Enforcement Heuristics for Argumentation with Deep Reinforcement Learning

Dennis Craandijk[1,2] **and Floris Bex**[2,3]

[1] National Police-lab AI, Netherlands Police
[2] Department Information and Computing Sciences, Utrecht University
[3] Tilburg Institute for Law, Technology and Society, Tilburg University
{d.f.w.craandijk, f.j.bex}@uu.nl

Abstract

In this paper, we present a learning-based approach to the symbolic reasoning problem of dynamic argumentation, where the knowledge about attacks between arguments is incomplete or evolving. Specifically, we employ deep reinforcement learning to learn which attack relations between arguments should be added or deleted in order to enforce the acceptability of (a set of) arguments. We show that our Graph Neural Network (GNN) architecture EGNN can learn a near optimal enforcement heuristic for all common argument-fixed enforcement problems, including problems for which no other (symbolic) solvers exist. We demonstrate that EGNN outperforms other GNN baselines and on enforcement problems with high computational complexity performs better than state-of-the-art symbolic solvers with respect to efficiency. Thus, we show our neuro-symbolic approach is able to learn heuristics without the expert knowledge of a human designer and offers a valid alternative to symbolic solvers. We publish our code at https://github.com/DennisCraandijk/DL-Abstract-Argumentation.

Introduction

Recent years have seen rapid developments in neuro-symbolic computing, which aim to put together learning in (deep) neural networks with reasoning and explainability via symbolic representations (d'Avila Garcez et al. 2019). A growing body of literature in this field combines deep learning with reinforcement learning to automatically learn heuristics for symbolic reasoning problems (such as game playing (Silver et al. 2018) and combinatorial optimization (Bengio, Lodi, and Prouvost 2021)). The appeal of this deep reinforcement learning paradigm is that solvers can be learned end-to-end without the tailoring and expert knowledge of a human designer.

One domain where learning based methods are a promising alternative for symbolic methods is computational argumentation. With applications in multi-agent systems, decision-making tools, medical and legal reasoning, computational argumentation has become a major subfield of AI (Atkinson et al. 2017). The foundation of much of the theory of computational argumentation is based on the seminal work by Dung (1995), who introduced abstract argumentation frameworks (AFs) - representing arguments and attacks between these arguments - and several acceptability semantics that define which sets of arguments (*extensions*) can be reasonably accepted. Recent work demonstrates the use of a graph neural network (GNN) to learn to determine which arguments are (part of) an extension (Kuhlmann and Thimm 2019; Craandijk and Bex 2020; Malmqvist et al. 2020) - computational problems which are normally solved with handcrafted symbolic methods (Charwat et al. 2015; Gaggl et al. 2020). These approaches however, learn by supervision of an existing solver used to generate the training data, rather than end-to-end.

In this work we employ a graph-based deep reinforcement learning algorithm on the dynamic argumentation problem of enforcement (Baumann and Brewka 2010): given sets of arguments that we (do not) want to accept, how to modify the argumentation framework in such a way that these arguments are (not) accepted, while minimizing the number of changes (Baumann 2012). Here, it is possible to distinguish between *extension enforcement* – modifying an argumentation framework in such a way that a given set of arguments becomes an extension (Baumann and Brewka 2010) – and *status enforcement* (Niskanen, Wallner, and Järvisalo 2016) – modifying an argumentation framework in such a way that the arguments in one set become accepted while at the same time the arguments in another set are not accepted.

While there has been ample formal work on enforcement and its complexity (Doutre and Mailly 2018; Wallner, Niskanen, and Järvisalo 2017), only a few automated solvers currently exist, which all translate the problem into a symbolic formalism for which a dedicated solver exists. In contrast to solvers for static argumentation (i.e. determining acceptability or extensions under various semantics), for which there is a lively community (Gaggl et al. 2020), there is less work on solvers for dynamic enforcement problems, with enforcement not being part of the ICCMA competitions[1]. Existing algorithms on extension and status enforcement have been published by Coste-Marquis et al. (2015); Wallner, Niskanen, and Järvisalo (2017); Niskanen, Wallner, and Järvisalo (2016); Niskanen, Wallner, and Järvisalo (2018); Niskanen and Järvisalo (2020b) who only tackle some variants of status enforcement. Furthermore, we have found that existing symbolic solvers demonstrate a quite significant drop in run-

[1] See argumentationcompetition.org and (Gaggl et al. 2020)

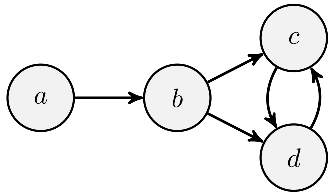

Figure 1: Graph representation of the AF F_e.

time efficiency when confronted with some of the enforcement problems that are higher in the complexity hierarchy, which severely limits their practical applicability. So there is a need for efficient heuristics to tackle these problems. However, designing such heuristics takes considerable effort and domain knowledge on the part of an expert, who generally has to come up with a new heuristic for every type of problem or semantics. As far as we are aware, there exist no heuristics for enforcement problems.

In this paper, we therefore show that it is possible to *learn* an enforcement heuristic for all extension and status enforcement problems under the most common semantics with only a single architecture. We compare our Enforcement Graph Neural Network (EGNN) to symbolic solvers and two other deep learning models based on recent work in deep learning for argumentation (Craandijk and Bex 2020; Kuhlmann and Thimm 2019; Malmqvist et al. 2020), showing that EGNN often outperforms these baselines on efficiency (solutions found within a time limit) while staying very close to the symbolic solvers in terms of optimality (steps taken to find a solution). We employ a deep reinforcement learning approach that automatically learns to generate solutions that only need to be verified. Such verification is a static argumentation problem for which there are many existing algorithms, thus allowing us to also learn heuristics for dynamic enforcement problems for which no symbolic solvers yet exist.

