An Ensemble Approach for Automated Theorem Proving Based on Efficient Name Invariant Graph Neural Representations

Achille Fokoue, Ibrahim Abdelaziz, Maxwell Crouse, Shajith Ikbal, Akihiro Kishimoto, Guilherme Lima, Ndivhuwo Makondo, Radu Marinescu
IBM Research
achille@us.ibm.com, {ibrahim.abdelaziz1, maxwell.crouse, akihiro.kishimoto, guilherme.lima, ndivhuwo.akondo}@ibm.com, shajmoha@in.ibm.com, radu.marinescu@ie.ibm.com

Abstract

Using reinforcement learning for automated theorem proving has recently received much attention. Current approaches use representations of logical statements that often rely on the names used in these statements and, as a result, the models are generally not transferable from one domain to another. The size of these representations and whether to include the whole theory or part of it are other important decisions that affect the performance of these approaches as well as their runtime efficiency. In this paper, we present NIAGRA; an ensemble Name InvAriant Graph RepresentAtion. NIAGRA addresses this problem by using 1) improved Graph Neural Networks for learning name-invariant formula representations that is tailored for their unique characteristics and 2) an efficient ensemble approach for automated theorem proving. Our experimental evaluation shows state-of-the-art performance on multiple datasets from different domains with improvements up to 10% compared to the best learning-based approaches. Furthermore, transfer learning experiments show that our approach significantly outperforms other learning-based approaches by up to 28%.

1 Introduction

At its core, automated theorem proving is a complex search problem, wherein the construction of a proof is treated as a search through sound inference steps until a satisfactory conclusion is derived. While proof search is a generally intractable problem, significant amounts of research devoted towards the design and implementation of domain-specific search heuristics have allowed automated theorem provers (ATPs) to function effectively in application areas spanning from distributed systems [Hawblitzel *et al.*, 2015; Garland and Lynch, 2000] to medicine [Masci *et al.*, 2014; Lucas, 1993]. However, as the set of domains that ATPs have been applied to has grown, the need to move away from hand-crafted search heuristics has been recognized [Schulz, 2017].

Recently, there has been strong interest in applying deep learning to proof search [Paliwal *et al.*, 2020; Bansal *et al.*, 2019; Loos *et al.*, 2017; Crouse *et al.*, 2021], with the ultimate goal being methods capable of guiding proof search that can automatically adjust themselves to a particular domain with little-to-no manual feature engineering. The earliest deep learning-based approaches [Loos *et al.*, 2017] to proof guidance operated over shallow representations of their inputs, where logical formulas were treated as bags of symbols. While able to achieve impressive results at the time, these methods fell short against later approaches designed to capture the rich graph structure inherent to logical formulas.

With the advent of geometric deep learning [Bronstein *et al.*, 2017], methods that could explicitly account for the structural characteristics of logical formulas quickly began to gain traction. Early usages of graph neural networks (GNNs) for theorem proving often focused on offline tasks like premise selection [Wang *et al.*, 2017; Crouse *et al.*, 2019; Rawson and Reger, EasyChair 2020], as the added computational expense of graph neural networks outweighed any value they could provide when used during proof search. It has only been quite recently that works incorporating graph neural networks directly into purely neural proof guidance [Paliwal *et al.*, 2020; Abdelaziz *et al.*, 2022; Aygün *et al.*, 2022] have been able to demonstrate state-of-the-art results.

Most state-of-the-art deep learning-based proof guidance systems leverage graph representations of logical formulas in roughly the same way. These systems first convert logical formulas into directed acyclic graphs (DAGs). They then apply any of the many GNN variants to each formula's DAG representation to produce a dense, real-valued vector that can be processed by other neural components. These approaches convert each logical formula into a separate graph that is disconnected from the graphs representing other formulas in the same theory (an example of this is shown in Figure 1). When these isolated graphs are processed by GNNs, they are thus processed independently of one another. This simple approach is very efficient in terms of run-time performance and computational overhead for two main reasons: (1) it enables the parallel computation of each formula's graph embedding, and (2) it enables the reuse of those formula embeddings within a proof attempt, since a formula's graph will remain unchanged throughout the reasoning process. Though attractive from an efficiency perspective, in general, this approach is not faithful to the semantics of logical formulas. This is because the meaning of a formula in a particular proof attempt

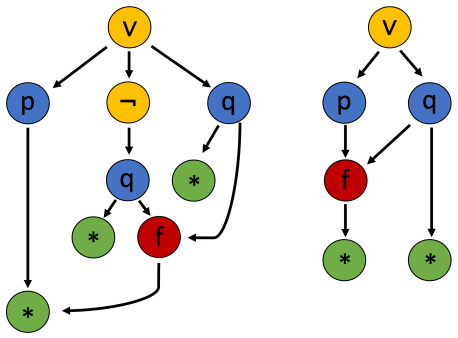


Figure 1: DAG representations for two clauses, $\forall A, B, C.\ p(A) \vee \neg q(B, f(A)) \vee q(C, f(A))$ and $\forall X, Y.\ q(f(X), Y) \vee p(f(X))$, with anonymized variables to ensure renaming invariance.

is a function of all other formulas available to the reasoner.

