ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-333

Argumentation Accelerated Reinforcement Learning for Cooperative Multi-Agent Systems

Yang Gao and **Francesca Toni** [1]

Abstract. Multi-Agent Learning is a complex problem, especially in real-time systems. We address this problem by introducing Argumentation Accelerated Reinforcement Learning (AARL), which provides a methodology for defining heuristics, represented by arguments, and incorporates these heuristics into Reinforcement Learning (RL) by using reward shaping. We define AARL via argumentation and prove that it can coordinate independent cooperative agents that have a shared goal but need to perform different actions. We test AARL empirically in a popular RL testbed, RoboCup Takeaway, and show that it significantly improves upon standard RL.

1 Introduction

Learning to coordinate in cooperative multi-agent systems (MAS) is recognised as a complex problem and has attracted much attention [2,8,10,13]. In this context, coordination is defined as *'the ability of two or more agents to jointly reach a consensus over which actions to perform in an environment'* [12]. Argumentation [4], studying the concept of "good" arguments among conflicting arguments, is widely viewed as a powerful tool in solving conflicts and reaching agreement (e.g. [5]). In this paper, we investigate the use of argumentation to coordinate multiple independent learning agents.

We focus on *Reinforcement Learning* (RL), because it allows agents to learn by interacting with the environment and has been shown to be a generic and robust learning algorithm to achieve coordinated behaviours [18]. However, RL may converge slowly in cooperative MAS, mainly because of the huge joint action space which is exponential in the number of agents [2]. *Potential-based reward shaping* [16] has been used to improve performance of RL in cooperative MAS (see, e.g., [3]), but its effectiveness heavily relies on the quality of the heuristics this technique is deployed with. Obtaining high-quality heuristics for RL is challenging in cooperative MAS, not only because the domain knowledge given by domain experts can be error-prone or even self-conflicting, but also because heuristics are required to instruct individual agents as well as to coordinate multiple independent agents. We propose a methodology, based on *value-based argumentation frameworks* (VAFs) [1], to tackle this problem. Compared with existing research in integrating argumentation into RL [7], our research focuses on complex *multi-agent* problems, and provides more generic techniques in proposing heuristics. We prove that our methodology recommends different actions to different agents so that, when making decisions, an agent only needs to know its teammates' arguments and does not need to explore the joint action space. We then use this VAF-based methodology to generate high-quality heuristics that we incorporate into RL using potential-based reward shaping. The resulting *Argumentation Accelerated RL* (AARL) is, to the best of our knowledge, the first generic algorithm that uses argumentation to aid the definition of heuristics and to improve the performance of RL in cooperative MAS. We empirically show the effectiveness of AARL in the RoboCup Soccer Takeaway game, a widely used real-time testbed for MAS [19], interesting as it takes place in a continuous space.

[1] Imperial College London, UK, email: {yg211,ft}@imperial.ac.uk

2 Background

First we give fundamentals of abstract and value-based argumentation. Then we describe RL, followed by an introduction to potential-based reward shaping, by means of which we integrate argumentation-based heuristics into RL. Finally, we describe the RoboCup Soccer Takeaway game, which we use as a testbed.

2.1 Argumentation Frameworks

An *abstract argumentation framework* (AF) [4] is a pair (Arg, Att) where Arg is a set of *arguments* and $Att \subseteq Arg \times Arg$ is a binary relation ($(A, B) \in Att$ is read 'A *attacks* B'). Suppose $S \subseteq Arg$ and $B \in Arg$. S *attacks* B iff some member of S attacks B. S is *conflict-free* iff S attacks none of its members. S *defends* B iff S attacks all arguments attacking B. Semantics of AFs are defined as sets of "rationally acceptable" arguments, known as *extensions*. For example, given some $\mathsf{F} = (Arg, Att)$, $S \subseteq Arg$ is an *admissible extension* for F iff S is conflict-free and defends all its elements; S is a *complete extension* for F iff S is conflict-free and $S = \{a | S \text{ defends } a\}$; S is the *grounded extension* for F iff S is minimally (wrt. $\subseteq$) complete for F. The (possibly empty) grounded extension is guaranteed to be unique [4], consisting solely of the uncontroversial arguments and being thus "sceptical". For example, consider two arguments:

a: Let's have dinner at home today
b: Let's have dinner in a restaurant today

Sceptically, we may think that neither is acceptable since neither of them is convincingly good. Formally, we can build an AF (Arg_T, Att_T), where $Arg_T = \{a, b\}$ and $Att_T = \{(a, b), (b, a)\}$. This AF is depicted as a directed graph in Fig. 1(a). $\{a, b\}$ is not admissible because it is not conflict-free. $\{a\}$ ($\{b\}$) is admissible because it is conflict-free and it can defend itself. The grounded extension is $\emptyset$, consistent with our intuition that neither is convincing.

