# Policy generalisation in reinforcement learning for abstract argumentation

**Sultan Alahmari**[1] and **Tommy Yuan** [2] and **Daniel Kudenko**[3]

**Abstract.** Policy generalisation is an important attribute for an argumentative learning agent to apply the learned solutions to different environments. Learning agent needs to know the specific argumentation patterns which can help to identify optimal argument in different argumentation graphs. This paper demonstrates some difficulties in identifying patterns for learning in abstract argumentation systems. We propose to look into the internal structure of the arguments in order to facilitate the identification of useful argument patterns.

## 1 Introduction

Argumentation is a type of communication between agents with the purpose of reaching an agreement on what to believe [14]. There has been increasing research in agent argumentation over the past decade [18]. For an agent to be an effective dialogue participant, it needs to have a set of dialogue strategies in order to make high quality dialogue contributions. By reviewing the literature in computerised dialogue systems, such as [19] and [16], it was noted that the dialogue strategies for most implemented systems are hardwired into the agent [2]. However, given the dynamic nature of argumentation, pne problem with the hardwired strategy is that because the heuristics are fixed, it is not possible to refine or extend the dialogue strategy especially when dealing with newly arising dialogue situations. One way to address this is to make the agent search for the optimal strategies based on each situation, for instance using trial and error, the agent with the best strategy wins the dialogue [6].

Machine learning is believed to be able to meet this challenge [1] since it is flexible for an agent to learn dialogue strategies through past experiences. In addition, learning makes an agent easier to adapt to not only a deterministic environment but also a stochastic environment [1]. A common approach for machine learning in an agent context is reinforcement learning (RL) [10]. RL maps each state with an action by interacting with the environment, the agent can then learn what to do and how to connect a different situation with an action in order to maximise the cumulative reward [10]. However, the agent does not know which action to take initially so it needs to explore all actions by randomly trying them out. Figure 1 illustrates how reinforcement learning works. Some work in the literature combines reinforcement learning with dialogues, for instance [5]. Their focus, however, is on negotiation as opposed to persuasive argumentation, which is a different kind of dialogue [12]. Our chief interest is reinforcement learning for argumentation.

In [1] and [2], we present the ***ARGUMENTO+*** system, named after its predecessor *ARGUMENTO* as reported in [19]. ***ARGUMENTO+*** allows an RL agent to play an argument game against different baseline agents. The result is promising when an agent learns and plays in the same argumentation graph. It would be ideal if the knowledge learned from one argument graph could be applied to a different argument graph. This technique is called policy generalisation in the area of reinforcement learning. The key challenge here is to identify state action patterns in abstract argumentation that can be effectively applied to different argumentation graphs. [11].

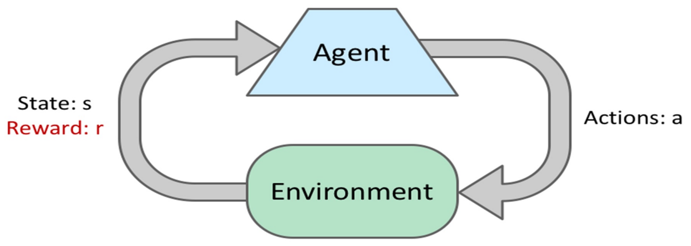

**Figure 1.** Reinforcement learning

The aim of this paper is to report our ongoing work in policy generalisation. The remainder of this paper is organised as follows. We first introduce the work done so far, we then propose a generalisation approach and discuss the result. Finally, we discuss our intended future work in this area.

