geometric elements with relationships when encountering specific problems. Such a process involves a data fusion of multiple detected results both from the figure and the textural descriptions, so that the problem is formally and consistently represented. The latter stage mostly focuses on logical reasoning, which simulates the human's cognitive deliberation and rational thinking. A feasible reasoning sequence, with each step being a known theorem as one step forward the final solution, is obtained in this stage according to the knowledge base that is usually composed of axioms and proved theorems.

To limit the scope, this paper mainly focuses on the symbolic reasoning stage of the task. As can be seen in the next section, recent methods have mostly exploited neural network-based techniques to automatically discover eligible solving steps. Such a kind of methods, however, is greatly impacted by the expert training data, which limits the searching space for potential solutions. To improve the accuracy, this paper proposes a new method called the counterfactual evolutionary reasoning for the geometric or other problem solving. Specifically, the problem solver starts with a generation of initial reasoning sequences, using a pre-trained generative adversarial network (GAN). Taking these sequences as initial solution candidates, counterfactual reasoning is introduced for interven-

tions to investigate more potential solutions. By a direct operation of the reasoning sequence, our method is able to sufficiently extend the exploration space other than expert training solutions. The intervention is then heuristically evolved to optimally select a final solution by setting an appropriate evaluation criterion. Comparative experiments on recent proposed Geometry3k, the largest geometry problem solving dataset, indicate that our method generally achieves higher accuracy than most previous methods, especially the transformer models. In summary, the main contribution of this paper is two-fold. First, we introduce the counterfactual reasoning into the geometry problem solving. This is able to directly operate the solution sequence, so that the local optima of neural network (NN) models trained by expert data can be easily jumped out. Second, an evolutionary mechanism is introduced to enhance the heterogeneity of solutions, promoting the exploration of new potential solutions. This could alleviate the complex training of the sequential models (such as the transformer).

2 Related Work

As alluded before, the geometry problem solving mainly involves the abstract problem understanding and the symbolic reasoning. Many fruitful researches come from the field of theorem proving (Wu, 1986; Chou et al., 1996; Yu et al., 2019). This section will summarize related researches according to these two categories.

2.1 Problem Understanding

The problem understanding includes a diagram and text parsing. Kahou et al. introduced FigureQA, a visual reasoning corpus to synthesize questions from 15 templates. They studied the visual reasoning task that can be expanded to diagram parsing and reasoning (Kahou et al., 2018). Lewis et al. introduced generative models for the language understanding and reasoning (Lewis & Fan, 2019). Seo et al. first introduced NLP techniques to extract the representations of geometry problems (Seo et al., 2014; 2015). They proposed a method for understanding the geometric graphs by identifying the elements in the diagram with their relative spatial places and geometry properties, and matching them by maximizing the consistency between the textual description and the visually identifiable elements. Hopkins et al. combined the methods of machine learning and logical reasoning to propose a problem solver system called EUCLID (Hopkins et al., 2017). Their system could propagate uncertainty from multiple sources (e.g. coreference resolution or verb interpretation) until it can be confidently resolved. A second approach to build a neural sequence-to-sequence translator to map questions to sequences, with an arithmetic tree adopted, was proposed by Roy and Roth (Roy & Roth, 2018). Their work could be viewed as an intelligent parser of math expressions. The idea of combining reasoning and machine learning in this system was innovative, yet achieved only marginal improvements over random baselines. Kembhavi et al. proposed to use diagram parse graphs (DPGs) to encode elements with their relationships (Kembhavi et al., 2016). They formulated the problem of graph syntactic parsing as a task of learning to infer the DPG that best interprets the graphs.

2.2 Problem Solving

Huang et al. introduced GamePad to apply machine learning in the theorem proving. They solved the position evaluation and tactic prediction task (Huang et al., 2019). Zhang et al. proposed a neural network component that allows a robust object counting in natural images. This component can solve visual question answer problems with a higher accuracy (Zhang et al., 2018). By treating the geometric relations as constraints, Seo et al. proposed the first automatic geometry problem solver, GEOS, which formulated the task as an optimization problem and found a solution by satisfying all the constraints (Seo et al., 2014; 2015). However, it was not a reasoning method that used the relationship between the elements. In the work from Kembhavi et al., the semantic interpretation of graphs and the reasoning about elements with their relationships were studied in the context of graph question answering (Kembhavi et al., 2016). They defined the task of graph parsing and reasoning. Sachan et al. collected theorems from multiple textbooks and parsed them into horn clause rules. Such a collection of theorems solved the problem with less annotation and low redundancy (Sachan et al., 2017; Sachan & Xing, 2017; Sachan et al., 2020). Moreover, re-use of these horn clauses could reduce the computational complexity and improve the accuracy. The work sufficiently exploited the