In some contexts, the attack relation between arguments is not enough to decide what is "rationally acceptable", and the "values" promoted by arguments must be considered. *Value-based argumentation frameworks* (VAFs) [1] incorporate values and preferences over them into AFs. The key idea is to allow for attacks to succeed or fail, depending on the relative worth of the values promoted by the competing arguments. Given a set V of values, an *audience Valpref* is a strict partial order over V (corresponding to the

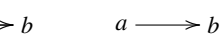


Figure 1: (a) An argumentation framework and (b) its simplification. The argument in the tail of an arrow attacks the argument in the head. A double-sided arrow stands for mutual attack.

preferences of an agent), and an *audience-specific VAF* is a tuple $(Arg, Att, V, val, Valpref)$, where (Arg, Att) is an AF and $val : Arg \rightarrow V$ gives the values promoted by arguments. In VAF, the ordering over values, *Valpref*, is taken into account in the definition of extensions. The *simplification* of an audience-specific VAF is the AF (Arg, Att^-), where $(A, B) \in Att^-$ iff $(A, B) \in Att$ and $val(B)$ is not higher than $val(A)$ in *Valpref*. $(A, B) \in Att^-$ is read '*A defeats B*'. Then, (acceptable) extensions of a VAF are defined as (acceptable) extensions of its simplification (Arg, Att^-). We refer to (Arg, Att^-) as the *simplified AF derived from* $(Arg, Att, V, val, Valpref)$. For example, extend our earlier illustrative AF with the following two values:

$v1$: Cheap $v2$: Time-saving

Let val_T be such that $val_T(a) = v1$, $val_T(b) = v2$ and let $Valpref_T$ give $v1 >_v v2$.[2] Then we obtain a VAF $(Arg_T, Att_T, V_T, val_T, Valpref_T)$ and derive the simplified AF (Arg_T, Att_T^-), where $Att_T^- = \{(a, b)\}$, as shown in Fig. 1(b). The grounded extension for (Arg_T, Att_T^-) is $\{a\}$. This can be interpreted as follows: if we think cheap is of higher importance than time-saving, we will choose to eat at home. Thus, VAF is a powerful tool for determining the "rationally acceptable" arguments with conflicting domain knowledge.

2.2 Markov Decision Process (MDP)

MDP is one of the most widely used RL models [21]. A MDP is a tuple (S, A, T, R), where S is the *state space*, A is the *action space*, $T(s, a, s') = Pr(s'|s, a)$ is the *transition probability* of moving from state s to state s' by executing action a, and $R(s, a, s')$ gives the immediate *reward* received when action a is taken in state s, moving to state s'. The goal of planning in a MDP is to find a policy $\pi : S \rightarrow A$, specifying for each state the action to take, which maximises the expected future rewards. In many real problems, the transition probabilities and the reward functions are not known. In these cases, *temporal difference* updates [20] are used to propagate information about values of states, $V(s)$, or state-action pairs, $Q(s, a)$.

SARSA(λ) with eligibility traces [21], as shown in Algorithm 1, is a popular temporal difference RL algorithm and has been widely used in cooperative MAS [9, 19]. In Algorithm 1, α is a learning rate parameter and γ is a discount factor governing the weight placed on the future. e represents *eligibility traces*, which store the credit that previous action choices should receive for current rewards, while λ governs how much credit is delivered back to them. The policy used in line 5 and line 8 is *ϵ-greedy*: the action with highest $Q(s, a)$ value will be selected for a proportion $1 - \epsilon$; for the other ϵ proportion, actions will be selected randomly.

2.3 Potential-Based Reward Shaping

Potential-based reward shaping was proposed by Ng et al. [16] as the difference of some potential function Φ over the current state s and the following state s'. Wiewiora et al. [22] extended the potential-based method to the case of shaping functions based on both states and actions: $\Phi(s, a)$. In particular, Wiewiora et al. proposed *look-back advice* for incorporating *action-based* domain knowledge into

[2] Given two values V_1 and V_2, $V_1 >_v V_2$ stands for 'V_1 is preferred to V_2 in *Valpref*', and $V_1 =_v V_2$ stands for $V_1 >_v V_2 \wedge V_2 >_v V_1$.

Algorithm 1 SARSA(λ) with replacing eligibility traces

1: Initialise $Q(s, a)$ arbitrarily for all states s and actions a
2: **for** each episode **do**
3: Initialise $e(s, a) = 0$ for all s and a
4: Initialise current state s_t
5: Choose action a_t from s_t using the policy derived from Q
6: **while** s_t is not a terminal state **do**
7: Execute action a_t, observe reward r_t and new state s_{t+1}
8: Choose a_{t+1} from s_{t+1} using the policy derived from Q
9: $\delta \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$
10: $e(s_t, a_t) \leftarrow 1$
11: **for** all s and a **do**
12: $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
13: $e(s, a) \leftarrow \gamma\lambda e(s, a)$
14: **end for**
15: $s_t \leftarrow s_{t+1}$; $a_t \leftarrow a_{t+1}$
16: **end while**
17: **end for**

RL. When integrating look-back advice into Alg. 1, line 9 becomes

$$\delta \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) + F(s_{t-1}, a_{t-1}, s_t, a_t)$$

where

$$F(s_{t-1}, a_{t-1}, s_t, a_t) = \Phi(s_t, a_t) - \gamma^{-1}\Phi(s_{t-1}, a_{t-1})$$

is obtained when moving from state s_{t-1} to s_t by action a_{t-1}. Although it has been empirically shown that look-back advised RL can converge regardless of Φ values, the convergence speed still heavily relies on the values of Φ [14]. Potential values can be viewed as numerical representatives of the heuristics, which can be difficult to obtain, especially when the domain knowledge is conflicting. In Section 4 we will show that, by using argumentation, high-quality heuristics can be extracted from conflicting domain knowledge, and these heuristics can be naturally represented by Φ values.