In order to preserve the inter-dependencies between formulas, it would be more appropriate to represent the set of all formulas with a single, connected DAG. In practice, this is too expensive (in terms of both memory and computation time) to execute within modern, highly optimized theorem provers. Prior work [Jakubuv *et al.*, 2020] exploring more holistic graph embeddings found a middle ground by batch processing subsets of formulas together. That is, their approach would aggregate unprocessed clauses until a size threshold was reached and then embed the aggregated set of clauses together with a GNN. While effective, their approach (1) introduces a delay between when a clause was inferred and when it was embedded, (2) requires frequent large-scale graph embedding operations, and (3) computes only a single, fixed relevance score for each clause.

In this work, we introduce NIAGRA (Name InvAriant Graph RepresentAtion). NIAGRA aims to achieve a representational fidelity that surpasses previous works while simultaneously avoiding the significant computational expense involved in repeatedly computing graph embeddings of the evolving large connected graph representing the state of the prover (i.e., the graph containing the representations of the original axioms and the derived formulas). NIAGRA achieves this through a novel embedding process that, prior to proof search, uses the initially provided theory to generate embeddings for individual non-logical symbols (i.e., constant, function and predicate names), that are representative of the complete structural context in which the symbols occur. Then, during theorem proving the embeddings for formulas are computed independently (as with prior approaches), but using the theory-level symbol embeddings as the graph's initial node embeddings.

In addition to our main contribution of a novel graph embedding process, we also consider and evaluate alternative ensembling schemes for neural-guided theorem proving. In particular, we show how standard ensembling methods can be improved by taking into account not only the weights of the neural architecture, but also the configuration of the underlying theorem prover that executes inference steps.

Through extensive experimentation, we find that this approach yields state-of-the-art performance as compared to other unsupervised deep learning-based and standard theorem provers on standard benchmark datasets. Notably, on the hard subset of Mizar [Grabowski *et al.*, 2010] (MPTP), it significantly outperforms the best unsupervised learning-based system by 10% (HER: [Aygün *et al.*, 2022]) and the best classical ATPs by 3% (Vampire) and 18% (E), and transfer learning experiments show that our approach significantly outperforms other unsupervised learning-based approaches by up to 28%. In summary, our contributions are:

- A name invariant graph representation for logical formulas that attempts to balance between being faithful to the semantics of these formulas while providing minimal computation overhead.
- An ensemble approach for ATPs that leverages different configurations of the underlying un-optimized reasoners to gather more diverse high quality training data.
- Experimental evaluation which shows that in comparison to the best unsupervised learning-based reasoners, NIAGRA overall solves more problems, finds most proofs in fewer steps, and has a significantly better transferability especially on hard datasets with the most diverse domains and vocabulary.

NIAGRA's code, data, and trained models are publicly available at https://github.com/ibm/TRAIL.

2 Background

This work focuses on *saturation-based* theorem proving [Robinson, 1965] in first-order logic (FOL) with equality. In this setting, the automated theorem prover (ATP) is given a *conjecture* (i.e., a formula to be proven true or false) and a logical theory (i.e., an initial set of *axioms* assumed to be true). In addition, the ATP is equipped with a set of *inference rules*, i.e., rules that can be applied to true formulas to yield new true formulas. To execute proof search, the prover proceeds by refutation, i.e., it attempts to show that the negated conjecture together with the axioms entails a contradiction.

In refutation-based theorem proving, the prover adjoins the negated conjecture to the initial set of axioms and applies the inference rules repeatedly to this set until either (1) a contradiction is found, which means that the conjecture is true; or (2) a saturated set is obtained, i.e., a set from which no new formula can be derived (which means the conjecture is false); or (3) its time limit is reached, in which case no conclusion can be derived regarding the validity of the conjecture. A *proof* is a sequence of *inferences* (i.e., inference rule applications) leading from the initial set of formulas to either a contradiction or a saturated set.

The logical theories considered in this work will be theories expressed in the *conjunctive normal form* (CNF). This is the most common representation utilized by state-of-the-art saturation-based theorem provers. A CNF theory consists of a conjunction of clauses, where each clause is a disjunction of literals. The literals within a clause are (possibly negated) atomic formulas, where all variables occurring within the formula are implicitly universally quantified.