proved theorems in the reasoning process, but it did not provide users with readable proof steps, and the search process for applying theorems was not fully controllable. Zhu et al. systematically described the whole process of geometry problem solving. By using predicates and parameters, they developed a textual and graphic parser to accurately extract the geometric relationship between elements (Lu et al., 2021). In the reasoning stage, they further built a predictor to construct a sequence of theorems that would be applied to solve the problem. Compared with other work, the reasoning is more interpretable and the predicted theorem sequence can reduce the searching space. However, it was the compressed searching space that may bring further improvement of accuracy. Seohyun et al. introduced the NeurQuRI to answer questions based on the reasoning with multiple different constraints (Back et al., 2020).

3 Counterfactual Evolutionary Reasoning for Theorem Sequence Optimization

Generally, the NN-based model such as the transformer requires a complex training using expert solutions. It highly depends on the heuristics implied by the training data, which may narrow the searching space when getting a potential solution. To improve the accuracy, this section will elucidate our counterfactual evolutionary reasoning method for the geometry problem solving.

3.1 Problem and Solution Representation

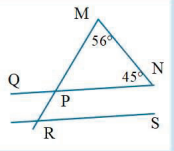
Problem Text	Diagram	Choices	Text Literals (Logic Forms)	Diagram Literals (Logic Forms)
In triangle MNP, ∠PMN=56°, ∠PNM=45°, QN is parallel to RS. Find ∠PRS.		A. 56° B. 45° C. 79° D. 101° Answer: C	Triangle(M,N,P) Equals(MeasureOf(Angle(P,M,N)),56) Equals(MeasureOf(Angle(P,N,M)),45) Parallel(Line(Q,N),Line(R,S)) Find(MeasureOf(Angle(P,R,S)))	Triangle(M,N,P) Equals(MeasureOf(Angle(P,M,N)),56) Equals(MeasureOf(Angle(P,N,M)),45) PointLiesOnLine(P,Line(Q,N)) PointLiesOnLine(P,Line(M,R)) Parallel(Line(Q,N),Line(R,S))

Figure 1: A Geometry Problem Example.

A geometry problem P is usually defined as a tuple (t, d, c), where t is a textual description, d is a diagram image and $c = \{c_1, c_2, c_3, c_4\}$ is a set of multiple result candidates in the format of numerical values. Given the text t and diagram d, an automated solver is required to predict the correct answer $c_i \in c$. We use a predicate to represent a geometric shape entity, geometric relation, or arithmetic function. A literal is an application of one predicate to a set of arguments like variables or constants. A set of literals makes up the semantic description from the problem text and diagrams in the formal language space Ω. A primitive is a basic geometric element like a point, a line segment, etc.. Figure 1 shows a simple geometry problem as an example. As illustrated, the primitive $Triangle$ defines a triangle and the literal $MeasureOf$ applies the angle to a constant (say 45 or 56).

Given a geometry problem formally represented as a set of literals, the objective of a solver is to find a feasible solution that results in an answer from choices. A solution is defined as a theorem sequence where a group of selected theorems from the axioms and proved theorems are arranged in a certain order. Each theorem has a premise and a conclusion. When applying a theorem, the original problem assumptions and the conclusions achieved from previous applied theorems will be matched with its premise. If the match is successful, the new conclusion of the applying theorem is obtained and is added into achieved conclusions. This operation extends the theorem sequence. When a sequence is able to finally result in an eligible answer, it is called a feasible solution. For example, the problem in Figure 1 can be solved by sequentially applying the interior angles of a triangle and the parallel line theorems (elicits $Equals(MeasureOf(Angle(M, P, N), 79))$ and $Equals(MeasureOf(Angle(P, R, S), 79))$ respectively). These two theorems are indexed as 1 and 10 in the axiom and theorem base. Thus, the theorem sequence $< 1, 10 >$ is a feasible solution. Note that the feasible solution is not unique.