2.4 RoboCup Soccer Takeaway Game

The *Takeaway* game is proposed [11] to facilitate RL research in the context of RoboCup Soccer. In N-Takeaway ($N \in \mathbb{N}$, $N \geq 1$), $N + 1$ hand-coded *keepers* are competing with N independent learning *takers* on a fixed-size field. Keepers attempt to keep possession of the ball, whereas takers attempt to win possession of the ball. The game consists of a series of *episodes*, and an episode ends when the ball goes off the field or any taker gets the ball. A new episode starts immediately with all the players reset.

To facilitate RL in the RoboCup Soccer, *macro actions* were proposed by Stone et al. [19] and then adjusted for Takeaway [11]. In Takeaway, there are two macro actions:

- **TackleBall()**: move directly towards the ball to tackle it
- **MarkKeeper**(i): go to mark keeper K_i, $i \neq 1$

where K_i represents the ith closest keeper to the ball (so that K_1 is the keeper in possession of the ball). When a taker marks a keeper, the taker blocks the path between the ball and that keeper. Thus, a taker is not allowed to mark the ball holder, and the action set in N-Takeaway consists of $M{=}N{+}1$ actions.

Each taker's observation of its environment is represented by a *state vector*, whose elements, known as *state variables*, are listed in Table 1. Most existing research on Takeaway assumes that each taker has a 360° vision and uses the *ball-holder-oriented* state variables [3,11], collecting information from the ball holder's perspective. Our state vector, however, not only includes some ball-holder-oriented state variables, but also includes takers' self-oriented state variables in order to facilitate coordination between takers.

State Variable(s)	Description
$dist(K_i,Me), i\in[1,N+1]$	Distance between keepers and self
$dist(T_j,Me)$, $j\in[2,N]$	Distance between other takers and self
$ang(K_i,Me)$, $i\in[2,N+1]$	Angle between the free keepers and self, with vertex at K_1.
$dist(K_i,K_1)$, $i\in[2,N+1]$	Distance between K_1 and the other keepers
$dist(T_j,K_1)$, $j\in[2,N]$	Distance between K_1 and the other takers
$\min_{j\in[1,N]} ang(K_i,T_j)$, $i\in[2,N+1]$	The smallest angle between K_i and the takers with vertex at K_1.

Table 1: State variables in N-Takeaway for learning taker T_1 ($i,j\in\mathbb{N}$). The top three rows give self-oriented variables, the others give ball-holder-oriented variables.

3 Argumentation For Agent Coordination

We consider N-player MAS problems with N cooperative independent learning agents, denoted $Agent_1$, ..., $Agent_N$, with $N\in\mathbb{N}$, $N\geq 2$,[3] where each agent has the same action set $Act=\{a_1,\ldots,a_M\}$, where $M\in\mathbb{N}$, $M\geq 2$, is the number of available actions. The domain knowledge contributing to heuristics is action-based, i.e. recommending action(s) in specific states. We use arguments to represent this knowledge, where an argument A is of the form:

$$con(A) \text{ IF } pre(A)$$

where $con(A)$ (the *conclusion* of A) is the recommended action and $pre(A)$ (the *premise* of A) describes under which conditions argument A is *applicable*. Throughout this section, we make i and j range resp. over agents and actions. An argument A *supports* an action a_j iff $con(A)=a_j$. We denote $Agent_i$'s observation of the current state by Sta_i and that argument A is applicable in Sta_i by $Sta_i \models pre(A)$.

$$Arg^* = \cup_{j=1}^{M} Arg_j^*$$

is the set of all *candidate arguments* s.t. $A\in Arg_j^*$ iff $con(A)=a_j$. So Arg_j^* is the set of arguments supporting action a_j. We assume that each agent is aware of all arguments in Arg^*.

Example 1 (Candidate arguments in Takeaway)**.** Given the macro actions (Section 2.4) in Takeaway and our observation of the game, it is unnecessary for multiple agents to tackle the ball or mark the same keeper because agents can fulfil these tasks individually. So we propose the following domain knowledge for taker T_i:

1. T_i should tackle the ball if T_i is closest to the ball holder;
2. If a keeper is in a quite 'open' position, T_i should mark this keeper;
3. If a keeper is 'far' from all takers, T_i should mark this keeper;
4. If the angle between T_i and a keeper, with vertex at the ball holder, is the smallest, T_i should mark this keeper;
5. If T_i is closest to a keeper, T_i should mark this keeper.

Note that this domain knowledge is action-based. Given the state variables in Table 1, we "translate" the knowledge above into the following five categories of candidate arguments:

1. T_i**TK**: **TackleBall()** IF $i=\arg\min_{1\leq t\leq N} dist(K_1,T_t)$
2. $T_i\mathbf{O}(p)$: **MarkKeeper**(p) IF $\min_{1\leq t\leq N} ang(K_p,T_t)\geq 15$
3. $T_i\mathbf{F}(p)$: **MarkKeeper**(p) IF $\min_{1\leq t\leq N} dist(K_p,T_t)\geq 10$
4. $T_i\mathbf{A}(p)$: **MarkKeeper**(p) IF $i=\arg\min_{1\leq t\leq N} ang(K_p,T_t)$
5. $T_i\mathbf{C}(p)$: **MarkKeeper**(p) IF $i=\arg\min_{1\leq t\leq N} dist(K_p,T_t)$

where $p\in\{2,\cdots,N+1\}$ for arguments referred to as $T_i\mathbf{O}(p)$, $T_i\mathbf{F}(p)$,$T_i\mathbf{A}(p)$ and $T_i\mathbf{C}(p)$, because K_1 cannot be marked. 15 and 10 in items 2 and 3 are threshold values we used to define resp. 'open' and 'far'. Overall, for a N-Takeaway game, there are $4N^2+N$ candidate arguments[4] in Arg^*.