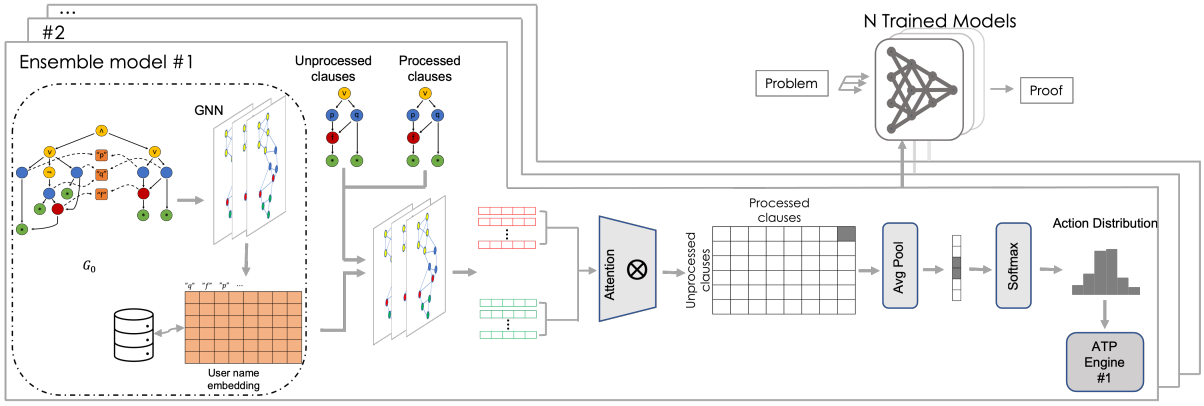


Figure 2: NIAGRA Overview: N ensembles models are trained, each with their own GNN embedder and policy network. An initial set of node embeddings are computed for the entire input theory, defining a set of contextualized embeddings for user-defined nodes which are later used as the initial embeddings for nodes in individual clause graphs. At inference time, NIAGRA attempts to solve the input problem via the various learned models and output the final proof when found.

3 NIAGRA

We now describe NIAGRA, our approach that improves neural proof guidance systems for saturation-based ATPs, e.g., E [Schulz, 2002], Vampire [Kovács and Voronkov, 2013], etc. Specifically, we present (a) a novel, efficient method for generating name invariant graph neural representations of clauses that provides more meaningful embeddings of non-logical symbols like predicates, functions, and constants and (b) an ensemble method that learns different proof guidance models for different configurations of the underlying ATP.

As our contributions focus on graph representations and ensemble methods (and not on designing a general neural proof guidance system), we adopt the existing TRAIL [Abdelaziz *et al.*, 2022] architecture for proof guidance. We emphasize, however, that the method we introduce is not TRAIL-specific. In principle, as it involves replacing the initial embeddings of symbols that serve as the lowest layer of each formula's graph embedding, it is complementary to other high performing unsupervised learning-based proof guidance systems like HER [Aygün *et al.*, 2022].

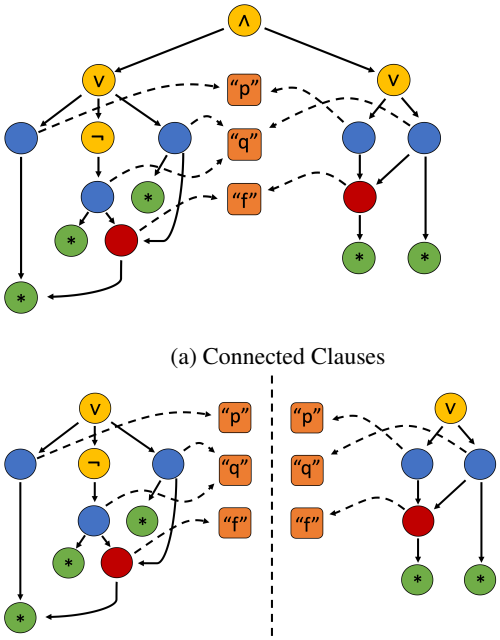
As the details of proof guidance are provided in [Abdelaziz *et al.*, 2022], we only lightly cover them again here. Figure 2 shows the entire proof guidance architecture, which is a deep reinforcement learning-based system that employs an attention-based policy network. The state of the theorem prover is represented in terms of the set of processed and unprocessed clauses. The policy network decides which inference to explore based on attention scores computed for pairs of clauses (one from the processed set and one from the unprocessed set). During learning, a temperature τ parameter controls the trade-off between exploration and exploitation, decaying throughout iterations to promote more exploitation over time. Lastly, the reward is computed as the ratio of time taken to solve the problem by the underlying ATP and the time NIAGRA took to solve the same problem, thus giving the system a higher reward if it found proofs faster than the underlying reasoner. We next describe NIAGRA's design and provide the details on our proposed ensembling technique.

3.1 The Challenge of an Efficient Name Invariant Graph Representation

Graph-based neural formula representations underlie the most successful deep learning-based proof guidance systems [Abdelaziz *et al.*, 2022; Aygün *et al.*, 2022]. In these systems, formulas are converted into DAGs (see Figure 1) by collapsing identical subtrees of a formula's abstract syntax tree into single nodes with multiple parents. Generally, DAGs are constructed for each individual clause in the provided problem and, during inference, embedded independently of one another with a GNN.

Clause embeddings computed independently can recover some of the dependencies between clauses by using the user-defined names of predicates, functions, and constants to select initial node embeddings; however, such schemes are ultimately limited in that they are leverage name-sensitive representations. Name-sensitive representations can become problematic when testing on a dataset with a completely different vocabulary or, perhaps worse, with a vocabulary that overlaps with the training vocabulary but in which the common names have an entirely different meaning (e.g., functions f and g in the training dataset have been swapped in the test dataset). In principle, predicate, function and constant names are irrelevant artifacts, since the meaning of a given logical theory is unchanged after consistently renaming predicates, functions, and constants. Thus, unnecessarily including specific names in clause representations could result in learning of spurious correlations between those irrelevant artifacts and hinder generalization.

