

---

## Responses for the Paper Submitted to ICLR 2023 (Paper ID: 6413)

### Geometry Problem Solving based on Counterfactual Evolutionary Reasoning

We sincerely thank the reviewers for their constructive comments. All the questions and confusions are responded in details with an example provided in supplementary material. We tried to show all the technical details and do hope the revision could be re-considered for potential publication.

#### Reviewer zfWo

Weakness: I also have some questions about the design of the proposed approach.

(1) What is the intuition behind Eqn1 and Eqn2? If the main goal of evolution is to escape from the impact of expert solutions, why not just replace one theorem in the solution with a random theorem?

Thanks for your question. The main idea behind is to replace the theorem  $b$  with another one that has a common antecedent node  $u$ . The probability of such replacement is computed by these two equations. Admittedly, a simple random replacement can also implement the intervention, but it would be much more inefficient when the searching space is large. The expert solution here essentially provides a heuristic direction of the intervention.

(2) Why do you use GAN to generate the initial solution? If the discriminator is trained with expert solutions, why not just drop the discriminator and train the generator with expert solutions? Have you tried this method?

Thanks for your questions. Initial solution generation using GAN, rather than the generator network alone, is to keep a certain degree of heterogeneity. Heterogenous solutions can expand the searching space, avoiding an early convergence to a local optimum. However, a pure random initialization may result in a chaotic search and may bring a loss of computational efficiency (according to our pre-experiments). Therefore, to seek the balance, we use the discriminator to “softly” constrain initial solutions “near” the expert ones.

(3) Do you keep evolving solutions until finding the optimal one?

Not exactly. The optimal solution is theoretical. It may be reached via a very long-time computation in practice (especially when the searching space is quite large). Therefore, in this paper, the evolution stops when either the iteration reaches a maximum number or the solution fitness exceeds a given threshold. The latter stop condition may define an “optimal” solution as you said.

(4) It seems that you are querying the prover if the theorems of a solution are applicable during evolution. If you can query the prover multiple times, why not set up a search procedure to try different solutions in the prover until a solution is found within a fixed time budget? Also, if the baseline could only query the prover once for each problem, the comparison is unfair.

Thanks for your good question. We think the “prover” you mentioned here refers to the Symbolic Geometry Problem Solver (Symbolic Solver for short) in Figure 2. Please note the Symbolic Solver is provided by the original Inter-GPS research, and it cannot generate solutions by itself. More like an assistant (not a validator), it is used to check whether the premise of a given theorem matches that of the problem to be solved. The Symbolic Solver is not necessary (we can easily develop a module to complete the same function). It does not impact the solution at all. The 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 (according to Equation (3)) drawn in Figure 2.

---

More like an assistant (not a validator), it is used to check whether the premise of a given theorem matches that of the problem to be solved. The Symbolic Solver is not necessary (we can easily develop a module to complete the same function). It does not impact the solution at all. The solution is a theorem sequence that determines which theorems and in what order to be applied to solve a specific geometry problem.

Actually, in the original Inter-GPS research, the transformer-based Theorem Predictor (the module for solution generation) also calls the Symbolic Solver many times, after getting the theorem sequence. And the Symbolic Solver does not help modify the solution at all. We guess the additional arrow from the Formal Representation box to the Symbolic Geometry Problem Solver box made you misunderstand. We have corrected this figure. Hope it is clear now.

### **Reviewer Zd2j**

Weaknesses:

Most of the paper is very difficult to follow, and there is not enough background or details given for the reader to understand what was done. (Note that the reviewer is quite familiar with the literature on proof synthesis in other systems, e.g., work on problem solving in generic systems like Metamath and Lean, and is also experienced in Olympiad geometry problems.)

Apologize to bring you some confusion in your review. Due to the page limit, we provide an explicit example in the supplementary material, which can be also referred to in the original Inter-GPS reference. Hope it clearly clarifies our research issue.

Specifically:

- It is essential to have at least one example of what the reasoning sequences, theorems, and evolution steps look like.

Thanks for your constructive suggestion. Due to the page limit, the concrete example is elucidated in the supplementary material. And we have also added some details in the manuscript for a better illustration.

- No code is provided, which makes it impossible to check the parts that are unclear in the text by referring to the implementation.

Thanks for your comments. We have provided our source code in Github:

<https://github.com/pySourceCode/Geometry-Problem-Solving-based-on-CER.git>

Hope it can be helpful for your review.