We start by discussing argumentation and enforcement preliminaries in Section . Section then sets up enforcement in dynamic argumentation as a reinforcement learning problem, and Section discusses our Deep Q Network (DQN) model based on a Graph Neural Network (GNN). Section discusses the experimental setup (data, training parameters), and Section discusses the results. We end with a conclusion in Section .

Preliminaries

Argumentation Frameworks

We recall abstract argumentation frameworks (Dung 1995).

Definition 1. *An abstract* argumentation framework *(AF) is a pair $F = (A, R)$ where A is a (finite) set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that a* attacks *b. A set $S \subseteq A$ attacks b if there is an $a \in S$, such that $(a, b) \in R$. An argument $a \in A$ is* defended *by $S \subseteq A$ iff, for each $b \in A$ such that $(b, a) \in R$, S attacks b.*

Example 1. *Figure 1 illustrates the AF $F_e = (\{a, b, c, d\}, \{(a, b), (b, c)(b, d), (c, d), (d, c)\})$.*

Dung-style semantics define the sets of arguments that can jointly be accepted (*extensions*). A σ-extension refers to an extension under semantics σ. We consider admissible sets and preferred, complete, grounded and stable semantics with the following functions respectively adm, prf, com, grd, stb.

Definition 2. *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is* conflict-free *(in F), if there are no $a, b \in S$, such that $(a, b) \in R$. The collection of sets which are conflict-free is denoted by $\mathsf{cf}(F)$. For $S \in \mathsf{cf}(F)$, it holds that:*

- *$S \in \mathsf{adm}(F)$, if each $a \in S$ is defended by S;*
- *$S \in \mathsf{prf}(F)$, if $S \in \mathsf{adm}(F)$ and for each $T \in \mathsf{adm}(F)$, $S \not\subset T$;*
- *$S \in \mathsf{com}(F)$, if $S \in \mathsf{adm}(F)$ and for each $a \in A$ defended by S it holds that $a \in S$;*
- *$S \in \mathsf{grd}(F)$, if $S \in \mathsf{com}(F)$ and for each $T \in \mathsf{com}(F)$, $T \not\subset S$;*
- *$S \in \mathsf{stb}(F)$, if for each $a \in A \setminus S$, S attacks a.*

Example 2. *The extensions of F_e under the preferred, complete and grounded semantics are: $\mathsf{prf}(F) = \{\{a, c\}, \{a, d\}\}$; $\mathsf{com}(F) = \{\{a\}, \{a, c\}, \{a, d\}\}$; $\mathsf{grd}(F) = \{a\}$; $\mathsf{stb}(F) = \{\{a, c\}, \{a, d\}\}$.*

Given an AF, we can thus determine all σ-extensions of some AF F (enumeration), or verify whether a given set S is a σ-extension of F (verification), the latter being denoted by $\mathsf{Ver}_\sigma(S, F)$.

Furthermore, for some argument a that is part of F, we can determine if it is *credulously accepted* under semantics σ – a is contained in at least one σ-extension – or *sceptically accepted* under σ – a is contained in all σ-extensions.

Example 3. *Under the preferred semantics, only argument a is sceptically accepted and arguments a, c and d are credulously accepted in F_e.*

Enforcement

Extension Enforcement Extension enforcement concerns modifying an argumentation framework in such a way that a given set of arguments S becomes an extension (Baumann and Brewka 2010). Enforcing an extension can be accomplished by changing the attack structure, the arguments in an AF, the evaluation semantics or a combination of those (Doutre and Mailly 2018). Since under some change operators it is impossible to enforce an extension, Coste-Marquis et al. (2015) constrained the problem to argument-fixed extension enforcement, where arguments and semantics are fixed and only the attack relations are subject to change. Changing the attack structure while fixing the arguments and semantics can be relevant in situations where information about the attack structure is not complete (the existence or direction of an attack may be unknown for instance). As argument-fixed extension enforcement problems are guaranteed to have a solution - making solvers easier to verify - this approach is adopted in this research. Finally, a distinction is made between a *strict* setting where an AF should be modified such that S becomes an extension or the *non-strict* setting where it suffices that S is a subset of an extension.

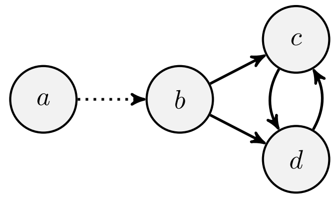

Figure 2: Changing the AF F_e from Figure 1 into AF F_e' by removing the attack (a, b).

Definition 3. *Given an AF $F = (A, R)$, a semantics σ and a set of arguments $S \subseteq A$. The goal is to change the attack structure R into R' such that for strict enforcement, S becomes an extension under semantics σ in the modified AF $F' = (A, R')$.* $\mathsf{Strict}_\sigma(S, F)$ *is then the set*

- $\{R' \mid F' = (A, R'), S \in \sigma(F')\}$

For non-strict enforcement, S should become a subset of an extension in the modified AF $F' = (A, R')$. $\mathsf{Non-strict}_\sigma(S, F)$ *is then the set*

- $\{R' \mid F' = (A, R'), \exists S' \in \sigma(F') : S \subseteq S'\}$

Example 4. *Consider enforcing $\{b\}$ in the argumentation framework F_e from example 1. Under complete, grounded, preferred and stable semantics $\{b\}$ can be non-strictly enforced by removing the attack (a, b) as shown in Figure 2, as after removal we have $\mathit{com}(F_e') = \mathit{prf}(F_e') = \mathit{grd}(F_e') = \mathit{stb}(F_e') = \{\{a, b\}\}$.*

Note that $\mathsf{Non-strict}_{\mathsf{adm}} = \mathsf{Non-strict}_{\mathsf{com}} = \mathsf{Non-strict}_{\mathsf{prf}}$ (Wallner, Niskanen, and Järvisalo 2017, Theorem 1).