To avoid name-sensitive representations, we take inspiration from recent works that have explored preserving properties such as invariance to predicate, constant, and function names, and variable quantification order [Rawson and Reger, 2020; Olšák *et al.*, 2019; Paliwal *et al.*, 2020]. The system most similar to ours is TRAIL [Abdelaziz *et al.*, 2022; Crouse *et al.*, 2021], which does not rely on user-defined predicate, function, and constant names. Rather, it produces independent graph embedding using a representation similar



(a) Connected Clauses

(b) Disconnected Clauses

Figure 3: Example representation of clauses from Fig 1 with (a) and without (b) user-defined names sharing. Information about user-defined names appears only in the square nodes.

to Figure 3b. In its GNN computation, the initial embedding of a node is only a function of its type (i.e, in the figure this is the color of the node and its shape, but not its label). Unfortunately, this solution has two shortcomings. First, the initial embedding does not preserve the individuality of each symbol (e.g., all square name nodes have the same initial embedding). Second, there is very limited information exchange between disconnected clause graphs. Within a single clause, all applications of the same function or predicate (e.g., connections to the "q" node in Figure 3b) have a direct link to the same user-defined name node (e.g., the square node labeled "q"), however, these square user-defined name nodes are not connected or shared across clause graphs (as shown in Figure 3b), meaning that the only information they share stems from how they are initialized.

3.2 Name Invariant Graph Representations

Our proposed solution is based on the following three key observations:

1. In conjunctive normal form, after an equivalence-preserving variable renaming, the graph connecting all clauses is such that the only shared nodes across clauses are nodes representing the names of constants, functions, and predicates (the square nodes shown in Figure 3a). To see why, first note that the universal quantifier distributes over conjunctions. Thus, in the clausal CNF, we can push the top level universal quantifiers inside each clause (i.e., $[\forall X_1, .., X_n : cl_1 \wedge ... \wedge cl_m] \equiv [(\forall X_1, .., X_n : cl_1) \wedge ... \wedge (\forall X_1, .., X_n : cl_m)]$) and then safely rename variables within each clause in such a way that two distinct clauses do not have any common variables. The only shared symbols between clauses are then constants, functions and predicate names (including skolem constants and functions introduced during the normalization process to remove existentially quantified variables).
2. All user-defined and skolem names of constants, functions, and predicates are already present in the original normalized theory (with the exception of a select few predicates generated by an infrequently applied E inference strategy).
3. The meaning of all such constant, function and predicate names is fully specified by the set of initial axioms and the conjecture.

Based on those three observations, our proposed solution employs the following two-part process:

1. First, before reasoning begins, compute from the fully connected graph of the normalized theory and the negated conjecture (e.g., Figure 3a), the embeddings of the vocabulary of predicates, functions, and constants (i.e., square nodes in the figure). This provides meaningful embeddings of non-logical symbols in a way that is name invariant and dependent only on the input theory and the conjecture to prove.
2. Second, at each step of reasoning, use the embeddings computed previously as the initial embeddings of named nodes (i.e., square nodes) in the independent graph (e.g., in Figure 3b) of each clause so that, across all independent graphs, all name nodes with different labels receive different initial embeddings while those with the same label start with the same initial embeddings. In doing so, we address the two key issues of lack of name node individuality and limited information sharing across independent clause graphs.

This procedure has the advantage that it is fairly efficient to compute (requiring only one theory-level embedding computation) while also taking into account much richer structural information than previous embedding methods that isolated each formula. In our implementation, the same GNN model is used both when computing node embeddings for the connected graph of the whole theory and conjecture (see Figure 3a) as well as when computing the embeddings for isolated clause graphs such as those in Figure 3b.

3.3 Ensemble Method for ATPs

Ensemble methods are a standard technique used to improve learning performance by combining predictions from multiple models. In the context of learning-based proof guidance systems for ATPs, HER [Aygün *et al.*, 2022] uses a simple approach that relies on 10 different randomly initialized models to increase the diversity of the search. However, these different models learn from examples collected from proof attempts using the same configuration of the underlying ATP. Thus, the major source of diversity comes from the random initialization process.

We first observe that mature ATPs such as E [Schulz, 2002] have many configuration options that are orthogonal to the

Domain	#Problems
Set Theory	247
Geometry	207
Kleene Algebra	114
Commonsense Reasoning	88
Logic Calculi	86
Semantic Web	79
Relation Algebra	48
Processes	45
Knowledge Representation	34
Group Theory	25

Table 1: Top 10 problems domains in TPTP dataset.

clause selection process that learning-based proof guidance systems such as HER [Aygün *et al.*, 2022] or TRAIL [Abdelaziz *et al.*, 2022] aim to replace. For example, E provides options to select different term order strategies, different literal selection strategies, etc. Those different configurations of the underlying ATP system can provide a greater source of diverse proofs than relying only on the random initialization of multiple models.