- I could not understand the evolutionary algorithm in 3.3:
  - In (2), how are  $P(b|u)$  and  $P(\hat{b}|u)$  computed? Where does the reward  $R(b)$  come from? (As far as I understood, these are not actually querying a model, but using dataset statistics, which, I suppose, is responsible for the very low efficiency.)

Thanks for your good question.  $P\{b|u\}$  ( $P\{\hat{b}|u\}$ ) is computed by the frequency that  $b$  ( $\hat{b}$ ) subsequently applies after  $u$  in the expert solutions.  $R\{b\}$  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.

As you said, the evolution is a statistical model but a querying one. Compared with a heuristic search, it may suffer from a low efficiency. But when the searching space is quite large (where heuristic search is time consuming or even impossible), our method can focus the exploring area around the expert solutions and thus can avoid a “blinded” search.

- Can it be measured whether the conditions in Theorem 1 are satisfied in practice? If I understand correctly, this means: is there an instance of every theorem preceding every other theorem in the expert training data?

Thanks for your question. Theorem 1 proves 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  $a_0 \xrightarrow{T_1} \dots \xrightarrow{T_n} a_n$ . Then according to the intervention in this paper (given by Eq. (1) and Eq. (2)), the expert data needs to contain a common antecedent node  $u_{0,1}$  from the reasoning steps  $u_{0,1} \rightarrow a_0$  and  $u_{0,1} \rightarrow a_1$  in two specific solutions. Similarly, other common antecedent nodes  $u_{1,2}, u_{2,3}, \dots$  need to be also included, which gives each transition a positive probability. Since the  $u_{0,1}, u_{1,2}, u_{2,3}, \dots$  may not be the same,  $a_0$  is not necessarily the ancestor of  $a_n$ . Please note that the expert data is constantly expanded by adding solutions that are already validated. So, the evolutionary patterns are also enlarging in the problem solving.

Admittedly, such a condition seems much strict in practice. But our discussion here aims to strictly guarantee the global convergence in theory. In application, an arbitrarily set probability can be also introduced to explore different reasoning patterns out of the expert data.

- I did not understand the sentence including the words "we randomly select 100 successive problems from the test dataset" (p.8). Do you not evaluate on all test problems for each method? If you evaluate only on 100 problems, why are results in Table 1 not whole numbers (in %)?

Thanks for your question. We randomly select 100 successive problems to test the accuracy of our method, 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. 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). Thus, it may get an accuracy that is not a whole number.

- The results in Table 2 and 3 do not make sense. According to Table 2, the average time is similar between all methods. However, the problem solving time for a test sample is an order of magnitude smaller for baselines, and 800x larger for GAN+CER than for GAN alone! Where is my misunderstanding?
  - What is the "pseudo-optimal solution generation for training data" in Table 3? It takes 20 hours for 2101 training problems, or about 34 seconds per problem. This is actually more efficient than the average time for any GAN-based method.

Thanks for your question. GAN+CER takes much longer time than GAN alone because the CER step includes multiple evolutions. It involves an iterative searching and optimization for the solutions, and thus is quite time consuming. The raw training data in Geometry3K only give problems and final results, without reasoning sequences. "pseudo-optimal solution generation for training data" means to generate some pseudo-optimal reasoning sequences that can be used to train the network. This is a prerequisite step for both Inter-GPS and GAN.

#### **Reviewer 4Jop:**

It's interesting to see the idea tried out on geometry math problems. And I really appreciate the engineering efforts made in this work. However, I do not think the technical contribution is worth publishing.

- There is literally no novelty in this work. Training a GAN is not anything new for generative learning these days. And CER itself is neither intuitive nor effective search. It's simply trying out different combinations

---

without any guidance (hence the long problem solving time). In addition, the GAN design details are extremely confusing. To start, why considering training convolutional GAN rather than other generative models for a sequence-sequence problem? There are certainly more effective methods like tuning pretrained language models, or simply by prompting foundation models. Besides, why appending a "randomly generated sequence as input" for the GAN generator? How is the sequence generated? What do you need this randomly generated sequence for? For the discriminator, what is the point of concatenating a fake or real solution? If you do concatenation, then what is the point of this entire generative adversarial process? You can simply classify based on what you concatenate.