Status Enforcement Status enforcement (Niskanen, Wallner, and Järvisalo 2016) concerns modifying an argumentation framework in such a way that every argument in a positive set P becomes (sceptically/credulously) accepted, and at the same time every argument in a negative set N becomes not (sceptically/credulously) accepted. Thus, it connects static argumentation (credulous/sceptical acceptance) and dynamic argumentation (extension enforcement) – for example, enforcing P to be (a subset of) an extension of some AF F' also leads to F' enforcing positive credulous acceptance for the arguments in P.

Definition 4. *Given an AF $F = (A, R)$, a semantics σ, sets of arguments $P \subseteq A$ and $N \subseteq A$, where $P \cap N = \emptyset$. The goal is to change the attack structure R into R' such that for credulous status enforcement* Cred *under semantics σ in the modified AF $F' = (A, R')$, each argument in P is credulously accepted (in at least one σ-extension) and each argument in N is not credulously accepted (not in any σ-extension).* $\mathsf{Cred}_\sigma(P, N, F)$ *is then the set*

- $\{R' \mid F' = (A, R'), P \subseteq \bigcup \sigma(F'), N \cap \bigcup \sigma(F') = \emptyset\}$

For sceptical status enforcement Scept*, each argument in P should be sceptically accepted (in all σ-extensions) and each argument in N should not be sceptically accepted (excluded from at least one σ-extension) in the modified AF $F' = (A, R')$.* $\mathsf{Scept}_\sigma(P, N, F)$ *is then the set*

- $\{R' \mid F' = (A, R'), P \subseteq \bigcap \sigma(F'), N \cap \bigcap \sigma(F') = \emptyset\}$

Example 5. *Consider argument enforcement in the AF F_e from Figure 1 (extensions in Example 2), with $P = \{b\}$ and $N = \{c, d\}$. Under complete, preferred, grounded and stable semantics, these arguments can be enforced credulously and sceptically by removing the attack (a, b) (Figure 2), as after removal we have $\mathit{com}(F_e') = \mathit{prf}(F_e') = \mathit{grd}(F_e') = \mathit{stb}(F_e') = \{\{a, b\}\}$.*

Note that $\mathsf{Cred}_{\mathsf{adm}} = \mathsf{Cred}_{\mathsf{com}} = \mathsf{Cred}_{\mathsf{prf}}$ and $\mathsf{Cred}_{\mathsf{grd}} = \mathsf{Scept}_{\mathsf{grd}} = \mathsf{Scept}_{\mathsf{com}}$ (Niskanen, Wallner, and Järvisalo 2016, Proposition 6).

Minimal Change Since one generally wants to avoid endlessly adding or removing elements in order to enforce the acceptability status of arguments, enforcement is modelled as an optimization problem where the goal is to enforce arguments while minimizing the amount of change to the framework (Baumann 2012). The number of changes in argument-fixed extension enforcement is defined by the Hamming distance between two attack structures.

$$|R \Delta R'| = |R \setminus R'| + |R' \setminus R|$$

The problem thus becomes enforcing arguments by changing the attack structure while minimizing the Hamming distance between the original and modified attack structures.

Algorithms and Complexity

Wallner, Niskanen, and Järvisalo (2017) and Niskanen, Wallner, and Järvisalo (2016) study the computational complexity of argument-fixed extension and argument enforcement, respectively. Like many argumentation problems (static and dynamic), the complexity of the extension enforcement problem quickly becomes quite complex. For nearly all extension enforcement problems and semantics, the complexity is NP-complete, with $\mathsf{Strict}_{\mathsf{prf}}$ and nearly all status enforcement problems making a further jump in complexity to Σ_2^P. The hardest problem is $\mathsf{Scept}_{\mathsf{prf}}$ at Σ_3^P. These different levels of complexity will become clearly visible in our experiments (Section).

To the best of our knowledge only Coste-Marquis et al. (2015) and Wallner, Niskanen, and Järvisalo (2017) provide algorithms for solving argument-fixed extension enforcement problems. These reduction-based approaches first reduce the problem to a formula in propositional logic to benefit from existing dedicated Boolean satisfiability (SAT) solvers. For instance, Wallner, Niskanen, and Järvisalo (2017) use a maximum satisfiability solver (MaxSAT) in order to find an optimal solution for extension enforcement under various semantics. Wallner, Niskanen, and Järvisalo (2017) implement the algorithms in the Pakota and Maadoita software[2], which support optimal extension enforcement, both strictly and non-strictly, under all current semantics.

Niskanen, Wallner, and Järvisalo (2016) further provide status enforcement algorithms in Pakota for optimal credulous and sceptical enforcement under stable and admissible semantics (the latter of which is also a solution to $\mathsf{Cred}_{\mathsf{com}}$ and $\mathsf{Cred}_{\mathsf{prf}}$, see Section). For $\mathsf{Scept}_{\mathsf{com}}$ (equal to $\mathsf{Scept}_{\mathsf{grd}}$

[2] https://www.cs.helsinki.fi/group/coreo/pakota/

and Cred_{grd}), see Section) and $\text{Scept}_{\text{prf}}$, there are as far as we are aware no (symbolic) solvers currently available.