Given a time limit T for each proof attempt, our ensemble approach for ATPs consists of three important steps:

- First, we create N different configurations of the underlying ATP to guide (each with their proof guidance or clause selection strategy *disabled*).
- Second, we train, in parallel and independently, each of the N differently configured ATPs with a modified time limit of T/N using, for example, the reinforcement learning based approach of TRAIL or the Hindsight Experience Replay of HER. In each of the N independent branches, at the start of the first iteration, a randomly initialized model is used. In each branch, at each iteration, attempts to solve all the problems in a dataset are made with the time limit of T/N and the resulting examples are collected to update the model of the branch. The updated model is then used at the next iteration.
- Finally, at the end of each iteration, the set of problems solved is simply the union of problems solved in each of the N independent and parallel branches.

Note that we give equal importance to the N different configurations by dividing the time limit equally among them (T/N). This can indeed be extended to give different time limits for different configurations to weight their importance. However, we leave this for our future work.

4 Experimental Evaluation

4.1 Datasets and Baselines

We evaluate NIAGRA on three benchmarks; **MPTP**, **M2K** and **TPTP**. M2k and MPTP are standard benchmarks that have been used for evaluation by many learning-based systems [Aygün *et al.*, 2022; Abdelaziz *et al.*, 2022; Crouse *et al.*, 2021; Zombori *et al.*, 2020]. Both benchmarks are parts of the larger Mizar [Grabowski *et al.*, 2010] dataset, with M2K and Mizar consisting of 2,003 and 2,078 Mizar problems, respectively. The main difference between both benchmarks stems from how their constituent problems were selected. M2K problems were selected randomly from the subset of Mizar that is known to be provable by existing ATPs while MPTP problems were selected regardless of whether or not they could be solved by an ATP system. Since problems in both benchmarks are from a single domain (mathematics), we also used TPTP dataset (Thousands of Problems for Theorem Provers)[1]. TPTP is the definitive benchmarking library for theorem provers, designed to test ATP performance across a wide range of problem domains (e.g., biology, geography, number theory, etc.). Using TPTP allows us to better test how NIAGRA generalizes to a broader set of domains than would be possible by evaluating with solely Mizar problems. Table 1 shows the domains comprising the subset of TPTP.

We compare NIAGRA against state-of-the-art traditional and learning-based reasoners. In particular, we compare against the following: (1) **E** [Schulz, 2002]: state-of-the-art theorem prover that relies on manually designed proof guidance heuristics. In our experiments, we used E 2.6 Floral Guranse[2] which is the latest available version of E (as of August 2022). We ran E prover in *auto-schedule* mode, which is known to solve the most number of problems. (2) **Vampire** [Kovács and Voronkov, 2013]: another popular theorem prover and the world champion in ATP. We used Vampire version 4.7[3]; the latest version of Vampire when this paper was submitted. We also ran Vampire in best mode, *CASC*, which has competition specific presets for schedule, etc. (3) **HER** [Aygün *et al.*, 2022]: the most recent learning based approach that uses hindsight experience replay in an incremental learning setting. (4) **TRAIL** [Abdelaziz *et al.*, 2022] is another recent reinforcement learning based theorem prover that relies on GNNs for logical formula representation. (5) **rlCop** [Kaliszyk *et al.*, 2018] and (6) **plCop** [Zombori *et al.*, 2020] are two other learning approaches based on reinforcement learning that leverage connection tableau-based theorem proving techniques.

NIAGRA was implemented using E as the underlying reasoner after turning off E's proof guidance/clause selection. We set the maximum time limit for solving a problem for all systems to *100 seconds*. We ran E and Vampire ourselves on the same hardware setup used for NIAGRA and used the numbers reported in TRAIL and HER papers (both systems used a 100-second time limit as well). Note that the numbers we obtained for E prover are comparable to those reported by HER and better than those reported by TRAIL. Detailed information about software & hardware used is provided in supplemental materials.

4.2 Number of Problems Solved

In this experiment, we compare the number of problems solved by NIAGRA to other baselines within the allowed time limit. Table 2 shows the results for each system (including two variants of NIAGRA where the model update

[1]http://tptp.cs.miami.edu/

[2]https://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html

[3]https://github.com/vprover/vampire/

			Traditional		Unsupervised Learning Based						
Dataset	Domain	#Problems	E (auto-sched)	Vampire (casc)	rlCop	plCop	TRAIL	HER	NIAGRA (10 epochs)	NIAGRA (1 epoch)	Stat. Sig. (z-test)
MPTP	Math	2,078	63.7	73.0	31.6	42.8	58.4	68.5	72.7	**75.5**	✓($z = 1.84$)
M2K	Math	2,003	97.0	**99.0**	67.4	70.7	90.2	94.6	96.2	98.0	✓($z = -2.65$)
TPTP	Multiple	1,955	81.5	**84.6**	-	-	-	-	-	82.1	✓($z = -2.15$)

Table 2: Percentage of problems solved across various datasets for multiple traditional and learning-based reasoners. Stat. Sig. indicates whether the improvement over the next best system is statistically significant ('checkmark' for p-value < 0.05).