## 2 Argumento+

We have built a reinforcement learning argumentation test-bed, ***ARGUMENTO+***, using the Java programming language. An Abstract Argumentation Framework [4] is used to represent the argumentation process. The argument game presented in [13] was adapted for reasons of simplicity and flexibility. The details of the argument game are as follows:

The argument game can be represented as a tuple of: G=<A, D, R, P> where: A is the argumentation system, D is the dialogue history which contains a set of moves made by the players, R is the set of rules that players need to follow when making a move, P is the set of players, normally 2 denoted as 0, 1. In [13], Wooldridge defines six rules that each participant must follow in a simple argument game and they are:

1. First move in D is made by player0 e.g. $P_0 = 0$

[1] University of York, UK, email: smsa500@york.ac.uk
[2] University of York, UK, email: tommy.yuan@york.ac.uk
[3] University of York, UK, email: daniel.kudenko@york.ac.uk

2. Players take turns making moves (one move per turn). $P_i = P_{imod2}$.
3. Players cannot repeat a move $\forall a_i, a_j \in D, a_i \neq a_j$.
4. Each move must attack (defeat) the previous move $a_i \rightarrow a_{i-1}$
5. The game is ended if no further moves are possible $\forall a_i \in A \wedge \notin D, a_i a_n$
6. The winner of the game is the player that makes the final move $G_{winner} \doteq P_{nmod2}$

In ***ARGUMENTO+***, the learning agent adopts the most commonly used reinforcement learning method, the Q-Learning algorithm. The formula for the Q-learning algorithm is: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, A_t) - Q(S_t, A_t)]$ The aim of this algorithm is to make an agent learn from experience and map each state with an action by choosing the maximum value from the Q-table, which is updated after each episode (episode is a number of repeated game between players, each episode is one game). To enable Q-learning, we need to identify state, action and reward function.

The state representation in the literature (e.g.[14];[3]) is adapted where states are nodes in the argumentation graph and actions are the attack relation between arguments. The aim of reinforcement learning is to allow the agent to learn how to act in the environment to maximise the long term cumulative reward, and to explore the optimal actions for each state to achieve the agents goal. It is supported in [15] that learning will occur iteratively and through a trial-and-error method, depending on the experience of interaction between the agent and the environment and the reward it received.

In this research, the reward for the agent is designed as the number of acceptable arguments in the grounded extension. The reason for the adoption is that the grounded extension contains a set of acceptable argument that have been put forward by the dialogue participants, and each individual agent wishes to maximise the acceptability of their own arguments in each episode [1].

After performing an initial experiment to investigate whether the learning agent can learn to argue against the baseline agents [1]; [2], we found that it was generally encouraging to apply reinforcement learning to argumentation. However, we discovered issues with state representation, where a state is defined as the argument itself in the argumentation graph. An argument sometimes appears in different dispute lines and therefore cannot represent a unique dialogue state. As a result of the confusion over state representation, a learning agent picking an argument with a high value may sometimes lose the game. This issue has an negative impact on the agent's performance. To tackle this issue, we proposed and experimented with a more sophisticated state representation, that is (levelOfTree, agentID, currentState, previousState). The results are promising and it can clearly be seen that the learning agents perform better against different baseline agents as demonstrated in Figure 2, 3 and 4 where the performance of the learning agent is in green and the baseline agent in blue.

The agent so far learns and performs in the same argumentation graph. When facing a new argument graph, the agent has to learn from scratch. It would be ideal if the learning agent could transfer what has been learned in one graph to a different argument graph. This relates to RL policy generalisation which will be discussed next.

## 3 Policy generalisation for abstract argumentation

Policy generalisation intends to generalise the policy that have been learned by a RL agent. As a result, a leaned policy that has been learned from one argument graph should be able to applied to a different argument graph. One solution is to identify possible argument patterns (e.g. state-action pairs) that can be effectively applied across a range of different argument graphs.