When learning new heuristics for enforcement problems, we need at the very least be able to verify whether an extension or argument is enforced, as a reinforcement learning algorithm needs to know whether a terminal state (in which some set of arguments is enforced) is reached. Luckily, for verifying the acceptability status of (a set of) arguments numerous algorithms already exist (Gaggl et al. 2020; Dvorák and Dunne 2017). Furthermore, verifying acceptance is in many cases less computationally complex than enforcing acceptance (Charwat et al. 2015).

Reinforcement Learning

Unlike supervised machine learning, reinforcement learning (RL) does not rely on labelled training data. Rather, RL considers an *agent* that explores an *environment* by taking *actions* and aims to learn a *policy* that maximizes the received cumulative *reward* (indicating how well it achieved its goal). Applying this to enforcement, we consider an argumentation framework as the environment in which an agent is rewarded for enforcing a set of arguments by introducing or removing attacks between arguments (i.e. *flipping* an attack relation).

RL problems are typically modelled as a Markov Decision Process (MDP). Thus, more formally, we define the MDP for enforcement as the tuple (St, Ac, Tr, Re), where given an AF $F = (A, R)$, a semantics σ and a set of arguments to be enforced $S \subseteq A$ (where for status enforcement $S = P \cup N$).

- St denotes the set of states where each state $s \in St$ represents (a modification of) the AF F. The initial state $s_0 \in St$ consists of the original AF F. A terminal state $\hat{s} \in St$ consists of a modified F' where S is enforced.
- Ac denotes the action space, consisting of all possible attack relations $A \times A$.
- $Tr : St \times Ac \to St$ denotes the transition function, mapping a state-action pair to the next state. The transition function takes the current state $s \in St$ with AF F and flips the attack relation $a \in Ac$ to produce the new state $s' \in St$ with the modified AF F'.
- $Re : St \to \{0, -1\}$ is the reward function where the reward r is 0 for reaching a terminal state and -1 otherwise.

Given a state $s \in St$ an agent performs an action $a \in Ac$ that produces a transition to the next state $s' \in St$ and provides the agent with a reward r. The goal is to find a policy $\pi : St \to Ac$, describing the probability of taking an action from a given state, that maximizes the cumulative reward at the end of an episode. Our MDP formulation of enforcement encourages the agent first to find a solution (to receive a non-negative reward), and only then to minimize the number of changes made to the AF (to maximize the reward). Additionally, our reward function only requires verifying when a terminal stage is reached (i.e. verifying the acceptance status of the arguments which are to be enforced).

Deep Q-Learning

Traditionally, RL problems are solved with tabular methods such as Q-learning, where a Q-table (a look-up table consisting of all possible combinations of states and actions) is iteratively updated with Q-values reflecting the goodness of each action given a state (Sutton and Barto 1998). The Q-value of a state-action pair $(s \in St, a \in Ac)$ is given by the expected discounted sum of rewards

$$Q_\pi(s, a) = \mathsf{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t) | s_0 = s, a_0 = a, \pi] \qquad (1)$$

where $\gamma = [0, 1]$ denotes the discount factor and each action is chosen according to the current policy π. Although Q-learning can discover an optimal policy for any given MDP, for computationally complex problems such as enforcement, constructing the Q-table becomes intractable due to the large state-action space. Moreover, a policy based on Q-tables can only handle previously seen states and therefore does not generalize to new problem instances. To overcome these problems a Deep Q-network (DQN) (Mnih et al. 2015) replaces the look-up table with a neural network that learns to predict the Q-values for any given state. The learned Q-value function is used to guide a greedy policy $\pi(s) = \text{argmax}_a Q(s, a)$, yielding an approximation of the optimal policy.

Model

We propose our enforcement model, which uses a GNN to learn a Q-value function for our MDP formulation of enforcement problems. Our model takes an AF and maps it to a fully connected graph where nodes and edges have a vectorial representation denoting which nodes represent an argument that should be enforced and which edges represent an existing attack relation. Next, the node vectors are iteratively updated by exchanging vectorial messages with their neighbours in a process called *message passing*. After a number of message passing steps the node and edge representations are mapped to the predicted Q-value for each edge.

Formally, let $G = (V, E)$ be a directed and fully connected graph representation of an AF F, where at message passing step $t = 0$ each node i is assigned a real-valued vector $v_i^t \in V$ indicating whether it represents an argument that should be enforced, and each edge between a node i and j is assigned a real-valued vector $e_{ij} \in E$ indicating whether it represents an existing attack. At subsequent steps t, each node i aggregates messages m_{ij}^t from its neighbours j and updates its vector representation, such that

$$m_{ij}^{t+1} = \text{MSG}(v_i^t, v_j^t, e_{ij}, e_{ji}) \qquad (2)$$

$$v_i^{t+1} = \text{UPDT}(v_i^t, v_i^0, \underset{j \in N(i)}{\text{AGGR}}(m_{ij}^{t+1})) \qquad (3)$$

where $N(i)$ denotes all neighbours of node i. The message function MSG computes a messages based on the vectors of two connected nodes, along with the edge representations of both directed edges which mark the existence of a (counter-)attack and whether the attack has previously been flipped. The update function UPDT updates the node representation by taking the previous node vector, the initial node vector and the messages aggregated by AGGR. The message and update functions are parameterized neural networks which, in conjunction with the aggregation function, yield a neural message passing algorithm whose parameters can be tuned to solve various enforcement problems.