Thanks for your comments. The work proposes a CER method to generate problem solutions. While may be low efficient, it is effective for the solution optimization. The computation efficiency depends on the scope it explores. In this paper, we probabilistically exploit the theorem dependence in expert data (illustrated as the intervention mechanism) as a guidance of the searching. If the exploitation of expert data is more deterministically, the searching time will be reduced, but we are not able to achieve the current accuracy performance.

The goal of initialization is to generate a set of heterogenous solutions using an efficient method. We need to seek a balance between the random sampling and the exploitation of expert data. Admittedly, there are several other sequence-to-sequence models. But after a simple test, we find GAN is relatively suitable, keeping the initial solutions with certain heterogeneity around the expert data via a relatively efficient training. Specifically, popular pretrained language models (or foundation models) have encoded general words/sentences in a latent space. Simply tuning of them is not applicable for the theorem proving field, as dependences between theorems in a problem solution are quite different from those of language words/sentences. To use foundation models, we may thoroughly re-train the model using expert solutions rather than tuning existing ones. But such a process is quite inefficient. Therefore, we choose GAN to complete this task.

As explained above, 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. By sending it together with the problem representation (via concatenation) to the pre-trained generator, the solution "reduces" its heterogeneity and is constrained "near" the expert dataset. In this way, the "filtered" initial solutions will both retain a certain degree of heterogeneity and include some expert heuristics, providing the subsequent counterfactual evolution a suitable start point.

Though we use the generator network alone at last, its training relies on the discriminator to get an optimizing direction. According to the classic training of discriminator in conditional GAN, the problem representation is the condition part and the fake or real solution is to be checked. We concatenate them to be an input of discriminator since the solution mostly corresponds to a particular problem. For a given input *prob + sol* (as in Figure 3), if the *sol* comes from the generator, the label to discriminator should be 0, indicating that the solution is fake. If the *sol* comes from the expert data, the label should be 1, indicating that the solution is real.

- Confusing presentation. For one thing, the presentation of the paper is very unprofessional. Figure 1 is just a low-res crop from Inter-GPS. Similar low-res crops for other figures. There is no formal description of the problem or mathematical formulation for the solution process either. Very counterintuitively, the authors draw position encoding and pooling for attention modules. I'm also at a loss what Equation 2 is trying to say: the authors use unconventional notations to denote conditioning and Bayes rules, and if I understand conditioning correctly, the derivation is wrong: it should be  $P(u) / P(b)^2$ . I'd also suggest not to

use terms like counterfactual or intervention, they carry special meaning in causality and simply do not fit here. Your method is just perturbing the generated sequence.

Thanks for your comments. The formal representation of a geometric problem, involving the text literals and the diagram literals, are shown in Figure 1. A simple solution representation, denoted as a theorem sequence, is also given in the second paragraph of Section 3.1. Due to the page limit, we provide an explicit example in the supplemental file to clearly show how the problem and solution are represented and how to solve the problem. For the low-res problem, we have re-draw Figure 1. Hope it is much better now. The position encoding in Figure 3 is set to keep the position information. It is like an attention mechanism. The pooling has a property of transitional invariance. It can retain the dependence of adjacent theorems.

For Equation 2, thanks for pointing out the issue. The deduction does have some minor problems, with an additional square superscript as a typo. The result, however, is correct, and the whole process is as follows:

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

The first equal sign is the probability chain rule. The second equal sign is the independence. And the third equal sign is Bayes theorem. We have corrected the typo in the revision.

For the usage of counterfactual and intervention terms, we would like to give a response as follows. In the classic theory of causality, counterfactual reasoning needs to estimate the exogenous variables in Structure Causal Models via the fact already happened, and then arbitrarily set the variable to be intervened to observe the final model output. In our geometric problem solving, the SCM with all the exogenous variables is not known. Therefore, the intervention is directly conducted. The difference here is that the intervention is stochastic according to a probability distribution, which can be viewed as a “soft” intervention. (Reference: J. Peters, D. Janzing and B. Schölkopf. Elements of Causal Inference: Foundations and Learning Algorithms. The MIT Press, 2017.)

- The experimental results are insufficient. The method uses GT textual format and yet only compares with previous work without GT. This is unfair. If you take into the performance of Inter-GPS with GT, there is a huge performance gap. Besides, the method is much much slower. I would also recommend the authors to check whether pure symbolic methods could solve the problem as everything is now in the form of logics.