Recall that we face two main issues in extracting "good" heuristics from domain knowledge: (i) domain knowledge may have conflicts, and (ii) since each agent is self-interested, the domain knowledge for each agent may not result in "good" heuristics for the team. To tackle these issues, we define argumentation frameworks as follows:

Definition 1. Given $Sta=\langle Sta_1,\cdots,Sta_N\rangle$, where Sta_i is $Agent_i$'s observation of state s, then a *Sta-specific cooperative argumentation framework* is a tuple $SCAF=(Arg,Att)$ s.t.:
1. $Arg=\langle Arg_1,\cdots,Arg_N\rangle$ s.t. $Arg_i\subseteq Arg^*$ and $A\in Arg_i$ iff $Sta_i\models pre(A)$ (for all i)
2. $Att\subseteq\cup_{i=1}^{N}Arg_i\times\cup_{i=1}^{N}Arg_i$ s.t. $(A,B)\in Att$ iff for some $p,q\in\{1,\cdots,N\}$:
  (i) $con(A)=con(B)$, $A\in Arg_p$, $B\in Arg_q$ and $p\neq q$, or
  (ii) $con(A)\neq con(B)$ and $A,B\in Arg_p$.

We refer to $(\cup_{i=1}^{N}Arg_i,Att)$ as the *AF derived from SCAF*.

The argument set of a *SCAF* is a subset of the candidate argument set Arg^*: for $Agent_i$, only the arguments whose premises are true according to $Agent_i$'s observation of the current state are in Arg_i. By doing this, we ensure that all arguments in a *SCAF* are *applicable*. Attack(s) between two applicable arguments are built iff these two arguments are (i) applicable for different agents but support the same action, or (ii) applicable for the same agent but support different actions. These rules for building attacks are consistent with our intuition that one agent should perform only one action and different agents perform different actions. Note that given these rules for attacks, if argument A attacks B in a *SCAF*, then B also attacks A.

Example 2 (Continuation of Example 1)**.** We build the *SCAF* for *Sta* depicted in Fig. 2(a), First, we choose the applicable arguments. Let us consider T_1's candidate arguments one by one. Because T_1 is the closest taker to the ball, the premise of T_1**TK** is true and this argument is applicable. However, because the angle between K_2/K_3 and T_1, with vertex at K_1, is smaller than 15, K_2/K_3 is not open in this scenario. Therefore, $T_1\mathbf{O}(2)/T_1\mathbf{O}(3)$ is not applicable. Similarly, since neither K_2 nor K_3 are far, $T_1\mathbf{F}(2)$ and $T_1\mathbf{F}(3)$ are not applicable either. The angle between T_1 and K_2/K_3, with vertex at K_1, is smallest among all takers, so $T_1\mathbf{A}(2)$ and $T_1\mathbf{A}(3)$ are applicable in this scenario. The distance between T_1 and K_2 is smaller than the distance between T_2 and K_2, so argument $T_1\mathbf{C}(2)$ is applicable. However, since T_2 is closer to K_3 than T_1, $T_1\mathbf{C}(3)$ is not applicable. Overall, the applicable arguments for T_1 are: $Arg_1=\{T_1\mathbf{TK},T_1\mathbf{A}(2),T_1\mathbf{A}(3),T_1\mathbf{C}(2)\}$. Similarly, we can get the applicable arguments for T_2: $Arg_2=\{T_2\mathbf{C}(3)\}$.

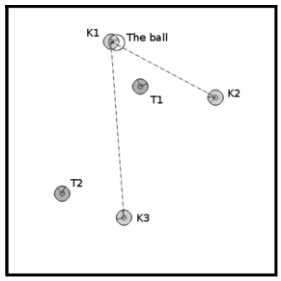
We then build attacks (*Att*) between these applicable arguments. To illustrate, consider T_1**TK** and $T_1\mathbf{A}(2)$: they are both applicable for T_1 but recommend different actions, so they attack each other. Consider also $T_1\mathbf{A}(3)$ and $T_2\mathbf{C}(3)$: they are applicable for different agents but recommend the same action, so they attack each other. The full attack relationship is given in Fig. 2(b).

Below, Arg_{ij} stands for $Arg_i\cap Arg_j^*$. Intuitively, Arg_{ij} are the arguments recommending a_j applicable in $Agent_i$'s observation of the current state. Each Arg_{ij} is "rationally acceptable", as follows:

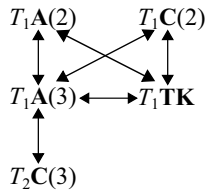
Proposition 1. Let (Arg,Att) be a *SCAF* and the AF derived from it be $F=(\cup_{i=1}^{N}Arg_i,Att)$. Then, Arg_{ij} is an admissible extension for F.

[3] Note that, theoretically, our technique also allows $N=1$.

[4] For taker T_j, T_j**TK** gives one argument and the other four categories of arguments each give N (as there are N free keepers to be marked). So there are $N\times(4\times N+1)$ candidate arguments in total.



(a)



(b)

Figure 2: (a) An example state in 2-Takeaway and (b) its AF.