After each message passing step, an edge can be read out with the readout function

$$Q_{ij}^t = \mathrm{READ}(v_i^t, v_j^t, e_{ij}) \tag{4}$$

that maps the updated node representation and the edge representation to the predicted Q-value for that edge. The readout function is also a parameterized function devised to map the multidimensional representations to a Q-value.

Neighbourhood Aggregation

Recently, several studies demonstrated that the choice of aggregation function can considerably affect the performance of a GNN. The purpose of AGGR is to aggregate the incoming messages into a single fixed-size vector such that it can be fed into the update function while sufficiently expressing the information contained in the incoming messages. A suitable aggregation function extracts information from an arbitrary number of messages and should be independent to the equally arbitrary ordering of those messages. Prior GNN studies proposed: learnable methods such as *convolutions* (Kipf and Welling 2017); injective methods such as *summation* (SUM) (Xu et al. 2019); and statistical methods such as *mean* (MEAN), *maximum* (MAX), *minimum* (MIN) and *standard deviation* (STD) (Corso et al. 2020). From these, convolutions and SUM have been applied to the static argumentation problem of predicting argument acceptance. Where Kuhlmann and Thimm (2019); Malmqvist et al. (2020) show reasonable performance with a Graph Convolutional Network, Craandijk and Bex (2020) show almost perfect performance on the static argument acceptance task using the SUM aggregator.

Although SUM is theoretically proven to be the most expressive aggregator (Xu et al. 2019), it might be unfit for enforcement problems. Since our model exchanges messages on a fully connected graph, the amount of incoming messages can vary significantly between different sized AFs. As the SUM aggregator amplifies messages, especially over multiple message passing steps, a large shift in node degree can lead to unstable node embeddings (Joshi et al. 2020). Recent work suggests using a combination of node degree agnostic statistical aggregation metrics (MEAN, MAX, MIN, STD), where each function serves a different purpose (Corso et al. 2020). MEAN computes the average of incoming messages, MIN and MAX can distinguish discrete signals, and STD quantifies the distribution of the incoming messages.

Experimental Setup

For all enforcement variants and semantics described in Section , we train three models end-to-end with deep Q-learning. GCN uses a convolutional aggregator as used by Kuhlmann and Thimm (2019); Malmqvist et al. (2020), AGNN uses a sum aggregator as used by Craandijk and Bex (2020), and EGNN uses a combination of MEAN, MAX, MIN, STD aggregators.

We sample AFs uniformly from all AF families implemented in the following generators from ICCMA (Gaggl et al. 2020): *AFBenchGen2*, *AFGen Benchmark Generator*, *GroundedGenerator*, *SccGenerator*, *StableGenerator*. The generation parameters are chosen randomly for each sample, resulting in a diverse set of AFs. To avoid duplicates, each AF is checked for isomorphism with *Nauty* (McKay and Piperno 2014). For each AF $F = (A, R)$ we generate extension enforcement problems by randomly selecting a set of arguments $S \subseteq A$ which currently isn't enforced and such that $0 < |S| < |A|$. Status enforcement problems are generated by taking S and randomly splitting it into a positive set P and negative set N such that $P \cup N = S$. We generate training instances with $|A|$ from $(3, 4, 5, ..., 9)$ and 1000 validation instances containing $|A| = 10$ arguments to train the network. We verify when arguments are successfully enforced by enumerating the extensions with the sound and complete *μ-toksia* solver (Niskanen and Järvisalo 2020a). Finally, to evaluate the performance we generate two test datasets of 1000 instances containing $|A| \in \{10, 20, 50\}$. The $|A| = 10$ test set tests how well the learned algorithm generalizes from the training data to AFs not seen during training. The $|A| = 20$ and $|A| = 50$ sets are used to evaluate the scalability of the solvers.

Results

We assess the performance of all methods with respect to two goals: finding a solution within an acceptable timeframe (efficiency) and minimizing the number of changes to the argumentation framework (optimality). We measure efficiency by the number of instances *solved* within a timeout limit, which is reached after 15 minutes[3] or if the maximum number of changes necessary to enforce a set of arguments (i.e. the total number of edges) have been performed. For optimality, we measure the *approximation ratio* with respect to the number of changes made by the exact symbolic solver[4] (Niskanen, Wallner, and Järvisalo 2016; Niskanen and Järvisalo 2020b). Note that for $\mathsf{Scept}_{\mathsf{grd}}$, $\mathsf{Scept}_{\mathsf{prf}}$, $\mathsf{Scept}_{\mathsf{com}}$, $\mathsf{Cred}_{\mathsf{grd}}$ we cannot obtain approximation ratios as currently no solver exists. Table 1 shows the results for all tasks under all semantics on the test sets for generalization ($|A| = 10$) and scalability ($|A| \in \{20, 50\}$).

Graph Neural Networks

First, we compare the different graph neural network models on the test sets where $|A| \in \{10, 20\}$. It is apparent from Table 1 that GCN is not able to learn an effective enforcement heuristic. On both the $|A| = 10$ and $|A| = 20$ test sets, it fails to solve a large fraction of the problem instances and exhibits high approximation ratios. AGNN and EGNN

[3]This is the same timeout limit as used by (Niskanen, Wallner, and Järvisalo 2016; Wallner, Niskanen, and Järvisalo 2017).

[4]*Maadoita* (for the grounded semantics) and *Pakota* (for the other semantics).