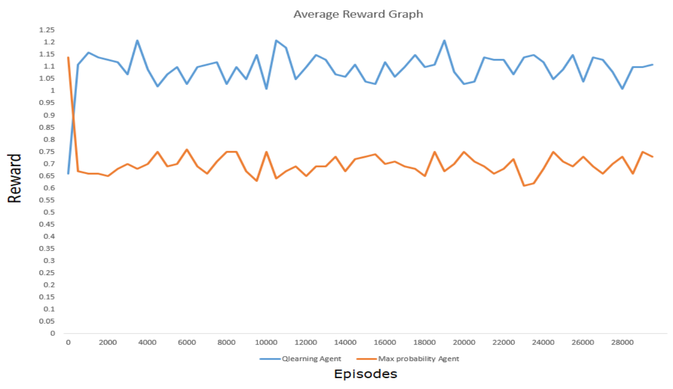

**Figure 2.** RL agent against Max-Probability agent

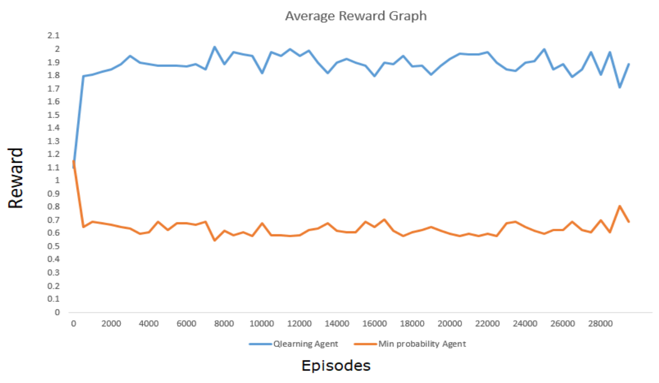

**Figure 3.** RL agent against Min-Probability agent

Since we are dealing with abstract argumentation system where only the arguments and the attacking relations are known, this leads us to looking into the attacking relations between the arguments which might form useful features for argument representation. We propose to take the feature of the number of attackers and the number of immediately winning attackers for use to represent an argument action. As an example shown in Figure 5, argument C, D has zero attackers, argument B has one immediately winning attacker and argument a has two attackers and one immediately winning attacker.

Number of attackers provides the number of possibilities that an argument can be attacked. Number of immediately winning attackers provides the number of immediately successful attackers. A further feature (currently named as category) can be derived by using the formula of (number of immediately winning attackers)/(number of attackers). This number provides a short term view on the proportion of the winning attackers. The value for category ranges from 0 to 1, it can therefore be further classified into different intervals {0, (0,0.25], (0.25,0.5), 0.5, (0.5,0.75], (0.75,1), 1}. The smaller the number is, the argument is likely win from a short term of view. As a result, the categories might be qualified as definite win, high likely win, likely

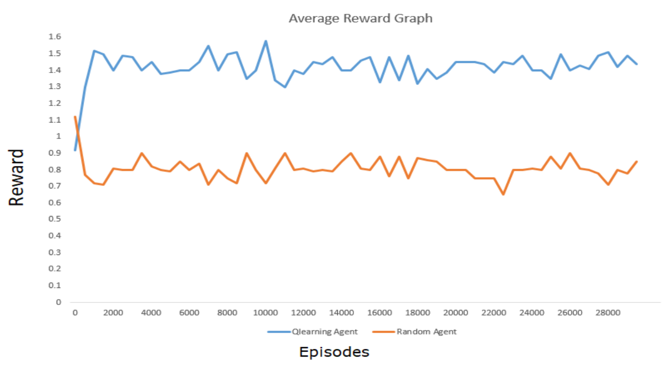

**Figure 4.** RL agent against Random agent

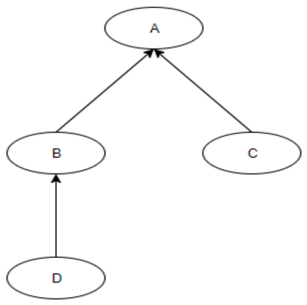

**Figure 5.** Argumentation graph

win, maybe win, unlikely win, high unlikely win, definitely lose from a short term view.

Number of attackers and category have been applied in representing argument actions and implemented in ***ARGUMENTO+***. To make the state representation unique for the current game, we use the following state representation: (depthOfTree, Argument, Category, NumOfAttackers). Two Q-tables have been maintained: one for the current argument game and the other is general that can be used by other argumentation graphs. After finishing each argument game we transfer the values to the general Q-table which contains only (Category, NumOfAttackers). However, if two arguments in the current game have the same category and number of attackers it will take an average of these two values then transfer that to the general Q-table.

