ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-342

Argumentation-Based Reinforcement Learning for RoboCup Soccer Keepaway

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

Abstract. Reinforcement Learning (RL) suffers from several difficulties when applied to domains with no obvious goal state defined; this leads to inefficiency in RL algorithms. In this paper we consider a solution within the context of a widely-used testbed for RL, that of RoboCup Keepaway soccer. We introduce *Argumentation-Based RL* (ABRL), using methods from argumentation theory to integrate domain knowledge, represented by arguments, into the SMDP algorithm for RL by using potential-based reward shaping. Empirical results show that ABRL outperforms the original SMDP algorithm, for this game, by improving the optimal performance.

1 INTRODUCTION

Reinforcement Learning (RL) is a paradigm which allows agents to learn actions they should perform in a given environment, by considering the various rewards and punishments which result from their courses of action. RL has been widely studied in computer science and AI, but also in other disciplines such as economics and game theory where the notion of rational action is central. Typically, however, the rewards agents receive relate to goal states defined in the particular problem area or environment, and the fact that non-goal states are unrewarded leads to two kinds of problems for RL algorithms [5]:

- the *temporal credit assignment* problem—that of determining which part of the behaviour deserves the reward; and
- the *slower convergence* problem—back-propagation of the goal reward over the state space is time-consuming.

In this paper we address these problems within the context of RoboCup Keepaway soccer, regarded as a testbed for single-agent as well as multi-agent RL [11]. Keepaway is played between N 'keepers', who must keep possession of a ball, and $N-1$ 'takers', who must try to take it. As in most existing Keepaway research [11, 12, 14], our work will study the behaviour of keepers. Although at any stage of the game only one keeper makes decisions and learns behaviour, the task involves several independent and autonomous agents, so this decision-making process can still be viewed as a multi-agent learning problem [11].

To solve the temporal credit assignment and slower convergence problems, we introduce heuristics into the RL algorithms. Existing approaches to incorporating high-level domain knowledge into RL [5, 6] require an explicit goal state, but many application domains—Keepaway included—have no such easily-definable goal state, so an approach that can use expert knowledge of the domain in the absence of goals is called for. Our method uses *value-based argumentation frameworks* [1], a form of computational argumentation. The latter studies the concept of a good set of arguments in favour of a given proposition or action [4], and in value-based argumentation the reasons for adopting a set of arguments are supplemented by reference to values which the actions may promote. Argumentation is regarded as a powerful tool in inference, decision-making and decision support, but has never, to our knowledge, been used to improve the performance of RL. We define plausible values for Keepaway soccer and propose *argumentation-based RL* (ABRL), which incorporates value-based argumentation into the Markov Decision Process (MDP) algorithms [13] for RL, with the help of reward shaping techniques [8, 15]. We focus on the Keepaway game in the present paper because it facilitates evaluation, although we see potential for the approach described to be extended to other applications of RL.

Section 2 contains background, in Section 3 we describe our approach, and Section 4 presents experimental results. In Section 5 we describe related work and in Section 6 we conclude.

[1] Imperial College London, UK, {y.gao11,ft,robert.craven}@imperial.ac.uk

2 BACKGROUND

We first describe the application domain of Keepaway soccer. Then we survey the model of RL, MDP, which is the basis of our approach, and describe 'reward shaping', which we use to include domain knowledge into MDP. Finally, we give the fundamentals of value-based argumentation, incorporated into MDP in our approach.

2.1 RoboCup soccer Keepaway

Keepaway is played by N keepers, who try to keep possession of the ball, and $N-1$ takers, who attempt to gain possession of the ball, in a fixed court. An *episode* terminates when the ball goes off court or the takers get the ball; a new episode starts immediately with all the players reset. In the rest of this paper, we consider a configuration with 3 keepers and 2 takers playing in a 20×20 court; at the beginning of each episode, the 3 keepers are each in a different corner, and the takers are all in the remaining corner, with keeper 1 (K_1) having possession of the ball. (The techniques proposed in this paper are, however, independent of the game configurations.)

The objective of the learning algorithm in Keepaway is to determine the best action in specific situations, as the game develops. However, in the RoboCup simulation platform, only primitive actions and coordinate positions of each robot are provided; with only this information, designers have had difficulty in incorporating high-level learning algorithms. As a result, *macro-actions* and *state variables* are proposed by Stone et al. [12]. Macro-actions include:

- **HoldBall():** stay still while keeping possession of the ball;
- **PassBall(i):** kick the ball directly towards keeper K_i;

- **GetOpen():** move to a position free from takers and open for a pass from the ball's current position;
- **GoToBall():** intercept a moving ball or move to a stationary ball.