3.2 GENERATION OF INITIAL SOLUTIONS USING GAN

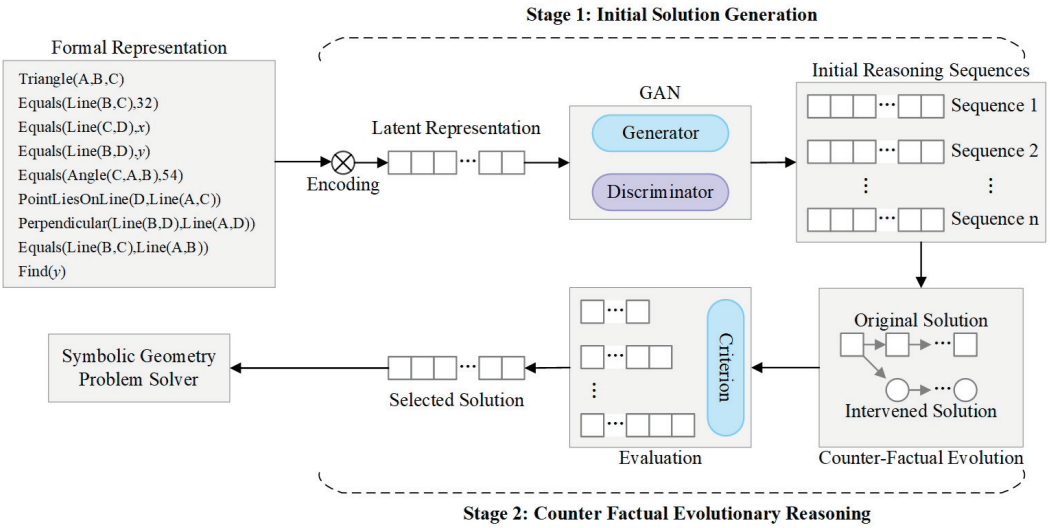


Figure 2: Counterfactual Evolutionary Reasoning for Geometry Problem Solving.

Our proposed method consists of two stages, the initial solution generation and the counterfactual evolutionary reasoning (see Figure 2). After receiving the formal representation (literals from both text and diagram) of a problem, the solver sends its latent encoding to a pre-trained GAN to get a collection of initial reasoning sequences. The initial solutions start by random sampling and are "filtered" by a pre-trained generator network. They both retain a certain degree of heterogeneity and include some expert heuristics, providing the subsequent counterfactual evolution a suitable start point. Compared with other sequence-to-sequence models, GAN is able to keep a good balance between the heterogeneity of initial solutions and the training efficiency. Then, a probabilistic intervention is applied to these initial solutions and an iterative evolution is conducted to optimize the reasoning sequences. Please note that we use a Symbolic Geometry Problem Solver that is provided by the original Inter-GPS research, to check whether the premise of a given theorem matches that of the problem to be solved. This symbolic solver can neither generate solutions by itself, nor impact the solution construction at all. Each solution is a theorem sequence that determines which theorems and in what order to be applied to solve a specific geometry problem. Its generation all depends on the Counter-Factual Evolution, Evaluation and Selection drawn in the figure. Therefore, using such a symbolic solver assistant does not influence our solver's performance.

The initial solution generation aims to get some reasoning sequences that are relevant to the given problem. This can set a suitable start point for the subsequent solution searching. Our problem solver adopts a conditional generative adversarial network (cGAN) to complete such a task. For each problem, the original formal representation is embedded into a lower-dimensional latent space by encoding the top k most frequent predicates in the training data. The latent encoding is used as a condition, which is a common part of the inputs both for the generator and discriminator networks (see Figure 3). For the generator training, the embedded condition is concatenated with a randomly generated sequence as the input. The "randomly generated sequence" is obtained by a purely random sampling over the whole theorem space. It can be viewed as the most chaotic solution, without any prior heuristics. After several convolution and full connected layers, the generator reduces the solution's randomness and constraints it "near" the expert training data, as an output solution sequence. The loss for back propagation is computed by a pseudo label returned by the discriminator. For the discriminator training, its input is the embedded problem condition concatenated with a generated (fake) or an expert (real) solution. The numerical output is between 0 and 1, indicating how the input is "real". The label for back propagation is set as 0 for a generated input and 1 for an expert input. To preserve the dependence between theorems in training data, we further adopt an attention module (drawn as the Positioning Encoding in the figure) to heuristically learn the theorem relative order in the sequence. The pooling operation has a property of transitional invariance. It is used to retain the significant dependence of adjacent theorems. In addition, we