In order to evaluate whether the generalisation method works, we needed to identify a data set in order to test the agent's performance. Initially, we tested three different graphs and found that the policy converged in episode 50 in all three, but there was a problem because the performance was not stable. We suspect that a big data set is needed in order to achieve the stability. Therefore, we randomly generated 50 different graphs, they are fully connected with a number of nodes between 5 to 10. To ensure a uniform choice, we chose Leave One Out Cross Validation in 50 graphs, and take an average at the end.

We ran the experiment over 50 games with each game having 50 graphs. The agent was trained on 49 graphs and then tested with the 50th graph. We decided to encourage our learning agent to take two different approaches based on the number of arguments in the grounded extension. We examined whether the RL agent was interested in winning the game with the minimum or maximum number of arguments. Indeed the reward shaping needs to be modified, as in the flowchart in Figure 6. The experiment also envisages to have one agent with knowledge and one without. After 50 games, an average of the rewards after every 5 episodes was taken as shown in Figures 7 and 8.

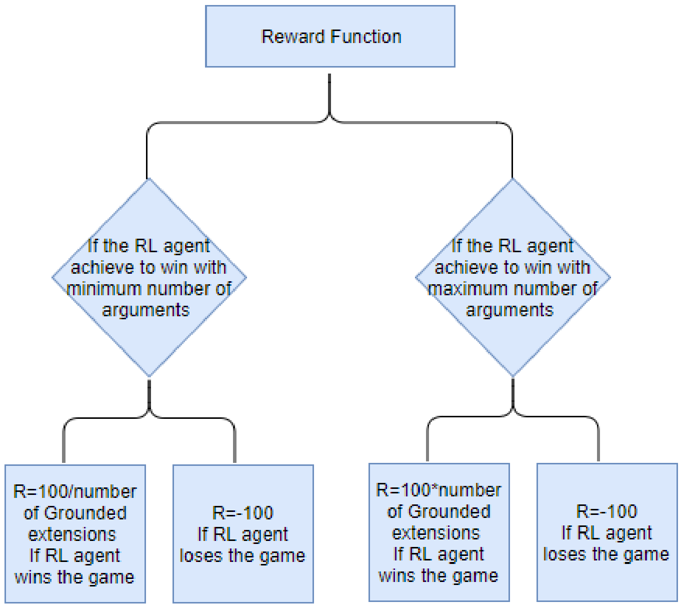
Reward Function
If the RL agent achieve to win with minimum number of arguments
If the RL agent achieve to win with maximum number of arguments
R=100/number of Grounded extensions If RL agent wins the game
R=-100 If RL agent loses the game
R=100*number of Grounded extensions If RL agent wins the game
R=-100 If RL agent loses the game

**Figure 6.** Reward shaping

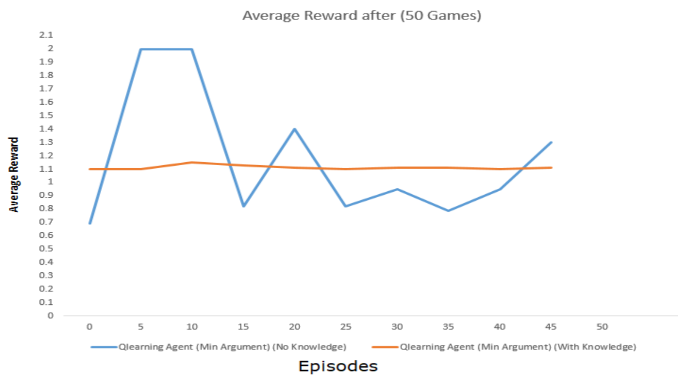

**Figure 7.** Cross validation for RL agent with and without knowledge with minimum numbers of arguments

In the first few episodes of both cases, the learning agent (in red line) demonstrates some advantage of the learnt knowledge but soon the advantage was overtaken by the agent learning from scratch (in blue line) most of the time. This is an unexpected result though the usefulness of the learned knowledge is merely demonstrated at the