These macro-actions can in turn be combined into further macro-actions. For instance, keepers not in possession of the ball can be required to execute the **Receive()** action which is defined as:

- **Receive():** if a teammate has the ball, or can get to it faster than this keeper can, then **GetOpen()**; otherwise, **GoToBall()**. Repeat until this keeper has possession of the ball or the episode ends.

In the following we use K_i to represent the ith closest keeper to the ball—so that K_1 is the keeper in current possession. The learning is restricted to K_1, and we further confine it to the macro-action **HoldBall()** and to **PassBall**(2)**ThenReceive** (i.e. **PassBall**(2) then **Receive()**) and **PassBall**(3)**ThenReceive** (similar to the previous one).

In the RoboCup simulation platform, which action is best will depend on the state of the court. This is represented by a vector of thirteen values, shown in Table 1. The list can be naturally gener-

Value name	Description
$dist(K_1, C)$ $dist(K_2, C)$ $dist(K_3, C)$	Distance between keepers and the centre of the court.
$dist(T_1, C)$ $dist(T_2, C)$	Distance between takers and the centre of the court.
$dist(K_1, K_2)$ $dist(K_1, K_3)$	Distance between K_1 and the other two keepers.
$dist(K_1, T_1)$ $dist(K_1, T_2)$	Distance between K_1 and the takers.
$Min_{j\in 1,2} dist(K_2, T_j)$	Distance between K_2 and its closest taker.
$Min_{j\in 1,2} dist(K_3, T_j)$	Distance between K_3 and its closest taker.
$Min_{j\in 1,2} angle(K_2, K_1, T_j)$	The smallest angle between K_2 and the takers with vertex at K_1.
$Min_{j\in 1,2} angle(K_3, K_1, T_j)$	The smallest angle between K_3 and the takers with vertex at K_1.

Table 1. State variables and their meaning (3 keepers and 2 takers)

alised to additional keepers and takers, leading to a linear growth in the number of state variables.

2.2 MDP algorithms

MDP is one of the most widely used mathematical models of RL and has several variants [13]. An 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 probability of moving from state s to state s' by executing action a, and $R(s, a, s')$ gives the immediate reward r received when action a is taken in state s, moving to state s'.

However, in most real environments, the transition probabilities and reward functions are not known. In these cases, *on-policy* learning algorithms are used, which apply temporal-difference updates to propagate information about values of states, $V(s)$, or state-action pairs, $Q(s, a)$. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value. The SARSA(λ) [13] algorithm is such an on-policy learning algorithm. We use the Semi-MDP (SMDP) version of the SARSA(λ) algorithm with replacing eligibility traces, shown as Algorithm 1, because it allows us to deal with partially observable environments, as required by our application domain. In the algorithm, α is a learning rate parameter and λ is a discount factor governing the weight placed on the future. The value e represents *eligibility traces*, which stores

Algorithm 1 Learning algorithm: SARSA(λ) with replacing eligibility traces (adapted from [13])

```
1  Initialise Q(s, a) arbitrarily for all states s and actions a
2  Repeat (for each episode):
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     Repeat until s_t is the terminal state:
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           δ ← r_t + γQ(s_{t+1}, a_{t+1}) − Q(s_t, a_t)
10          e(s_t, a_t) ← 1
11          For all s, a:
12                Q(s, a) ← Q(s, a) + αδe(s, a)
13                e(s, a) ← γλe(s, a)
14          s_t ← s_{t+1}; a_t ← a_{t+1}
```

the credit that previous action choices should receive for current rewards, while γ governs how much credit is delivered back to them. The linear tile-encoding method of [12] we use is suitable for representing states in in our application domain. The action selection rule used in line 5 and line 8 is ϵ-*greedy*: the action with highest $Q(s, a)$ value will be selected for a proportion $1 - \epsilon$ of the trials; for the other ϵ proportion, actions will be selected randomly.