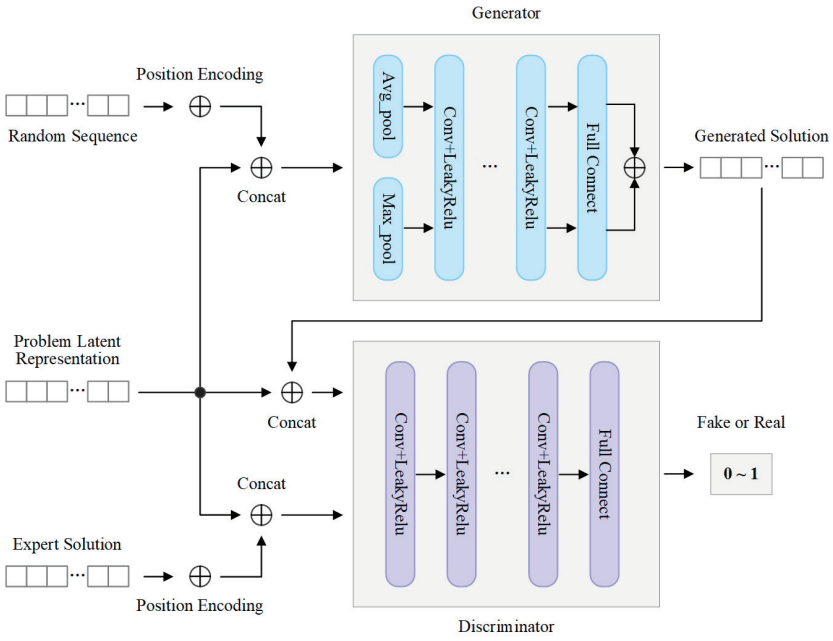


Figure 3: Structure of the Conditional Generative Adversarial Network for Initial Solution Generation.

refer to the Wasserstein GAN to fine-tune the network structure in order to avoid the mode collapse (Arjovsky et al., 2017). The fine-tune includes a direct exploitation of the output rather than the log form as its loss function, a removal of activation functions in the last layer of discriminator, a use of the RMSProp optimizer instead of the momentum-based ones, and a clamp of the discriminator parameters to (-0.01, 0.01).

3.3 Counterfactual Evolutionary Reasoning for Sequential Reasoning

The second stage in our problem solver is the counterfactual evolutionary reasoning. Its objective is to explore potential solutions so that the theorem sequence for a given problem can be optimized. The basic idea behind is to preserve the sequential theorem dependence from training data rather than a stochastic search in a solving process. It can sufficiently exploit heuristics from the expert solutions. Unlike the NN-based methods where theorems are encoded and selected by a pre-trained neural network, we introduce the intervention of counterfactual reasoning to directly manipulate the theorem candidates (Looveren & Klaise, 2020). Here, the "facts" are those existing solutions, which reflect the human expert experience for the encountered problems. The contextual dependence between theorems in a particular solution sequence implies their endogenous causal relationships. The "counter facts" are those different potential reasoning paths with the implicit causal dependence retained. By manually intervening theorem candidates instead of an NN selection, our prover can infer a possible intuitive reasoning. This is able to enhance the heterogeneity of solutions so that the searching space is extended.

Our counterfactual evolutionary reasoning includes the intervention and evaluation. Given a theorem sequence $< \cdots, a, b, c, \cdots >$ as shown in Figure 4, an intervention point is randomly determined, say the theorem b. An intervention is performed then by replacing b with another possible theorem b^*, which is computed as

$$b^* = \arg\max_{\hat{b}} \int_{\Omega} P\{\hat{b}, u|b\} R(\hat{b}) du \tag{1}$$

where

$$P\{\hat{b}, u|b\} = P\{\hat{b}|u, b\} \cdot P\{u|b\} = P\{\hat{b}|u\} \cdot P\{u|b\} = \frac{P\{u\}}{P\{b\}} \cdot P\{\hat{b}|u\} \cdot P\{b|u\} \tag{2}$$

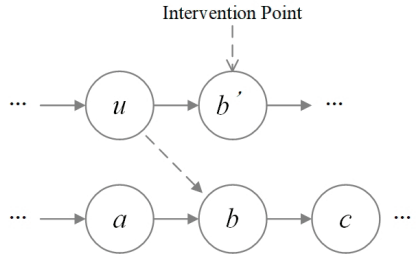


Figure 4: Intervention of Counterfactual Reasoning.