Thanks for your comments and suggestions. In the original Inter-GPS research, using the Ground Truth (GT) literals or not only impacts the problem parser. By contrast, our study focuses on the problem solving, rather than the parsing process. Therefore, we still use the same parser without GT from the original work in our experiments, and the comparative result does make sense. Obviously, our method is much slower. But this is the cost of the accuracy improvement. 20 minutes (in average) of extra running time for a 4.4% improvement overall seems acceptable. After all, there is no free lunch.

For pure symbolic methods, they also need to decide which theorems and in what order to be applied. This essentially returns to our focus in this paper---searching and determining a suitable theorem sequence as a solution.

## Supplementary File

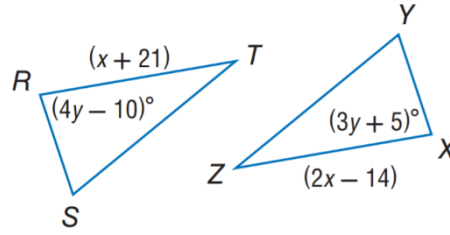


Figure S1 An Example from Geometry-3K.

To clearly show our geometric problem solving, we provide an explicit example to illustrate our method. Take the first problem in geometry-3k as an example. Its textual description is “Triangle R S T is congruent with Triangle X Y Z. Find y”. Its diagram is drawn in Figure S2. The choices are: [“5”, “14”, “15”, “35”]. After the parsing process, the problem is formally represented as:

```
{
  Congruent(Triangle(R,S,T),Triangle(X,Y,Z)),
  Equals (LengthOf (Line(T, R)), x+21),
  Equals(MeasureOf(Angle(T, R, S)), 4y-10),
  Equals(MeasureOf(Angle(Z, X, Y)), 3y+5),
  Equals(LengthOf(Line(Z, X)), 2x-14),
  Find(y)
}.
```

The formal representations are encoded into a 10-dimensional embedded vector (as shown in Figure 2 in our paper), which is called the problem latent representation. Here, the achieved embedded vector is <14, 84, 85, 92, 106, 114, 122, 150, 201, 222>.

Then, the problem latent representation is concatenated with a randomly generated 30-dimensional sequence as the input to the generator, and a 30-dimensional fake sequence is output (as shown in Figure 2 in our paper). Here, the problem latent representation plays as the condition part of input and the randomly generated sequence plays as the solution part.

After the transformation of the generator network, we get a fake sequence output. And it is concatenated with the problem latent representation (the 10-dimensional embedded vector mentioned before) to be the input of discriminator. The output of discriminator is a real value between 0 and 1, say 0.0108 here. This means that the discriminator tends to view the solution to be fake, that is, from the generator. We further concatenate the problem latent representation with its expert solution to be the input of discriminator. This time, the output will be a new real value, say 0.0424, which means the discriminator still treats the solution to be fake (this is certainly wrong). By giving the label 0 for solutions from the generator and 1 for expert solutions, the discriminator and generator can be updated using the classic GAN training method. As the training goes on, the output of a fake solution in discriminator gradually gets closer to 0.5.

Now suppose the GAN network has been successfully trained, as described in the last paragraph. We get a 30-dimensional solution output from the generator network, say <5, 71, ..., 314, 405>. Each element of this

---

255 solution sequence gives a theorem index that can be potentially applied for the problem solving. Then, we conduct  
an intervention and evolution according to Algorithm 1 in our paper. Specifically, the procedure involves a  
random selection of intervention point, a replacement of new theorem candidate (as shown in Figure 4, Equation  
(1) and Equation (2) in our paper), and a selection of optimized solution (according to Equation (3) in our paper).  
Still take first problem in geometry-3k as discussed before. The solution from generator is  
<2, 7, 5, 2, 1,17>

260 Note that we only give effective reasoning part. Other elements in the 30-dimensional vector are all set to be 0.  
After an intervention at the third element, the new solution is  
<2, 7, 10, 2, 1,17>

265 Each theorem indexed by the new solution sequence is checked whether its premise matches the formal problem  
representation together with intermediate conclusions. For instance, if the premise of Theorem 2 (the first  
element), indirect triangle sum theorem, is  
“if angles in triangle not has number”

It matches the problem. Then, it is called applicable and its conclusion is added into intermediate conclusions.  
After iteratively checking each theorem, the solution is scored according to Equation (3) in our paper. And we  
select the top solution as the final output of our problem solver. For the example problem, the final solution is

270 <2, 1, 10, 17, 1>

This means by using the second, the first, the tenth, the seventeenth and the first theorem in order, we will get the  
final answer 15 (choice C) for the problem.