2.3 Reward shaping

Reward shaping, as a medium to convey domain knowledge into RL, has been proven to be an effective technique to improve the convergence speed as well as optimality of MDP in both single-agent and multi-agent RL scenarios [3, 8]. However, despite its effectiveness in many experiments, if used improperly, reward shaping can also mislead the learning process [9]. To deal with such problems, *potential-based* reward shaping has been proposed by Ng et al. [8] as the difference of some potential function Φ over the current state s and the following state s'. Wiewiora et al. [15] extended the potential-based method to the case of shaping functions based on both states and actions: $\Phi(s, a)$. As recommended by Wiewiora et al. [15], *look-ahead advice* should be used if the prior knowledge is predominantly state-based. Since, in the Keepaway application, the domain knowledge mainly focuses on the state of the court, we will use look-ahead advice as the reward shaping method. To integrate look-ahead advice into Algorithm 1, line 9 of the algorithm should be replaced by:

$$\delta \leftarrow r_t + F(s, a, s', a') + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

where the (shaping) reward $F(s, a, s', a')$ is defined as:

$$F(s, a, s', a') = \gamma\Phi(s', a') - \Phi(s, a)$$

obtained when moving from state s to s' by action a based on the difference of potential values Φ between pairs (s, a) and (s', a'). Φ should be given according to the specific domain of application.

2.4 Argumentation frameworks

An *abstract argumentation framework* [4] is a pair (Arg, Att) where *Arg* is a set of *arguments* and $Att \subseteq Arg \times Arg$ is a binary relation representing the *attacks* between arguments. We say that a set $S \subseteq$ *Arg* attacks an argument B when some member of S attacks B. A set $S \subseteq$ *Arg* is said to be *conflict-free* when no member of S attacks a member (possibly itself) of S.

Using these definitions, semantics of abstract argumentation frameworks are defined as sets of arguments (extensions), reflecting

concepts of rational acceptability. Two widely-used semantics are as follows (given an abstract argumentation framework $\mathsf{F} = (Arg, Att)$):

- $S \subseteq Arg$ is an *admissible extension* for F iff S is conflict-free, and S attacks any argument in $Arg - S$ that attacks an argument in S;
- $S \subseteq Arg$ is a *preferred extension* for F iff S is an admissible extension, and there is no $\subset$-larger subset of Arg that is also admissible.

Intuitively, an admissible extension is one whose arguments can counter-attack any arguments attacking them; and a preferred extension is an admissible extension that cannot be made any larger, thus representing a maximally acceptable set of arguments.

In some contexts, particularly involving practical reasoning, the logical relations between arguments are not enough to decide what is rationally acceptable, and the values promoted by arguments must be considered. Value-based argumentation frameworks (VAFs; see, e.g., [1]) were therefore introduced to model reasoning where different agents face problems of practical rationality, when the agents' preferences between their values may differ. The key idea of VAFs 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* is a strict partial order over V; an audience corresponds to the scale of values of a particular agent. An *audience-specific VAF* is a tuple $(Arg, Att, V, val, Valpref)$, where:

- (Arg, Att) is an abstract argumentation framework;
- V is a set of values;
- $val : Arg \rightarrow V$ gives the value promoted by arguments (denoted, e.g., $A \mapsto v$);
- *Valpref*, the audience, is a strict partial order over V; we will typically write this relation as $>_v$.

The ordering over values, *Valpref*, within an audience-specific VAF, allows for the values promoted by a given argument to be taken into account in the definition of the semantics. The *simplification* of an audience-specific VAF of the form above is the abstract argumentation framework (Arg, Def), where $(A, B) \in Def$ iff $(A, B) \in Att$ and $val(B) \not>_v val(A)$. $(A, B) \in Def$ is pronounced 'A *defeats* B', so that defeat imposes a value-constraint on the relationship of attack. The definitions of 'conflict-free', 'admissible', and 'preferred', above, are then adapted to use the relation of defeat instead of (merely) attack, with semantics of the VAF defined in terms of extensions of those of its simplification (Arg, Def).

3 ARGUMENTATION IN KEEPAWAY

In this section, we identify arguments representing expert knowledge of the Keepaway domain (section 3.1), define the notion of domain-specific value-based argumentation framework (section 3.2) and discuss how to use it to associate numerical values with actions, feeding into reward shaping for RL (section 3.3). Even though we focus on the 3-2 Keepaway, the techniques we propose here apply to other Keepaway settings, or, more generally, to other RL problems, as well.