In the above equations, $P\{b|u\}$ stands for the probability that b is the direct successor of u in the expert training data. $R(b)$ is a reward of b, which characterizes the effect of changing b to $\hat{b}$ for solving. It is set to be 1 in the experiment to reduce more calculations. $P\{u\}$ and $P\{b\}$ are the probabilities emerged in the training data set. $P\{\hat{b}, u|b\}$ represents the intervention probability that b is replaced by $\hat{b}$ provided that b and $\hat{b}$ have a common parent u. The first equal sign in Eq. (2) is the probability chain rule. The second equal sign holds because of the independent given the common parent u. The third equal sign is Bayes theorem. Ω means the integration over the whole predicate space determined by the training data. Note that in the above method, we use a single intervention point for each reasoning sequence, but this can be easily generalized to multiple interventions.

The probabilistic intervention explores heterogenous solutions according to the current theorem sequences. And an evaluation for these evolved sequences would lead to a final optimized solution for the given problem. In our prover, each evolved sequence is firstly checked whether its theorems can be applied to the original problem representations and the intermediate conclusions. Specifically, if the original problem representations together with intermediate conclusions (for the first theorem in a sequence, there is no intermediate conclusion) can match the premise of a theorem, then its conclusion holds and we call this theorem applicable. The achieved conclusion is added into intermediate conclusions for the subsequent theorem check. By iteratively checking each theorem, a solution sequence is evaluated as

$$fit = \frac{n_a}{N} + k_p \cdot \frac{n_r}{N} \tag{3}$$

where N represents the total number of theorems in the sequence. n_a is the number of theorems that are applicable and n_r is the maximally repeated times of a theorem in the sequence. $k_p \in (-1, 0)$ is a penalty coefficient. The above fitness function characterizes a solution from two aspects, the applicability and the repetition rate. The former metric is usually pertinent to solving the problem. For example, theorems about circles may probably not be valid for the problems about triangles. Thus, more applicable theorems may result in a feasible solution with a higher probability. The latter metric measures the redundancy of a solution. Intuitively, a less repetition rate means fewer unnecessary reasoning steps. Thus, the solution is more sufficient. Given a geometry problem, our counterfactual evolutionary reasoning is iteratively conducted for several rounds and the solution with highest fitness according to Eq. (3) is selected as the final result. The pseudo code of the algorithm is summarized in Algorithm 1.

3.4 Convergence Proof

In order to valid our counterfactual evolutionary reasoning in theory, this subsection proves the convergence of the algorithm. We first define the reachability as follows.

Definition 1 (Reachability)**.** *Assume that for a given problem, x and y are two reasoning sequences in the training dataset D. We call that y is reachable from x if there exists a series of interventions $T_1, T_2, \cdots, T_m \in T$ such that $y = T_m(T_{m-1}(\cdots T_1(x)))$. Here, T is the set of all possible interventions.*

The reachability manifests that a particular theorem sequence has a positive probability to transfer into the other one. Accordingly, we can further define one-step reachability as $y = T_1(x)$. We now prove the convergence and global optimality of our counterfactual evolutionary reasoning.

Algorithm 1 Counterfactual Evolutionary Reasoning.

Input: expert training data, a geometry problem, a penalty coefficient k_p, a maximum iteration number, state rewards $R(b)$.
Output: a final solution.

```
Generate initial theorem sequences, Solutions, using a pre-trained cGAN and the problem formal representations;
while the iteration number does not reach to maximum, do
    EvoSeq ← {};
    for each theorem sequence in Solutions do
        Randomly select an intervention point;
        Conduct an intervention according to Eq. (1) and Eq. (2);
        Add the intervened theorem sequence into EvoSeq;
    end for
    Select top sequences from Solutions ∪ EvoSeq according to Eq. (3) as new Solutions;
end while
return the theorem sequence with the topmost fitness as the final solution.
```

Theorem 1. *Assume the formal language space Ω and the training dataset D are both finite, the counterfactual evolutionary reasoning will finally converge to an optimal solution x^*. Furthermore, if any theorem sequence y in the solution space S is reachable from a specific x in D, then*

$$\lim_{k\to\infty} P\{x^*(k)\ is\ globally\ optimal\} = 1$$

k is the iteration number.