$\lvert A \rvert$		Model	Strict_σ				$\mathsf{Non-strict}_\sigma$			Cred_σ			Scept_σ	
			grd	prf	stb	com	grd	prf	stb	grd	prf	stb	prf	stb
10	Solved	Symb.	1000	1000	1000	1000	1000	1000	1000	-	1000	1000	-	1000
		GCN	13	116	94	137	690	644	633	454	693	583	532	466
		AGNN	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
		EGNN	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
	Ratio	Symb.	1.000	1.000	1.000	1.000	1.000	1.000	1.000	-	1.000	1.000	-	1.000
		GCN	10.74	18.91	9.869	9.598	3.677	5.072	5.249	-	9.782	12.75	-	6.073
		AGNN	1.016	1.191	1.001	1.007	1.017	1.022	1.039	-	1.086	1.132	-	1.020
		EGNN	1.008	1.181	1.000	1.007	1.004	1.005	1.002	-	1.070	1.085	-	1.018
20	Solved	Symb.	1000	999	1000	1000	1000	1000	1000	-	986	984	-	995
		GCN	1	66	58	65	465	431	427	274	395	360	319	261
		AGNN	666	973	961	961	988	1000	986	998	876	848	974	926
		EGNN	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
	Ratio	Symb.	1.000	1.000	1.000	1.000	1.000	1.000	1.000	-	1.000	1.000	-	1.000
		GCN	2.500	63.97	19.51	14.64	8.280	12.35	12.59	-	22.83	29.31	-	15.94
		AGNN	10.01	2.303	1.654	3.228	3.458	4.091	7.667	-	7.865	5.538	-	9.960
		EGNN	1.091	1.211	1.032	1.010	1.036	1.040	1.055	-	1.605	1.534	-	1.251
50	Solv.	Symb.	984	880	1000	999	739	999	999	-	452	544	-	829
		EGNN	1000	945	1000	1000	1000	1000	1000	1000	1000	1000	1000	998
	Ratio	Symb.	1	1	1	1	1	1	1	-	1	1	-	1
		EGNN	1.576	2.515	1.270	1.029	1.147	1.668	1.792	-	2.934	4.515	-	3.799

Table 1: Enforcement results on the $|A| = 10$, $|A| = 20$ and $|A| = 50$ test sets. The approximation ratios are only reported for instances solved within the timeout limit. For all coinciding problems, notably $\mathsf{Non-strict}_{\mathsf{prf}} = \mathsf{Non-strict}_{\mathsf{com}}$, $\mathsf{Cred}_{\mathsf{prf}} = \mathsf{Cred}_{\mathsf{com}}$, and $\mathsf{Cred}_{\mathsf{grd}} = \mathsf{Scept}_{\mathsf{grd}} = \mathsf{Scept}_{\mathsf{com}}$ we only report the former.

on the other hand, show to generalize from the training data by learning an efficient heuristic that solves all $|A| = 10$ instances. When scaling up to size $|A| = 20$ however, AGNN fails to find a solution for a considerable number of problem instances, while EGNN still solves all instances. This difference is also apparent when comparing optimality. Where both AGNN and EGNN find near optimal solutions on the $|A| = 10$ set, scaling to $|A| = 20$ causes a significant drop in performance for AGNN, while EGNN stays in proximity to the optimal solution. These results confirm the scalability issues of the SUM aggregator as discussed in Section . Since EGNN outperforms the other GNN architectures on all evaluation methods, we use EGNN for our further experiments.

Symbolic Solver

If we now compare EGNN to the symbolic solver, the difference between an exact solver and the learnt heuristic become apparent. The symbolic solver is guaranteed to find an optimal solution, and does so for all enforcement tasks on the $|A| = 10$ set. However, on larger AFs, the time needed to find a solution on problems that exhibit a high computational complexity can exceed the acceptable limits, which leads to the symbolic solver not being able to solve all instances in the test sets for $|A| \in \{20, 50\}$. For $\mathsf{Strict}_{\mathsf{prf}}$ and all status enforcement problems the fraction of unsolved instances is relatively modest under $|A| = 20$. However, on the $|A| = 50$ test set the symbolic solver fails to find all solutions under almost all enforcement problems and semantics. The extension enforcement problems $\mathsf{Strict}_{\mathsf{prf}}$ and $\mathsf{Non-strict}_{\mathsf{grd}}$ show quite significant perfomance drops with only 880 and 739 solved, respectively. For credulous status enforcement problems, the performance of the symbolic solver drops even further, with the solver only finding a solution in just around half of the instances for $\mathsf{Cred}_{\mathsf{prf}}$ and $\mathsf{Cred}_{\mathsf{stb}}$.

EGNN, on the other hand, is primarily optimized to find a solution, with minimizing the number of changes as a secondary goal. EGNN solves all instances on both the $|A| = 10$ and $|A| = 20$ sets within the time limit and reaches near optimal approximation ratios on most problems. On the $|A| = 50$ set, EGNN solves all instances across almost all tasks, thereby outperforming the symbolic solver, especially on the status enforcement problems (Cred,Scept) and $\mathsf{Non-strict}_{\mathsf{grd}}$ and $\mathsf{Strict}_{\mathsf{prf}}$. Scaling up the AFs leads to an increase in approximation ratios and causes EGNN to not solve some instances for $\mathsf{Strict}_{\mathsf{prf}}$ and $\mathsf{Scept}_{\mathsf{stb}}$. This is not surprising since the symbolic solver results show that finding an optimal solution for these problems is generally hard.