Proof. By definition of *Att*, Arg_{ij} is conflict-free. Let $A \in Arg_{ij}$ and $B \in Arg - Arg_{ij}$. If $(B, A) \in Att$, then $(A, B) \in Att$ necessarily. So Arg_{ij} can defend all its elements. □

Proposition 1 sanctions that, in a SCAF, all actions supported by applicable arguments are "equally good" for an agent, since their arguments can defend themselves. There may be several such "equally good" actions for an agent, and different agents may have the same "equally good" actions: these situations are not desirable in a cooperative MAS. To address this problem, we introduce *values* into our argumentation frameworks, as shown next.

Definition 2. Given $Sta = \langle Sta_1, \cdots, Sta_N \rangle$ as in Definition 1, a *value-based Sta-specific cooperative argumentation framework* is a tuple $VSCAF = (SCAF, V, val, Valpref)$ s.t.:
1. *SCAF* is a Sta-specific cooperative argumentation framework
2. *V* is a set (of *values*)
3. $val : Arg^* \rightarrow V$ is a function from Arg^* to *V*
4. *Valpref* is a strict partial order over *V*
We denote $val(A) = v$, for $A \in Arg^*$, as $A \mapsto v$, and say that A *promotes* v. If $(\cup_{i=1}^{N} Arg_i, Att)$ is the AF derived from *SCAF*, then we call $(\cup_{i=1}^{N} Arg_i, Att, V, val, Valpref)$ the *VAF derived from VSCAF*.

Note that, as in standard VAFs, each argument can only promote one value, whereas each value can be promoted by several arguments. We assume that agents share the same value preference (*Valpref*), in line with our assumption that agents are cooperative. As in standard VAFs (see Section 2.1), a simplified AF can be derived from the VAF derived from a VSCAF. We use

$$AF^{-} = (\cup_{i=1}^{N} Arg_i, Att^{-})$$

to refer to this simplified AF derived from the VAF derived from $(SCAF, V, val, Valpref)$ (with $SCAF = (Arg, Att)$).

Example 3 (Continuation of Example 2)**.** We add values to *SCAF*. Consider arguments T_i**TK**, for instance. Performing this category of arguments is to prevent the ball holder from holding the ball for too long (value **VT**). Similarly, we give a value for each category of candidate arguments as follows:

1. VT: Prevent the ball being held by the keepers;
2. VO: Prevent the ball being passed to an 'open' keeper;
3. VF: Prevent the ball being passed to a 'far' keeper;
4. VA: Ensure that each pass can be quickly intercepted;
5. VC: Ensure that, after each pass, the ball can be quickly tackled.
The mapping from arguments to values (*val*) is defined as follows: $T_i\mathbf{TK} \mapsto \mathbf{VT}$, $T_i\mathbf{O}(p) \mapsto \mathbf{VO}$, $T_i\mathbf{F}(p) \mapsto \mathbf{VF}$, $T_i\mathbf{A}(p) \mapsto \mathbf{VA}$, $T_i\mathbf{C}(p) \mapsto \mathbf{VC}$. Further, we give the ranking of values (*Valpref*) as follows: $\mathbf{VT} >_v \mathbf{VA} =_v \mathbf{VC} >_v \mathbf{VO} >_v \mathbf{VF}$.[5] Given these rankings of values, we can simplify the AF in Fig. 2(b) and obtain the simplified AF^{-} as illustrated in Fig. 3.

[5] Note that, for simplicity, we assume the same ranking of values throughout the game, but our technique can be applied with value rankings that change over time.

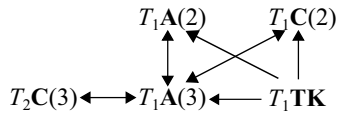


Figure 3: The simplified AF^{-} derived from Fig 2(b).

Lemma 1. If the grounded extension G for AF^{-} is non-empty, then $\exists i, j$ s.t. $G \cap Arg_{ij} \neq \emptyset$.

Proof. $G \cap Arg_{ij} = G \cap Arg_i \cap Arg_j^*$. Since $G \subseteq \cup_{i=1}^{N} Arg_i \subseteq \cup_{j=1}^{M} Arg_j^*$, the lemma trivially holds. □

Theorem 1. If the grounded extension G for AF^{-} is non-empty, then $\forall i$, if $\exists p, q \in \{1, \cdots, M\}$ s.t. $Arg_{ip} \cap G \neq \emptyset$ and $Arg_{iq} \cap G \neq \emptyset$, then $p = q$.

Proof. Necessarily, $\exists A, B \in G$ s.t. $A \in Arg_{ip}, B \in Arg_{iq}$. If $A = B$, then the theorem is obviously true. If $A \neq B$, by contradiction, assume $p \neq q$. Then, by definition of *Att*, $(A, B) \in Att$ and $(B, A) \in Att$. Since the simplification process may only eliminate attacks, (A, B) or (B, A) or both are in Att^{-}.[6] Hence, G is not conflict-free and so not grounded: contradiction. □

Theorem 2. If the grounded extension G for AF^{-} is non-empty, $\forall j$, if $\exists p, q \in \{1, \cdots, N\}$ s.t. $Arg_{pj} \cap G \neq \emptyset$ and $Arg_{qj} \cap G \neq \emptyset$, then $p = q$.