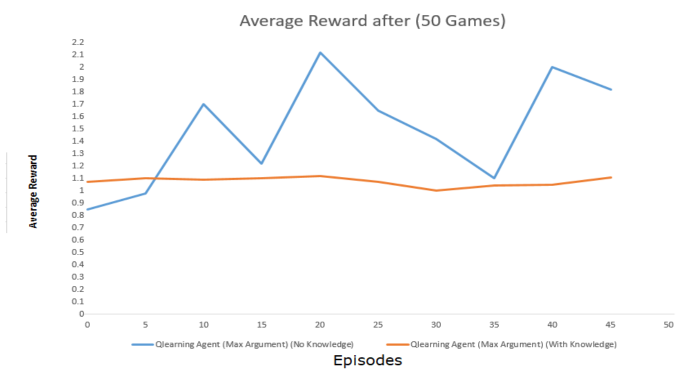

**Figure 8.** Cross validation for RL agent with and without knowledge with maximum numbers of arguments

start. By comparing both cases in Figures 7 and 8, the learning performs better when trying to win with minimal number of arguments. By inspecting the Q-tables, the only consistent finding is that the arguments with zero number attackers attract highest value in the winning with minimal number of argument scenario. On reflection of the experimental result, the argumentation graph in Figure 9 is used an an example to facilitate the analysis. A uniformed distribution is assumed where the winning possibility of an argument is 50/50 For example,the current state for the learning agent is argument 'A' and the agent needs to decide which argument to choose from 'B' or 'C' or 'X'. In our proposal, the agent can see the next level of the tree arguments 'D', 'E' and 'F' respectively. Therefore the chance of winning for argument 'B', 'C', 'X' are 0.25, 0.5 and 1 respectively.

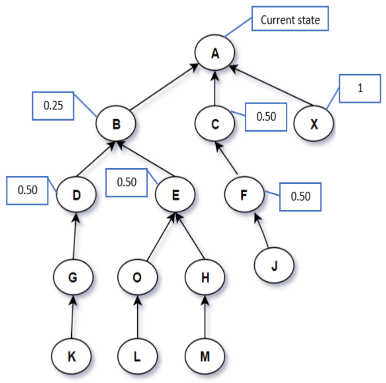

**Figure 9.** Argumentation graph with possibility of winning

We would normally expect the argument with a lower number of attackers to do better. This is the case for argument 'X' (with value 1) in a winning with the minimum number of arguments scenario. However when the learning agent is looking to maximise the number of grounded extensions, 'C' (with value 0.5) is the best choice. A further example can be seen from the argumentation graph in Figure 10. Although argument 'Q'(with value 0.5) has higher winning possibility value than 'R' (with value 0.125), the learning agent will choose R (with a lower value) because it is higher long term reward. We believe that this is the main reason why the learning agent cannot identify useful patterns to generalise the policy for different graphs.

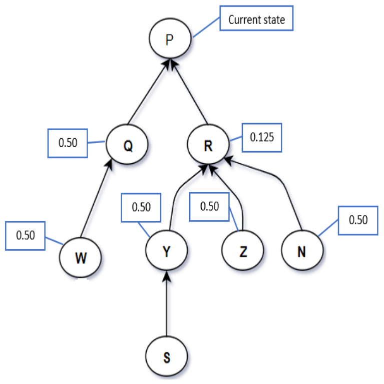

**Figure 10.** Different scenario of argumentation graph

## 4 Conclusions and Future work

We have designed an RL agent for abstract argumentation and the agent performed well in a single argument graph. We also reached the conclusion that in abstract argumentation it is hard to capture useful argument patterns that can be reused in different argument graphs. It might be sensible to move from the abstract argumentation to proposition-based argumentation where the internal structure of an argument is considered.

There are many dialogue games in the area of informal logic and computational dialectics that are operated at the propositional level (e.g. [12]; [8]; [17]). Informal logic dialogue games possess rich features in dialogue states and ample room for strategic formation where various argument patterns can be identified.

The first step of investigation could be to study the representation of goal, state, actions and reward functions for such dialogue so that reinforcement learning can effectively take place. For a persuasive dialogue the dialogue goal can be specified as converting each others' view point. Dialogue history, commitment stores and agent knowledge base contribute to the formulation of the dialogue state which should provide sufficient information for an agent to make decision for an action.