step after each iteration is done in either 1 or 10 epochs, respectively). The state-of-the-art traditional reasoners, E and Vampire, were able to solve 63.7%(1,324) and 73.0% (1,517) problems on the harder MPTP dataset, respectively. In comparison, learning-based approaches managed to provide competitive performance, with HER and NIAGRA solving 5% and 12% more problems compared to E (a relative improvement of 7% and 18%), respectively. The use of our ensemble method and the name-invariant representation of clauses allowed NIAGRA to outperform HER by 10% (1,569 vs. 1,424). The same behavior can be observed on the M2K dataset with NIAGRA being the only learning-based system that outperforms E, and only 20 problems away from Vampire. Similarly, on TPTP, NIAGRA slightly outperformed E by proving 12 more conjectures. This is somewhat unsurprising, as TPTP is a particularly hard dataset due to its varied domains that each contain smaller numbers of problems (thus making it a valuable benchmark to test on). Although NIAGRA slightly underperforms compared to Vampire in TPTP and M2k, it provides a very competitive performance which allowed it to outperform E across all datasets and Vampire on the hard Mizar dataset. We believe this is a very promising performance for a system that does not rely on any heuristics and is learning completely from scratch.

We also show in Figure 4 the progress of NIAGRA in terms of the percentage of problems solved across iterations. In all three datasets, the majority of the improvement happens in the first ten iterations, afterwards the model keeps improving but slowly. Compared to MPTP and TPTP, M2K model initial performance (iteration 0) is much higher (68.8) due to the relative simplicity of the dataset.

Finally, although we use the same 100 secs time limit as other tabula rasa approaches (HER, TRAIL & mlCOP), with our ensemble approach, we divide the overall time budget among 4 models, i.e., 25 secs per model. Only a small fraction of that time limit is used to solve the vast majority of problems. For example, on MPTP, all 4 models solved individually, in 25 secs, more problems than E with 100 secs: 1,343, 1,388, 1,409, and 1,361 compared to 1,324 by E. Furthermore, by the end of 20 iterations (for models trained with 10 epochs), most problems are solved in few seconds. On MPTP, one of the models used an average of 1 sec per problem solved (stddev: 1.6). Thus, the overhead introduced by our approach remains small.

4.3 Transfer Learning

Table 3 shows NIAGRA's performance when trained on one dataset and tested on another. Note that, while MPTP and M2K are both from the same mathematics domain, the TPTP problems are drawn from a different, varied set of domains (see Table 1). As expected, the number of problems solved is maximized when training and testing on the same domain. However, it can be seen that transfer learning is happening, e.g., the M2k-trained model was able to solve 53.6% TPTP problems in testing mode. Similarly, the TPTP-trained model showed promising performance when tested on MPTP and M2k datasets where it solved 48.7% (1,013) and 86.6%(1,736) of the problems, respectively. Importantly, Table 3 includes results for NIAGRA without the name invariant graph embeddings (i.e., only the ensemble scheme remained) in the second row. The results clearly show that NIAGRA with our efficient name invariant graph embeddings (first row) is superior to NIAGRA without it (second row). In addition, as compared to TRAIL (the only prior system reporting transfer learning results between MPTP and M2K), our method achieves significantly improved performance.

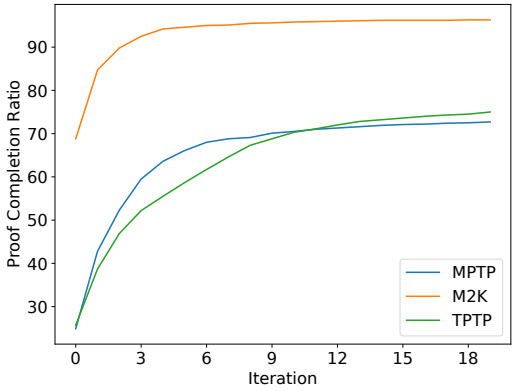


Figure 4: Percentage of problems solved across iterations

4.4 Ablation Experiments

Effect of Faithful Graph Representation

In this experiment, we show the effect of switching on and off the initial graph embedding. Our hypothesis is that having the full theory graph where user-defined names are shared allows NIAGRA to learn meaningful embeddings of non-logical symbols in a way that is both name invariant and dependent on the input theory. Table 4 shows the performance of NIAGRA with and without (ensemble only) the connected graph of the whole theory. It can be seen that the initial embeddings obtained from the normalized theory graph helped NIAGRA to achieve significantly better performance (relative

	TPTP → MPTP	TPTP → M2k	MPTP → M2k	MPTP → TPTP	M2k → MPTP	M2k → TPTP
NIAGRA	**48.7**	**86.6**	**93.5**	**50.8**	**64.5**	**53.6**
NIAGRA (Ensemble Only)	39.1	82.0	90.4	47.2	59.1	50.6
TRAIL	-	-	83.5	-	50.4	-
Stat. Sig.(z-test)	✓($z = 6.25$)	✓($z = 4$)	✓($z = 3.61$)	✓($z = 2.25$)	✓($z = 3.58$)	✓($z = 1.88$)

Table 3: Transfer learning: training on one dataset and testing on another. The numbers are the *percentages* of solved problems. "Ensemble Only" refers to NIAGRA operating *without* the theory-level graph embeddings.