Proof. $\exists A, B$ as in the proof of Theorem 1. Again, if $A = B$, the proof is trivial. If $A \neq B$ but $p \neq q$, (A, B) or (B, A) or both are in Att^{-} which contradicts that G is grounded. □

We have proven that, if there is a non-empty grounded extension for AF^{-}, at least one agent will get a recommended action (Lemma 1), each agent will be recommended at most one action (Theorem 1) and each action will be recommended to at most one agent (Theorem 2). These properties are significant as they guarantee that argumentation can help agents choose what to do (the recommended action) while being cooperative (since no two agents are recommended the same action).

Theorem 3. Let $V_{ach}^{i} = \{v | v \in V, \exists A \in Arg_i \text{ s.t. } A \mapsto v\}$, and $v_{max}^{i} \in V_{ach}^{i}$ s.t. $\forall v \in V_{ach}^{i}, v \not>_v v_{max}^{i}$. For any $i \in \{1, \cdots, N\}$, if $G \cap Arg_i \neq \emptyset$, then $\exists A \in G$ s.t. $A \mapsto v_{max}^{i}$.

Proof. Suppose $G \cap Arg_i \neq \emptyset$. We assume, by contradiction, that $\nexists A \in G \cap Arg_i$ s.t. $A \mapsto v_{max}^{i}$ and $\exists B \in Arg_i - G$ s.t. $B \mapsto v_{max}^{i}$. For any $C \in G \cap Arg_i$, $(B, C) \in Att^{-}$ and $(C, B) \notin Att^{-}$. As a result, G is not admissible and thus not grounded: contradiction. □

Theorem 3 sanctions that for each agent, if it has any recommended action, this action must promote the highest value among all the *achievable values*: values which are promoted by some applicable arguments for this agent. Hence, the recommended action is the best. Note that when the grounded extension for AF^{-} is empty, according to the semantics of grounded extension (see Section 2.1), no "uncontroversially acceptable" arguments can be obtained given the current domain knowledge (argument set). Then additional knowledge should be added; otherwise, no convincing heuristics can be extracted from the argumentation framework.

The grounded extension of AF^{-} (Fig 3) is $\{T_1\mathbf{TK}, T_2\mathbf{C}(3)\}$, satisfying Theorems 1, 2 and 3.

[6] According to the simplification rules of VAF (see Section 2.1), no attacks (if any) between arguments A and B can be eliminated if $val(A) =_v val(B)$.

4 Argumentation Accelerated RL

To encourage agents to perform the actions supported by arguments in the grounded extension, we give these actions positive shaping rewards, by using look-back advice (see Section 2.3). If the grounded extension is empty, then all actions' potential values are 0. For simplicity, we give all the recommended actions the same potential value. Formally, in a state s, given the observation vector *Sta*, its corresponding *SCAF*, *VSCAF* the derived simplified AF^- and its grounded extension G, the potential value function for action a_j is:

$$\Phi(s, a_j) = \begin{cases} 0 & \text{if } \nexists A \in G \text{ s.t. } con(A) = a_j \\ c & \text{otherwise} \end{cases} \quad (1)$$

where $c > 0$ is a constant.

Argumentation Accelerated RL is the integration into RL of argumentation-based heuristics via look-back reward shaping. We can obtain AARL from Alg. 1 by making the following revisions (all line numbers below are the original line numbers in Alg. 1):

- Between line 1 and 2, add *"Initialise Arg*, V, val and Valpref"*. We initialise all these values before the learning starts because they all remain the same throughout the learning process.
- Between line 4 and 5, add *"Initialise $lp = 0$"*, where lp is a variable used to store the potential value in the last learning step.
- Between line 6 and 7, add three steps in the following order:
  - *"Observe Sta_i in s_t, obtain Arg_i"*. Arg_i can be obtained by selecting the applicable arguments (see Def. 1).
  - *"Obtain Arg_q, $q \in \{1, \cdots, N\}, q \neq i$, then build SCAF, VSCAF and derive AF^-"*. To build *SCAF*, each agent needs to know all agents' arguments. If $Agent_i$ can observe other agents' states, it can directly compute other agents' applicable arguments; otherwise, it can obtain the other agents' applicable arguments by communicating with them. Based upon *SCAF*, *VSCAF* and AF^- can be easily built, by their definitions.
  - *"Compute $\Phi(s_t, a)$ for all action a as described in Eq. (1)"*. This requires computing the grounded extension of AF^-.
- Replace line 9 with "$\delta \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) + \Phi(s_t, a_t) - \gamma^{-1} lp$" to implement look-back advice.
- Between line 14 and 15, add *"$lp \leftarrow \Phi(s_t, a_t)$"* to update lp.

Note that when communication is needed, its burden is tractable: as all candidate arguments are known by all agents a priori, agents can just communicate the indexes of arguments. Also, since the *SCAF*, *VSCAF* and AF^- are the same for all agents (see Definitions 1 and 2), these argumentation frameworks' construction and computation can be performed by any agent, and other agents only need to know their recommended actions by communication. This property is especially valuable in applications where computation is more expensive than communication or when agents have heterogeneous computing capabilities. Note that, in AARL, an agent makes decisions based on its own observation of the environment and the arguments of its teammates. Since, for each action, the number of supporting arguments is independent of the total number of agents and actions, the number of arguments supporting a specific action can be viewed as a constant. Hence, for a cooperative MAS problem with N agents and M actions, the complexity of the joint action space is $O(M^N)$ whereas the complexity of the argumentation framework is $O(M \times N)$. So instead of searching the exponential joint action space, agents only need to search the polynomial argumentation framework to coordinate their behaviours. Also note that, if the grounded extension is empty at some learning steps, then AARL behaves like standard RL.