Reward functions for dialogue games are complicated to design. Ground extensions, which we have been used successfully in abstract argument games, can be applied here, with extra facilities to transform the pool of proposition-based commitments to abstract argument systems. ASPIC+ by Prakken and Modgil [7] and its implementation - TOAST [9] by Reed and Snaith will be useful here. Further reward functions can also be explored in order to capture the naturalness of a dialogue e.g. argument flows. Upon the successful learning of a single argumentation topic, the learned policies can be tested on a different topic or even a different game to see whether it is general.

## ACKNOWLEDGEMENTS

We would like to thank the referees for their comments which helped improve this paper.

## REFERENCES

[1] Sultan Alahmari, Tommy Yuan, and Daniel Kudenko, 'Reinforcement learning for abstract argumentation: Q-learning approach', in *Adaptive and Learning Agents workshop (at AAMAS 2017)*, (2017).

[2] Sultan Alahmari, Tommy Yuan, and Daniel Kudenko, 'Reinforcement learning for argumentation: Describing a phd research', in *Proceedings of the 17th Workshop on Computational Models of Natural Argument (CMNA17)*, (2017).

[3] Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon, 'Strategic dialogue management via deep reinforcement learning', *arXiv preprint arXiv:1511.08099*, (2015).

[4] Phan Minh Dung, 'On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artificial intelligence*, **77**(2), 321–357, (1995).

[5] Takuya Hiraoka, Kallirroi Georgila, Elnaz Nouri, David Traum, and Satoshi Nakamura, 'Reinforcement learning in multi-party trading dialog', in *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, p. 32, (2015).

[6] Piotr S Kośmicki, 'A platform for the evaluation of automated argumentation strategies', in *International Conference on Rough Sets and Current Trends in Computing*, pp. 494–503. Springer, (2010).

[7] Sanjay Modgil and Henry Prakken, 'The aspic+ framework for structured argumentation: a tutorial', *Argument & Computation*, **5**(1), 31–62, (2014).

[8] Henry Prakken, 'Formal systems for persuasion dialogue', *The knowledge engineering review*, **21**(2), 163–188, (2006).

[9] Mark Snaith and Chris Reed, 'Toast: Online aspic+ implementation.', *COMMA*, **245**, 509–510, (2012).

[10] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, volume 1, MIT press Cambridge, 1998.

[11] Matthew E Taylor and Peter Stone, 'Transfer learning for reinforcement learning domains: A survey', *Journal of Machine Learning Research*, **10**(Jul), 1633–1685, (2009).

[12] Douglas Walton and Erik CW Krabbe. Commitment in dialogue, 1995.

[13] Michael Wooldridge, *An introduction to multiagent systems*, John Wiley & Sons, 2002.

[14] Michael Wooldridge, *An introduction to multiagent systems*, John Wiley & Sons, 2009.

[15] Erfu Yang and Dongbing Gu, 'Multiagent reinforcement learning for multi-robot systems: A survey', Technical report, tech. rep, (2004).

[16] Tangming Yuan, David Moore, and Alec Grierson, 'A human-computer dialogue system for educational debate: A computational dialectics approach', *International Journal of Artificial Intelligence in Education*, **18**(1), 3–26, (2008).

[17] Tangming Yuan, David Moore, Chris Reed, Andrew Ravenscroft, and Nicolas Maudet, 'Informal logic dialogue games in human–computer dialogue', *The Knowledge Engineering Review*, **26**(2), 159–174, (2011).

[18] Tangming Yuan, Jenny Schulze, Joseph Devereux, and Chris Reed, 'Towards an arguing agents competition: Building on argumento', in *Proceedings of IJCAI2008 Workshop on Computational Models of Natural Argument*, (2008).

[19] Tangming Yuan, Viar Svansson, David Moore, and Alec Grierson, 'A computer game for abstract argumentation', in *Proceedings of the 7th Workshop on Computational Models of Natural Argument (CMNA07)*, (2007).