Proof. The convergence to x^* is intuitive. Since each iteration selects top solution candidates according to their fitness, there is $fit\{x^*(k+1)\} \geq fit\{x^*(k)\},\ k = 1, 2, \cdots$. With a finite Ω and D, the algorithm will stop after certain iterations (say n), leading to $fit\{x^*(n)\} > \cdots > fit\{x^*(1)\}$ and $x^* = x^*(n)$. For the global optimality of x^*, construct a Markov chain $Z(k)$ as follows. The state $z(k)$ is defined as the aggregation of all the theorem sequences after the k-th iteration. And we arbitrarily define a state z_0 if there is at least one global optimum in it. According to our algorithm, if a global optimum has been selected into the solution candidates, it will always remain among them. Therefore, z_0 is absorbing. By the reachability of $\forall y \in S$ from $\exists x \in D$, there is

$$\lim_{k\to\infty} P\{z(k) = z_0\} = 1$$

□

The proof demonstrates that the optimal solution achieved by our model is probabilistically global, if it can be reached from the expert solution x. In other words, there is at least one intervention sequence that transforms x to x^*. For a particular theorem in x, say a_0, assume it is transformed into a_n in x^* after n interventions. The transition is $a_0 \xrightarrow{T_1} \cdots \xrightarrow{T_n} a_n$. Then according to the intervention given by Eq. (1) and Eq. (2), the expert data needs to contain a common antecedent node $u_{0,1}$ from the reasoning steps $u_{0,1} \to a_0$ and $u_{0,1} \to a_1$ in two specific solutions. Similarly, other common antecedent nodes $u_{1,2}, u_{2,3}, \cdots$ need to be also included, which gives each transition a positive probability. Thus, it is better to use large-scaled expert solutions in practice to involve as more dependences as possible.

4 Experiments on Geometry Problem Solving

To verify the proposed method, we conduct experiments on Geometry-3K, the largest dataset of publicly available geometry problems, and compare our algorithm with other existing methods (Lu et al., 2021). This section will briefly introduce the dataset and experiment setting, followed by the report of our comparative studies.

4.1 DATASET AND EXPERIMENT SETTING

The Geometry-3K dataset contains 3002 geometry problems, including 2101 for training, 300 for validation, and 601 for test. All problems are collected from popular high school textbooks for grades 6-12. Apart from lines, triangles, quadrilaterals and circles, the problems also involve polygons and irregular quadrilaterals. The large scale of the dataset and the high diversity of problem types make it one of the representative testbeds for problem solving algorithms.

As the Geometry-3K dataset is proposed by Zhu, et al, it is natural to compare our method with their transformer-based model, the problem solver in Inter-GPS (Lu et al., 2021). Other baselines include the Q-only method that uses only a gated recurrent convolutional network to understand and solve the textual description of the question (Chou et al., 1996), the I-only method that only uses the residual network ResNet-50 to extract information from the image of the question and solve it (He et al., 2016), the Q+I method that combines the above two methods, the RelNet to model and reason the relationship between entities (Bansal et al., 2017), the FiLM to perform visual reasoning on topic icons (Perez et al., 2018), the FiLM-BERT that uses BERT model to perform language reasoning (Devlin et al., 2018), and FiLM-BART that uses the BART model (Lewis et al., 2020).

In our experiments, the knowledge base contains 17 theorems. The maximum length of each generated sequence is set to be 30. But this does not mean that our reasoning for each problem contains 30 theorems at most. As a matter of fact, such a theorem sequence after the intervention and evolution may probably exceed that length. The evolved theorem sequence is sent to a symbolic geometry problem solver for a validation (Lu et al., 2021). After receiving a solution candidate, the symbolic solver performs a symbolic reasoning using the theorems one by one. If the geometry problem cannot be solved by the solution candidate, a low-first search will be conducted by directly using the basic theorems in the knowledge base. We set the maximum reasoning step is 100, and if the problem is still unsolved after that, the symbolic solver finally gives a random guess of the 4 choices. To test each method, we randomly select 100 successive problems from the test dataset, because running all the test problems is quite time consuming. The 100 successive but not stochastically chosen problems can avoid the sampling bias that those "easy" problems for the solver are selected. The proportion of successfully solved problems is defined as the accuracy. For the problems that cannot be solved in these 100 test problems, we directly treat their accuracy as 25% (uniformly choose an answer from the 4 choices, and this operation is the same as Inter-GPS).