5 Empirical Results for Takeaway

We use Alg. 1 as our standard RL algorithm, with the same parameters as in [3]. The size of the field is 40×40. Each agent receives a reward of 10 when an episode ends and -1 in all other learning steps. Since allowing takers to update actions in each cycle[7] leads to poor performances [11], takers update their action(s) every 5 cycles. The hand-coded strategies of keepers are described in [19], and we design a hand-coded strategy for the takers, s.t. takers who have a recommended action will perform it; those receiving no recommendations will tackle the ball. c in Eq. 1 is 2. The results for 2- and 3-Takeaway are shown in Fig. 4. Since the hand-coded strategies are stable, we give their averaged performance (the horizontal straight lines in Fig. 4). Recall that takers' learning goal is to win possession of the ball more quickly, so the strategy that leads to shorter average episode duration is better. We see that the performance of AARL is constantly better than standard RL, and the standard deviation of AARL is also smaller, suggesting that AARL is more stable. Note that, although AARL and the hand-coded strategy for takers use the same domain knowledge, the former has much better performance. This indicates that AARL is robust to errors in prior knowledge.

Devlin et al. [3] also used look-back advice to integrate domain knowledge into Takeaway, and their performances in 2- and 3-Takeaway (also on a 40×40 field) are shown in Fig. 6a and Fig. 8a in [3], resp. All RL parameters they used are the same with ours[8]. They used three heuristics: *"separation-based shaping"* encourages each agent to take actions that increase its distance to other teammates; *"role-based shaping"* assigns each agent a role (either tackler or marker) a priori and only the tackler is encouraged to tackle; *"combined shaping"* is the integration of these two heuristics. Even though these heuristics successfully improve RL performances in 3-Takeaway (Fig. 8a in [3]), they mislead RL in 2-Takeaway (Fig. 6a in [3]). We believe the reason for these mixed results lies in their lack of systematic methodology to provide heuristics. Instead, AARL allows to integrate domain knowledge into RL while providing a high-level abstraction method (VAFs) for domain experts to propose domain knowledge. Also, the improvements of their heuristically augmented strategies over SARSA(λ) are not as significant as AARL.

6 Related Work

Some research has been devoted to incorporating domain knowledge into RL to improve its performance in MAS. Grzes and Kudenko [9] used high-level STRIPS knowledge in combination with reward shaping to search for an optimal policy and showed that the STRIPS-based reward shaping converges faster than the abstract MDP approach. But their approach requires an explicit goal state and STRIPS-style domain knowledge, unavailable in several applications (e.g. Takeaway). As for cooperative RL, Claus and Boutilier [2] distinguished two forms of multi-agent RL: *independent learners* (ILs), which only consider their own Q-values when choosing actions, and *joint action learners*, which search the exponential joint action space to maximise the sum of all agents' Q-values. Our agents can be seen as ILs. Guestrin et al. [10] used *coordination graphs* to restrain the coordination relationships so that actions are selected to maximise the sum of Q-values of only related agents. Thus, to obtain the Q-values of all related teammates, an agent has to compute all these Q-values or communicate with other agents. Hierarchical RL (HRL) has also been used to guide coordination. For example, Ghavamzadeh et al. [8] proposed *Cooperative HRL*, in which coordination is only

[7] In RoboCup Simulator, the platform receives and executes actions every 100 milliseconds [17], known as a cycle.

[8] However, the state variables used in [3] are slightly different from ours in that they did not use the takers' self-oriented state variables. Also, they use RoboCup Simulator v11.1.0. whereas we use v15.1.0. So their baseline performances are different from ours.

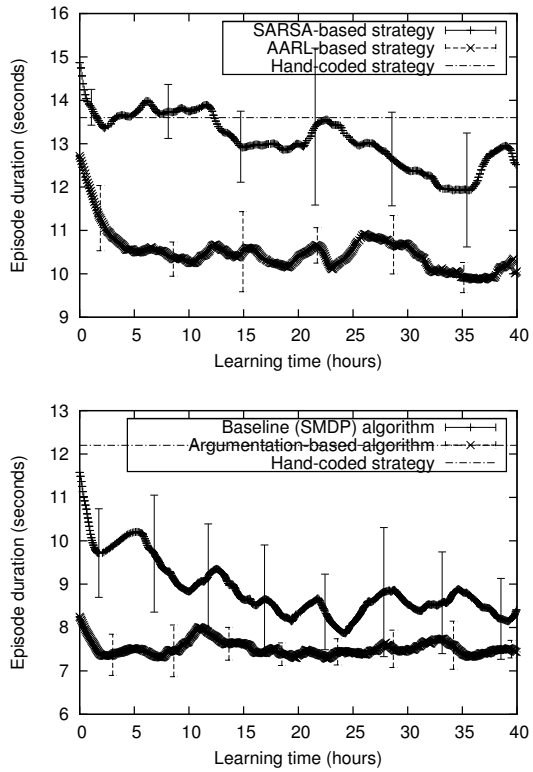


Figure 4: Performances in 2-Takeaway (the upper figure) and (b) 3-Takeaway (the bottom figure). Error bars represent one standard deviation. Results are averaged 30 independent experiments.

learnt in predefined *cooperative subtasks*, defined by domain experts as subtasks where coordination would significantly improve the performance of the whole team. Lau et al. [13] modelled coordination among agents as *coordination constraints* and used these to limit the joint action space for exploration. However, in all these cooperative RL approaches, domain knowledge is in the form of hard constraints and the action exploration is strictly constrained. Hence, learning cannot correct errors in the domain knowledge and the performances are highly sensitive to the quality of the domain knowledge.