New Solvers

For $\mathsf{Scept}_{\mathsf{com}}$ (equal to $\mathsf{Scept}_{\mathsf{grd}}$ and $\mathsf{Cred}_{\mathsf{grd}}$) and $\mathsf{Scept}_{\mathsf{prf}}$, we obtain, as far as we are aware, the first solver on these tasks. EGNN solves all problem instances on both test sets. Since currently no solver exists, we cannot report approximation ratios. However, the number of changes performed is

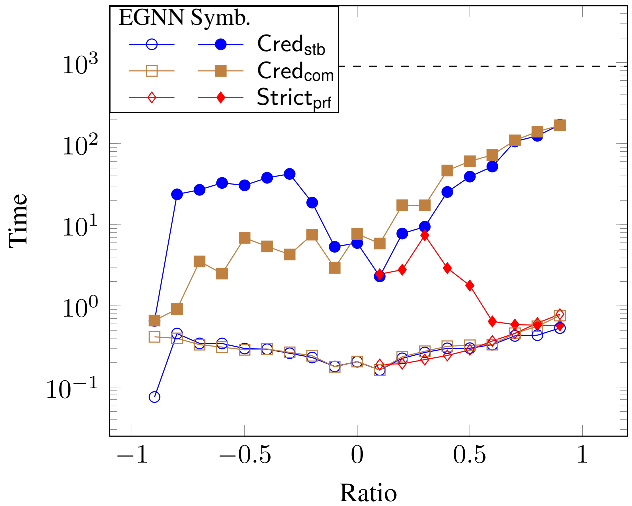

Figure 3: The runtime in seconds as a function of the enforcement ratio for EGNN and the symbolic solver. The dashed line indicates the timeout limit. Since Cred$_{\mathsf{com}}$ and Cred$_{\mathsf{prf}}$ coincide, we only report the former.

in line with the results on other tasks, indicating the approximation ratios are likely to be similar as well. These results indicate our method is able to learn near-optimal enforcement heuristics, even on problems for which currently no solver exists.

Efficiency

We now turn to a closer inspection of the differences between the symbolic solver and EGNN in terms of efficiency. We stress that it is not our main goal to design an algorithm with the lowest runtime. Implementing methods for speeding up deep learning models constitutes an active research area (Cheng et al. 2018) and falls beyond the scope of this research. Besides, due to differences between the deep learning and symbolic paradigms in terms of implementation (Python vs C++) and hardware (GPU vs CPU), runtimes can vary by two orders of magnitude and are thus hard to compare (Kool, van Hoof, and Welling 2019). Nevertheless, comparing runtimes provides insight in how the complexity of a problem affects the efficiency of both methods. We find that the runtime of the symbolic solver is dependent on the fraction of arguments that should be enforced positively or negatively. We express this *enforcement ratio* by $\frac{|P|-|N|}{|A|}$, where $P = S$ and $N = \emptyset$ for all extension enforcement problems. The enforcement ratio equals -1 when all arguments have to be enforced negatively, 1 when all arguments have to be enforced positively, and 0 when the number of arguments that should be enforced positively and negatively are equal. We demonstrate the effect on the runtimes by sampling 10.000 AFs with uniformly random enforcement ratios and $|A| = 20$. Figure 3 compares the runtimes of both solvers with respect to the enforcement ratio on problems where the symbolic solver reaches the timeout limit. From the figure, it is apparent that the runtime of the symbolic solver increases when the fraction of arguments that should be enforced positively increases for Cred$_{\mathsf{stb}}$ and Cred$_{\mathsf{com}}$. Under Strict$_{\mathsf{prf}}$ enforcement, the runtime increases when the fraction of arguments that should be enforced decreases. The efficiency of EGNN, on the other hand, is only dependent on the size of the AF, the number of message passing steps and the number of changes it performs. As a result, the runtimes stay almost constant and even outperform the symbolic solver on nearly all test instances, despite not being optimized for speed.

Discussion

Related Work

The past few years have seen an increasing amount of research effort directed to using GNNs for combinatorial optimization problems (Bengio, Lodi, and Prouvost 2021). However, existing work on neuro-symbolic methods for argumentation only focuses on the static problem of determining which arguments are (part of) an extension (Kuhlmann and Thimm 2019; Craandijk and Bex 2020; Malmqvist et al. 2020). Furthermore, all these approaches are supervised by an existing solver, which makes them unsuitable for learning new solvers like we did in this paper.

With regard to reinforcement learning, some work exists on applying tabular Q-learning methods for argument selection in an argumentation dialogue (Alahmari, Yuan, and Kudenko 2019; Georgila and Traum 2011). Furthermore, Gao and Toni (2014) use an argumentation framework to represent domain knowledge in a multi-agent reinforcement learning environment. Although these authors use (standard) reinforcement learning in conjunction with argumentation, they do not tackle the problem of (finding heuristics for) enforcement in argumentation.

Conclusion

The research has shown that it is possible to learn a near optimal heuristic for various enforcement problems with a single graph neural network architecture through reinforcement learning. The proposed approach is not dependent on the supervision of an existing solver, but learns a heuristic end-to-end simply by verifying when a set of arguments has been enforced, enabling the discovery of heuristics for previously unsolved problems. The experimental results support the idea that solvers can be learned end-to-end with deep reinforcement learning without the tailoring and expert knowledge of a human designer. Further research is required to determine whether the approximation ratios and scalability can be improved.