Dataset	NIAGRA (Ensemble Only)	NIAGRA	Stat. Sig. (z-test)
MPTP	67.6	**72.7**	✓($z = 3.59$)
M2K	94.6	**96.2**	✓($z = 2.43$)
TPTP	66.3	**78.9**	✓($z = 7.79$)

Table 4: Effect of faithful graph representation: NIAGRA's performance (percentage of problems solved) with and without the initial whole theory graph.

improvement) on the two hardest datasets: MPTP (+7.5%) and TPTP (+17%). As expected, our efficient name invariant scheme has the most significant impact (17% improvement) on the TPTP dataset, which uses the most diverse vocabulary and domains.

Ensemble Size

We also evaluated our ensemble against both non-ensemble methods and methods employing a standard ensemble setting (e.g. in HER) where multiple learners are used which only differ in their initial random model weights. Our results show that our ensemble approach performs much better overall. Due to lack of space, we add the details of this experiment to the supplementary material.

5 Related Work

Reinforcement learning (RL) has been used to learn for proof-guidance heuristics from scratch. For example, [Kaliszyk *et al.*, 2018; Zombori *et al.*, 2020] combine RL with Monte Carlo tree search to guide FOL tableau-based ATPs. Bansal *et al.* (2019) refer to deep RL-based interactive theorem proving in HOL Light and use imitation learning with an application to higher-order logic. Aygün *et al.* (2022) apply the idea of hindsight experience replay (HER) of Andrychowicz *et al.* (2017) to ATP. TRAIL [Abdelaziz *et al.*, 2022] leverages deep RL to learn how to guide a saturation-based theorem prover. It uses a graph neural network to represent a proof state as well as an attention-based policy network to efficiently learn interactions between clauses and actions. Unlike these methods, NIAGRA is based on two key contributions 1) capturing the name-invariant theory-level structural and semantic information when embedding logical clauses and 2) an ensemble approach that uses different configurations of the underlying unoptimized reasoner.

Other relevant non-RL-based approaches using deep learning for proof guidance include [Loos *et al.*, 2017; Paliwal *et al.*, 2020]. These methods use neural networks to select inferences, but, as mentioned before, use local formula representations that exclude theory-level structural information. In addition, the work of [Suda, 2021] used an RNN to process the derivation history of a clause to determine its relevance for proof search. The ENIGMA system of [Jakubuv *et al.*, 2020] uses the symbol-independent GNN introduced in [Olšák *et al.*, 2019] to embed clauses for relevance scoring. As mentioned in the introduction, their approach embeds batches of clauses taken from the unprocessed set together and computes a single fixed relevance score for each clause. Thus, while their method may result in clause representations that are more reflective of the current overall state of a theorem prover, it has the disadvantage of requiring larger-scale operations to be performed more frequently (in contrast to our work, which executes a theory-level embedding operation only once).

Our work on ensembling is related to so-called algorithm portfolios [Huberman *et al.*, 1997]. When solving combinatorial search problems, algorithm portfolios consist of different search algorithms or identical algorithms with different parameter configurations. Having a variety of portfolios allows to examine different portions of the search spaces. While the performance of the algorithm portfolios has been tested in various domains such as SAT solving, puzzle solving and domain-independent planning [Hamadi *et al.*, 2010; Valenzano *et al.*, 2010; Katz *et al.*, 2018], its empirical efficiency has not been explored in unsupervised learning-based ATPs.

6 Conclusion

In this paper, we proposed NIAGRA, a novel graph embedding approach for automated theorem proving aimed at finding a better balance in the trade-off between semantic-faithfulness and computational overhead. In addition, we contributed an ensemble technique for theorem proving that, unlike prior works using ensembles of identical models with different weight initializations, trains models with different configurations of the underlying unoptimized reasoner. Our results on datasets from various domains shows that NIAGRA achieves state-of-the-art performance outperforming the best learning-based reasoners by solving up to 10% more problems. It also provides comparable and sometimes better performance than the best heuristics-based reasoners, despite learning proof strategies completely from scratch. It also exhibits better transfer on hard datasets with diverse domains and vocabulary compared to other learning-based proof guidance systems.

References

[Abdelaziz *et al.*, 2022] Ibrahim Abdelaziz, Maxwell Crouse, Bassem Makni, Vernon Austel, Cristina Cornelio, Shajith Ikbal, Pavan Kapanipathi, Ndivhuwo Makondo, Kavitha Srinivas, Michael Witbrock, et al. Learning to guide a saturation-based theorem prover. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[Andrychowicz *et al.*, 2017] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[Aygün *et al.*, 2022] Eser Aygün, Ankit Anand, Laurent Orseau, Xavier Glorot, Stephen M Mcaleer, Vlad Firoiu, Lei M Zhang, Doina Precup, and Shibl Mourad. Proving theorems using incremental learning and hindsight experience replay. In *International Conference on Machine Learning*, pages 1198–1210. PMLR, 2022.