As for integration of argumentation and machine learning, most existing works are based on single-agent learning [7, 15]. An exception is our previous work [6], which analysed the emerged performance when both takers and keepers learn. However, it did not give any theoretical analysis, but only presented limited empirical results (only one performance of each strategy was presented).

7 Conclusions

We presented *Argumentation-Accelerated RL* (AARL), a new approach to RL where domain knowledge is represented and organised as an argumentation framework. We proved that, by obtaining the grounded extension of the argumentation framework, each agent is recommended at most one action and each action is recommended to at most one agent and, thus, the agents' behaviours are coordinated. We implemented AARL using the SARSA(λ) algorithm and performed experiments in RoboCup Soccer Takeaway. Empirical results showed that AARL outperforms standard RL and some other state-of-the-art heuristics for Takeaway games. Differently from other works (e.g. [3]), our approach allows users to provide conflicting information and resolves conflicts at run-time.

Our research could be extended in several directions. In the current version of AARL, preferences of values (*Valpref*) and potential values (Φ) are defined a priori and remain the same. Further research is needed to investigate the theoretical possibility of changeable *Valpref* and Φ and the consequent empirical performances. By doing so, domain experts can update their domain knowledge during the learning process and, hence, give "on-line" instructions to the learning agents. Also, since our theoretical results (see Section 3) are independent of any learning algorithm, we believe that our approach can in principle be integrated within other learning algorithms (not limited to RL) or within RL via other techniques (not limited to reward shaping). In addition to trying out our methodology with other learning methods and in other application domains, future work also includes studying how other coordination principles could be represented in argumentation. Finally, we focused on one-agent-one-action problems: we plan to extend our approach to allow each agent to execute multiple actions and multiple agents to play one action.

Acknowledgements

This research was partially supported by the EPSRC TRaDAr project: EP/J020915/1.

REFERENCES

[1] T. Bench-Capon, 'Persuasion in practical argument using value-based argumentation frameworks', *J. Log. Comput.*, **13**(3), 429–448, (2003).
[2] C. Claus and C. Boutilier, 'The dynamics of reinforcement learning in cooperative multiagent systems', in *Proc. of AAAI*, (1998).
[3] S. Devlin, M. Grzes, and D. Kudenko, 'An empirical study of potential-based reward shaping and advice in complex, multi-agent systems', *Advances in Complex Systems*, **14**, 251–278, (2011).
[4] P. M. Dung, 'On the acceptability of arugments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games', *Artificial Intelligence*, **77**(2), 321–357, (1995).
[5] X. Fan and F. Toni, 'Argumentation dialogues for two-agent conflict resolution', in *Proc. of COMMA*, (2012).
[6] Y. Gao and F. Toni, 'Argumentation accelerated reinforcement learning for robucup keepaway-takeaway', in *Proc. of TAFA*, (2013).
[7] Y. Gao, F. Toni, and R. Craven, 'Argumentation-based reinforcement learning for robocup soccer keepaway', in *Proc. of ECAI*, (2012).
[8] M. Ghavamzadeh, S. Mahadevan, and R. Makar, 'Hierarchical multi-agent reinforcement learning', *Autonomous Agents and Multi-Agent Systems*, **13**(2), 197–229, (2006).
[9] M. Grzes and D. Kudenko, 'Plan-based reward shaping for reinforcement learning', in *Proc. of IEEE Conference 'Intelligent Systems'*, (2008).
[10] C. Guestrin, M. Lagoudakis, and R. Parr, 'Coordinated reinforcement learning', in *Proc. of ICML*, (2002).
[11] A. Iscen and U. Erogul, 'A new perspective to the keepaway soccer: The takers (short paper)', in *Proc. of AAMAS*, (2008).
[12] S. Kapetannakis and D. Kudenko, 'Reinforcement learning of coordination in cooperative multi-agent systems', in *Proc. of AAAI*, (2002).
[13] Q. P. Lau, M. Li Lee, and W. Hsu, 'Coordination guided reinforcement learning', in *Proc. of AAMAS*, (2012).
[14] B. Marthi, 'Automatic shaping and decomposition of reward shaping', in *Proc. of ICML*, (2007).
[15] M. Mozina, J. Zabkar, and I. Bratko, 'Argument based machine learning', *Artificial Intelligence*, **171**, 922–937, (2007).
[16] A. Ng, D. Harada, and S. Russell, 'Policy invariance under reward transformations: theory and application to reward shaping', in *Proc. of ICML*, (1999).
[17] The RoboCup Federation, *RoboCup Soccer Server*, 2002.
[18] S. Sen, M. Sekaran, and J. Hale, 'Learning to coordinate without sharing information', in *Proc. of AAAI*, (1994).
[19] P. Stone, R. Sutton, and G. Kuhlmann, 'Reinforcement learning for robocup soccer keepaway', *Adaptive Behavior*, **13**, 165–188, (2005).
[20] R. Sutton, *Temporal credit assignment in reinforcement learning*, Ph.D. dissertation, University of Massachusetts, 1984.
[21] R. Sutton and A. Barto, *Reinforcement Learning*, MIT Press, 1998.
[22] E. Wiewiora, G. Cottrell, and C. Elkan, 'Principled methods for advising reinforcement learning agents', in *Proc. of ICML*, (2003).