3.1 Arguments for rewarding rules

In the original SMDP approach for Keepaway [11], the reward is defined by $r_t = \mathit{CurrentTime} - \mathit{LastActionTime}$. Here, *CurrentTime* is the time when a keeper holds the ball or an episode ends, and *LastActionTime* is the time when a keeper selected the last action. So, roughly speaking, this reward system is distance-oriented: passing the ball to farther keepers is more encouraged. However, it is clear that the presence of takers should affect this: if a keeper is far from the one in possession of the ball but very near to both takers, then the keeper in current possession should pass to the other keeper, even though he may be nearer. We therefore stipulate that K_i ($i \in \{2, 3\}$) is *far* iff $Min_{j\in 1,2} dist(K_i, T_j) \geqslant L$, where L is a parameter fed as input. Arguments regarding what to do when a keeper is *far* can then be formulated:

F2: PassBall(2)ThenReceive IF K_2 is *far*
F3: PassBall(3)ThenReceive IF K_3 is *far*

This rewarding system seems reasonable at first sight: the farther the target keeper is from takers, the more likely that the ball can be controlled by the keepers for a longer time. Nevertheless, this assumes that the ball can be successfully passed to the target keeper, which is not always so. Take the scenario in Figure 1, for example. Even though K_3 is farther from the takers than

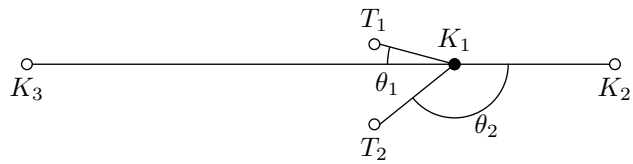

Figure 1. Example scenario: insufficiency of *far* arguments

K_2, $Min_{j\in 1,2} angle(K_2, K_1, T_j)$=$\theta_2$ is significantly larger than $Min_{j\in 1,2} angle(K_3, K_1, T_j)$=$\theta_1$. **PassBall(2)ThenReceive** is clearly the best choice in this situation as it has lower risk of interception, but according to the existing arguments **F2** and **F3**, **PassBall(3)ThenReceive** will get higher reward. Clearly, some important factors, e.g., $Min_{j\in 1,2} angle(K_i, K_1, T_j)$, need to be taken into account in the rewarding system. We define that K_i ($i \in \{2, 3\}$) is *open* iff $Min_{j\in 1,2} angle(K_i, K_1 T_j) \geqslant A$, where A is an input parameter acting as an acceptable threshold. We introduce the additional arguments:

O2: PassBall(2)ThenReceive IF K_2 is *open*
O3: PassBall(3)ThenReceive IF K_3 is *open*

to take better account of situations such as the one in Figure 1.

Another subtle but serious problem is that the reward for **HoldBall()** is often too small, especially when the takers are far from K_1. For example, in the beginning of each episode, as both takers are together in a different corner than K_1 and they are always chasing the ball, the best strategy should intuitively be holding the ball until the takers are close. But given the existing rewarding rule, passing the ball immediately has higher reward. Because the reward of passing the ball is always much higher than that of **HoldBall()**, when the learning proceeds, the $Q(s, \textbf{HoldBall()})$ will become smaller and smaller, leading in turn to to fewer chances for execution of **HoldBall()**. To break this vicious cycle, we need additional reward rules allocating **HoldBall()** higher rewards. We add the argument:

H: HoldBall() IF *True*

The fact that the body of this rule is *True* means that **HoldBall()** is the default action, always available to be supported by argument H.

All five arguments above can be viewed as reasons to make different decisions in various, possibly overlapping situations. Take the scenario in Figure 1 as an example, and suppose that given specific input values for L and A (defining *far* and *open*), K_2 is evaluated as far and open and K_3 is far and not open. Then passing to K_2 is a reasonable decision because both arguments *F2* and *O2* provide reasons

to support it. Similarly, passing to K_3 is reasonable because argument *F3* supports it. In addition, **HoldBall()** can be reasonably chosen since that is the default option. So we face a dilemma in this scenario: different arguments support different actions simultaneously, but only one action can be selected. To solve this kind of conflict, we can evaluate the strengths of different arguments in specific situations, and use this evaluation to rank the actions.