[Bansal *et al.*, 2019] Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. Holist: An environment for machine learning of higher order logic theorem proving. In *International Conference on Machine Learning*, pages 454–463. PMLR, 2019.

[Bronstein *et al.*, 2017] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[Crouse *et al.*, 2019] Maxwell Crouse, Ibrahim Abdelaziz, Cristina Cornelio, Veronika Thost, Lingfei Wu, Kenneth Forbus, and Achille Fokoue. Improving graph neural network representations of logical formulae with subgraph pooling. *arXiv preprint arXiv:1911.06904*, 2019.

[Crouse *et al.*, 2021] Maxwell Crouse, Ibrahim Abdelaziz, Bassem Makni, Spencer Whitehead, Cristina Cornelio, Pavan Kapanipathi, Kavitha Srinivas, Veronika Thost, Michael Witbrock, and Achille Fokoue. A deep reinforcement learning approach to first-order logic theorem proving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6279–6287, 2021.

[Garland and Lynch, 2000] Stephen J Garland and Nancy A Lynch. Using I/O automata for developing distributed systems. *Foundations of component-based systems*, 13(285-312):5–2, 2000.

[Grabowski *et al.*, 2010] Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.

[Hamadi *et al.*, 2010] Youssef Hamadi, Said Jabbour, and Lakhdar Sais. ManySAT: a parallel SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 6(4):245–262, 2010.

[Hawblitzel *et al.*, 2015] Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R Lorch, Bryan Parno, Michael L Roberts, Srinath Setty, and Brian Zill. Ironfleet: proving practical distributed systems correct. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 1–17, 2015.

[Huberman *et al.*, 1997] Bernardo A Huberman, Rajan M Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51–54, 1997.

[Jakubuv *et al.*, 2020] Jan Jakubuv, Karel Chvalovský, Miroslav Olsák, Bartosz Piotrowski, Martin Suda, and Josef Urban. Enigma anonymous: Symbol-independent inference guiding machine (system description). In *International Joint Conference on Automated Reasoning*, pages 448–463. Springer, 2020.

[Kaliszyk *et al.*, 2018] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 8836–8847, 2018.

[Katz *et al.*, 2018] Michael Katz, Shirin Sohrabi, Horst Samulowitz, and Silvan Sievers. Delfi: Online planner selection for cost-optimal planning. In *Ninth International Planning Competition (IPC-9)*, pages 55–62, 2018.

[Kovács and Voronkov, 2013] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer, 2013.

[Loos *et al.*, 2017] Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. In *Proceedings of the 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, pages 85–105, 2017.

[Lucas, 1993] Peter Lucas. The representation of medical reasoning models in resolution-based theorem provers. *Artificial Intelligence in Medicine*, 5(5):395–414, 1993.

[Masci *et al.*, 2014] Paolo Masci, Yi Zhang, Paul Jones, Paul Curzon, and Harold Thimbleby. Formal verification of medical device user interfaces using PVS. In *International Conference on Fundamental Approaches to Software Engineering*, pages 200–214. Springer, 2014.

[Olšák *et al.*, 2019] Miroslav Olšák, Cezary Kaliszyk, and Josef Urban. Property invariant embedding for automated reasoning. In *24th European Conference on Artificial Intelligence*, 2019.

[Paliwal *et al.*, 2020] Aditya Paliwal, Sarah Loos, Markus Rabe, Kshitij Bansal, and Christian Szegedy. Graph representations for higher-order logic and theorem proving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2967–2974, 2020.

[Rawson and Reger, 2020] Michael Rawson and Giles Reger. Directed graph networks for logical reasoning. In *Workshop on Practical Aspects of Automated Reasoning*, 2020.

[Rawson and Reger, EasyChair 2020] Michael Rawson and Giles Reger. Directed graph networks for logical entailment. EasyChair Preprint no. 2185, EasyChair, 2020.

[Robinson, 1965] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41, 1965.

[Schulz, 2002] Stephan Schulz. E–a brainiac theorem prover. *AI Communications*, 15(2, 3):111–126, 2002.

[Schulz, 2017] Stephan Schulz. We know (nearly) nothing! but can we learn? In *1st International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements*, 2017.

[Suda, 2021] Martin Suda. Vampire with a brain is a good ITP hammer. In *International Symposium on Frontiers of Combining Systems*, pages 192–209. Springer, 2021.

[Valenzano *et al.*, 2010] Richard Valenzano, Nathan Sturtevant, Jonathan Schaeffer, Karen Buro, and Akihiro Kishimoto. Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, pages 177–184, 2010.

[Wang *et al.*, 2017] Mingzhe Wang, Yihe Tang, Jian Wang, and Ji Deng. Premise selection for theorem proving by deep graph embedding. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pages 2783–2793, 2017.

[Zombori *et al.*, 2020] Zsolt Zombori, Josef Urban, and Chad E Brown. Prolog technology reinforcement learning prover: (system description). In *Automated Reasoning: 10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part II*, pages 489–507. Springer, 2020.