Acknowledgements

This research was supported by the Netherlands Police and was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

References

Alahmari, S.; Yuan, T.; and Kudenko, D. 2019. Reinforcement Learning for Dialogue Game Based Argumentation. In Grasso, F.; Green, N.; Schneider, J.; and Wells, S., eds., *Proceedings of the 19th Workshop on Computational Models of Natural Argument co-located with the 14th International Conference on Persuasive Technology, CMNA@PERSUASIVE 2019, Limassol, Cyprus, April 9, 2019*, volume 2346 of *CEUR Workshop Proceedings*, 29–37. CEUR-WS.org.

Atkinson, K.; Baroni, P.; Giacomin, M.; Hunter, A.; Prakken, H.; Reed, C.; Simari, G. R.; Thimm, M.; and Villata, S. 2017. Towards Artificial Argumentation. *AI Magazine*, 38(3): 25–36.

Baumann, R. 2012. What Does it Take to Enforce an Argument? Minimal Change in abstract Argumentation. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, 127–132.

Baumann, R.; and Brewka, G. 2010. Expanding Argumentation Frameworks: Enforcing and Monotonicity Results. In Baroni, P.; Cerutti, F.; Giacomin, M.; and Simari, G. R., eds., *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, 75–86. IOS Press.

Bengio, Y.; Lodi, A.; and Prouvost, A. 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2): 405–421.

Charwat, G.; Dvorák, W.; Gaggl, S. A.; Wallner, J. P.; and Woltran, S. 2015. Methods for solving reasoning problems in abstract argumentation - A survey. *Artificial Intelligence*, 220: 28–63.

Cheng, Y.; Wang, D.; Zhou, P.; and Zhang, T. 2018. Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges. *IEEE Signal Process. Mag.*, 35(1): 126–136.

Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; and Velickovic, P. 2020. Principal Neighbourhood Aggregation for Graph Nets. *CoRR*, abs/2004.05718.

Coste-Marquis, S.; Konieczny, S.; Mailly, J.; and Marquis, P. 2015. Extension Enforcement in Abstract Argumentation as an Optimization Problem. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2876–2882.

Craandijk, D.; and Bex, F. 2020. Deep Learning for Abstract Argumentation Semantics. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 1667–1673. ijcai.org.

d'Avila Garcez, A. S.; Gori, M.; Lamb, L. C.; Serafini, L.; Spranger, M.; and Tran, S. N. 2019. Neural-symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning. *FLAP*, 6(4): 611–632.

Doutre, S.; and Mailly, J. 2018. Constraints and changes: A survey of abstract argumentation dynamics. *Argument & Computation*, 9(3): 223–248.

Dung, P. M. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2): 321–358.

Dvorák, W.; and Dunne, P. E. 2017. Computational Problems in Formal Argumentation and their Complexity. *FLAP*, 4(8).

Gaggl, S. A.; Linsbichler, T.; Maratea, M.; and Woltran, S. 2020. Design and results of the Second International Competition on Computational Models of Argumentation. *Artificial Intelligence*, 279: 103193.

Gao, Y.; and Toni, F. 2014. Argumentation Accelerated Reinforcement Learning for Cooperative Multi-Agent Systems. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 333–338. IOS Press.

Georgila, K.; and Traum, D. R. 2011. Reinforcement Learning of Argumentation Dialogue Policies in Negotiation. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, 2073–2076. ISCA.

Joshi, C. K.; Cappart, Q.; Rousseau, L.; Laurent, T.; and Bresson, X. 2020. Learning TSP Requires Rethinking Generalization. *CoRR*, abs/2006.07054.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations)*.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Kuhlmann, I.; and Thimm, M. 2019. Using Graph Convolutional Networks for Approximate Reasoning with Abstract Argumentation Frameworks: A Feasibility Study. In *Proceedings of the 13th international conference on Scalable Uncertainty Management (SUM)*, 24–37.

Malmqvist, L.; Yuan, T.; Nightingale, P.; and Manandhar, S. 2020. Determining the Acceptability of Abstract Arguments with Graph Convolutional Networks. In *SAFA@ COMMA*, 47–56.

McKay, B. D.; and Piperno, A. 2014. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60: 94–112.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nat.*, 518(7540): 529–533.

Niskanen, A.; and Järvisalo, M. 2020a. μ-toksia: An Efficient Abstract Argumentation Reasoner. In Calvanese, D.; Erdem, E.; and Thielscher, M., eds., *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, 800–804.

Niskanen, A.; and Järvisalo, M. 2020b. Strong Refinements for Hard Problems in Argumentation Dynamics. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and Lang, J., eds., *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 841–848. IOS Press.

Niskanen, A.; Wallner, J. P.; and Järvisalo, M. 2016. Optimal Status Enforcement in Abstract Argumentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 1216–1222.

Niskanen, A.; Wallner, J. P.; and Järvisalo, M. 2016. Pakota: A System for Enforcement in Abstract Argumentation. In Michael, L.; and Kakas, A. C., eds., *Proceedings of the 15th European Conference on Logics in Artificial Intelligence (JELIA 2016)*, volume 10021 of *Lecture Notes in Computer Science*, 385–4000. Springer.

Niskanen, A.; Wallner, J. P.; and Järvisalo, M. 2018. Extension Enforcement under Grounded Semantics in Abstract Argumentation. In Thielscher, M.; Toni, F.; and Wolter, F., eds., *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018)*, 178–183. AAAI Press.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press. ISBN 978-0-262-19398-6.

Wallner, J. P.; Niskanen, A.; and Järvisalo, M. 2017. Complexity Results and Algorithms for Extension Enforcement in Abstract Argumentation. *J. Artif. Intell. Res.*, 60: 1–40.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.