4.2 COMPARATIVE STUDIES WITH BASELINES

The experiment results with baselines are shown in Table 1. As can be seen, the theorem sequences generated by the GAN outperform others in Ratio, Triangle and Quad. But its overall performance is not as good as the original Inter-GPS solver. When the counterfactual evolutionary reasoning incorporated, 5 of 9 metrics have reached the best performance. However, our method performs not stably for different types of problems. It may result from the unbalanced distribution of problem types in the test dataset. For example, with much fewer problems about Area and Other, it is easier to bring a low or high accuracy. Despite the higher variance, the overall evaluation has increased about 4.4% compared with the original Inter-GPS. These results indicate that our counterfactual evolutionary reasoning is valid in searching for optimal decision sequences for problem solving.

To further compare the properties of generated solutions between Inter-GPS and counterfactual evolutionary reasoning, we further compute statistical metrics in Table 2. Clearly, our method gets fewer average steps and less average time for all problems, whereas these two metrics grow larger than the Inter-GPS for solved problems. It manifests that our method prefers to generate longer theorem sequences. By contrast, solutions from our method have fewer steps than those from the GAN without counterfactual reasoning. It shows the effect of the fitness optimization. The average time for training and problem solving is listed in Table 3. As can be seen, the GAN training in our method requires less time, but the evolutionary reasoning takes much more time cost. Thus, our method is more computationally expensive.

5 CONCLUSIONS AND FUTURE WORK

Geometry problem solving is a representative topic in natural language processing and automated theorem proving. This paper proposes a new method by introducing a counterfactual reasoning

Table 1: Experiment Results with Baselines.

Method	All	Angle	Length	Area	Ratio	Line	Triangle	Quad	Circle	Other
Q-only	25.3	29.5	21.5	28.3	33.3	21.0	26.0	25.9	25.2	22.2
I-only	27.0	26.2	28.4	24.5	16.7	24.7	26.7	30.1	30.1	25.9
Q+I	26.7	26.2	26.7	28.3	25.0	21.0	28.1	32.2	21.0	25.9
RelNet	29.6	26.2	34.0	20.8	41.7	29.6	33.7	25.2	28.0	25.9
FiLM	31.7	28.7	32.7	39.6	33.3	33.3	29.2	33.6	30.8	29.6
FiLM-BERT	32.8	32.9	33.3	30.2	25.0	32.1	32.3	32.2	34.3	33.3
FiLM-BART	33.0	32.1	33.0	35.8	50.0	34.6	32.6	37.1	30.1	37.0
Inter-GPS (No GT)	55.1	58.2	57.3	30.2	58.3	63.0	63.5	51.7	41.8	29.6
GAN	54.4	57.0	56.3	32.1	58.3	56.8	65.6	52.4	34.3	25.9
GAN+CER (ours)	**59.5**	**58.3**	**71.5**	11.4	50.1	51.9	**73.3**	46.2	**42.0**	**88.9**

Table 2: Some Properties of Generated solutions of Inter-GPS and Counterfactual Evolutionary Reasoning.

Method	Inter-GPS (No GT)	GAN	GAN+CER (ours)
Accuracy (%)	55.07±0.1	54.4±0.5	59.5±0.75
Average Steps for All Problems	39.97	49.05	39.94
Average Steps for Solved Problems	7.33	16.41	10.72
Average Time for All Problems (sec.)	57.13	66.69	45.76
Average Time for Solved Problems (sec.)	10.64	13.85	15.73

Table 3: Average Time for Training and Problem Solving.

Method	Inter-GPS (No GT)	GAN	GAN+CER (ours)
Pseudo-Optimal Solution Generation for Training Data	About 20 hours	About 20 hours	About 20 hours
Transformer Training	812 seconds	-	-
GAN Training	-	376 seconds	376 seconds
Problem Solving for a Test Sample	2 seconds	2 seconds	1602 seconds

and evolution mechanism. The method uses a generative adversarial network to generate initial reasoning sequences and then an intervention to explore potential solutions. By directly operating the theorem candidates rather than the neural network selection, our method can sufficiently extend the exploration space to get a more appropriate reasoning step. The method is validated on the Geometry-3K dataset, bringing an overall 4.4% improvement of accuracy compared with the most advanced transformer model.

Though the counterfactual evolutionary reasoning outperforms other methods in an overall evaluation, it still suffers from some critical issues. First, the generated theorem sequence has more reasoning steps than Inter-GPS, which indicates that the solution has more redundant theorems. A heuristic setting in the evaluation may further reduce the solution's complexity. Second, albeit the proposed method is more explorable and adaptive to the problem, it is more computational expensive as well. As the transformer model requires a complicated training, the training of the GAN in this paper is much easier. However, this essentially shifts the computation to the later evolutionary stage. Since the solver needs interventions and evolutions for every test problem, it may suffer from a heavier computational burden. Setting a larger knowledge base via a parallel/cloud computing in our future work may help reduce the evolutionary search and compress the solving time.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Leon Bottou. *Wasserstein GAN*, 2017. URL https://arxiv.org/abs/1701.07875.

Seohyun Back, Sai Chetan Chinthakindi, Akhil Kedia, Haejun Lee, and Jaegul Choo. Neurquri: Neural question requirement inspector for answerability prediction in machine reading comprehension. In *In Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 26-May 1, 2020.

Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. *RelNet: End-to-End Modeling of Entities & Relations*, 2017. URL https://arxiv.org/abs/1706.07179.

Mohan Chinnappan. Schemas and mental models in geometry problem solving. *Educational Studies in Mathematics*, 36(3):201–217, 1998.

Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants. *Journal of Automated Reasoning*, 17(3):325–347, 1996.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *In Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, New Orleans, LA, USA, June 2018. URL https://arxiv.org/abs/1810.04805.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti, and Vidur Joshi. Beyond sentential semantic parsing: Tackling the math sat with a cascade of tree transducers. In *In Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pp. 795–804, 2017.

Daniel Huang, Prafulla Dhariwal, Dawn Song, and Ilya Sutskever. Gamepad: A learning environment for theorem proving. In *In Proceedings of the 7th International Conference on Learning Representations*, New Orleans, FL, USA, May 2019.

Samira Ebrahimi Kahou, Adam Atkinson, Vincent Michalski, Akos Kadar, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. In *In Proceesings of the 6th International Conference on Learning Representations*, Vancouver, Canada, April 2018.

Aniruddha Kembhavi, Michael Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. *A Diagram is Worth a Dozen Images*, 2016. URL https://arxiv.org/abs/1603.07396.

Mike Lewis and Angela Fan. Generative question answering: Learning to answer the whole question. In *In Proceedings of the 7th International Conference on Learning Representations*, New Orleans, FL, USA, May 2019.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 7871–7880, Seattle, WA, USA, July 2020.

Arnaud Van Looveren and Janis Klaise. *Interpretable Counterfactual Explanations Guided by Prototypes*, 2020. URL https://arxiv.org/abs/1907.02584.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, Bangkok, Thailand, August 2021.

Andi Saparuddin Nur and Evy Nurvitasari. Geometry skill analysis in problem solving reviewed from the difference of cognitive style students junior high school. *Journal of Educational Science and Technology*, 3(3):204–210, 2017.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3942–3951, 2018.

Subhro Roy and Dan Roth. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics (TACL)*, 6:159–172, 2018.

Mrinmaya Sachan and Eric Xing. Learning to solve geometry problems from natural language demonstrations in textbooks. In *In Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, pp. 251–261, 2017.

Mrinmaya Sachan, Kumar Dubey, and Eric Xing. From textbooks to knowledge: A case study in harvesting axiomatic knowledge from textbooks to solve geometry problems. In *In Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pp. 773–784, 2017.

Mrinmaya Sachan, Avinava Dubey, Eduard Hovy, Tom Mitchell, Dan Roth, and Eric Xing. Discourse in multimedia: A case study in extracting geometry knowledge from textbooks. *Computational Linguistics*, 45(4):627–665, 2020.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. Diagram understanding in geometry questions. In *In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2831–2838, 2014.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In *In Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1466–1476, 2015.

Wen-Tsun Wu. Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated Reasoning*, 2(3):221–252, 1986.

Xinguo Yu, Mingshu Wang, Wenbin Gan, Bin He, and Nan Ye. A framework for solving explicit arithmetic word problems and proving plane geometry theorems. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(07):1940005, 2019.

Yan Zhang, Jonathon S. Hare, and Adam Prugel-Bennett. Learning to count objects in natural images for visual question answering. In *In Proceesings of the 6th International Conference on Learning Representations*, Vancouver, Canada, April 2018.