3.2 Argumentation framework

Consider again the scenario illustrated in Figure 1. As discussed earlier, four arguments hold in this situation: *O2*, *F2*, *F3* and *H*. **PassBall**(2)**ThenReceive** is supported by both *O2* and *F2*, **PassBall**(3)**ThenReceive** is supported by *F3*, whereas **HoldBall()** is supported by *H*. Intuitively, any two arguments that support different actions conflict with one another. The corresponding abstract argumentation framework is shown in Figure 2 (i). Generically, by

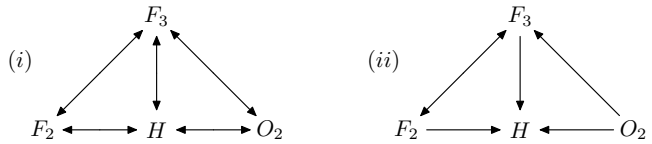

Figure 2. (i) Attacks between arguments in scenario shown in Figure 1; (ii) Simplified argumentation framework obtained after eliminating unsuccessful attacks

extending the notion of abstract argumentation framework (see Section 2.4), we can define a *3-2-Keepaway scenario-specific argumentation framework* as follows:

Definition 1. A *3-2-Keepaway scenario-specific argumentation framework* (SAF) is a tuple (Sta, Arg^*, AF) such that:

1. *Sta* is the specific state of the game, which is represented by the values of the 13 state variables in Table 1;
2. $Arg^* = \bigcup_{i=1}^{3} Arg_i^*$, where $Arg_1^* = \{H\}$, $Arg_2^* = \{F2, O2\}$ and $Arg_3^* = \{F3, O3\}$;
3. $AF = (Arg, Att)$ is an abstract argumentation framework where:
   - $Arg = \bigcup_{i=1}^{3} Arg_i$, where $Arg_i \subseteq Arg_i^*$ such that $Arg_1 = Arg_1^*$, and for $i \in \{2, 3\}$, (i) $Fi \in Arg_i$ iff K_i is *far*; and (ii) $Oi \in Arg_i$ iff K_i is *open*;
   - $Att \subseteq Arg \times Arg$ such that $(A, B) \in Att$ iff $A \in Arg_i$ and $B \in Arg_j$ with $i \neq j$.

Note that for any two arguments *A* and *B* in a SAF, if $(A, B) \in Att$ then $(B, A) \in Att$. Specifically, all the arguments in a SAF fall into three categories: Arg_1, which just contains *H*, and Arg_i $(i = 2, 3)$, which contains all available arguments in the scenario characterised by *Sta* that support **PassBall**(i)**ThenReceive**. Based on the given division of arguments, it is easy to see that arguments in the same category (Arg_i) are conflict-free; arguments in different categories symmetrically attack each other. Also, each preferred extension corresponds to an action, because it corresponds to exactly one Arg_i.

Theorem 1. Given $SAF = (Sta, Arg^*, AF = (Arg, Att))$, S is a preferred extension for AF iff $S = Arg_i$ for some $i \in \{1, 2, 3\}$.

Proof. First we prove that if $S = Arg_i, i \in \{1, 2, 3\}$, then S is a preferred extension. Assume $S = Arg_i$ for some $i \in \{1, 2, 3\}$. Then S is conflict-free, and as Arg_i attacks all arguments not in Arg_i, S is admissible and cannot be extended without losing the property of being conflict-free. So S is preferred.

For the other direction, assume S is a preferred extension. We must show there is i s.t. $S = Arg_i$. Assume not. Then either $S \subset Arg_i$ for some i, or $\exists A, B \in S$ such that $A \in Arg_i$ and $B \in Arg_j$ for $i \neq j$. The latter is clearly impossible or else S would not be conflict-free. So $S \subset Arg_i$ for some i. But each Arg_i is admissible because it is conflict-free and all attacks in the framework are symmetric. So S is not maximal-$\subseteq$ admissible, and so not preferred. Contradiction. □

We extend the SAF to *3-2-Keepaway audience-scenario-specific value-based argumentation framework* (VSAF) by introducing the concept of audience-specific values [1], where intuitively the audience is a domain expert. To be specific, we introduce the value set V which consists of three values:

- **Lower the risk of marking (MK)**: reducing the risk of K_2 and K_3 being marked is valued;
- **Lower the risk of interception (IT)**: reducing the risk of the ball being intercepted is valued;
- **Lower the risk of tackle (TK)**: reducing the risk of the ball being taken from K_1 is valued.

Because K_1 can attract takers to approach by holding the ball, the value *MK* is promoted by the argument *H*. With regard to passing, the risk of interception can be reduced if the ball is passed to a keeper which is open. So *IT* is promoted by *O2* and *O3*. In addition, by passing the ball to the other keepers who are far from the takers, the risk of being tackled will be lower, and the value *TK* can be, therefore, promoted by *F2* and *F3*.

Definition 2. A *3-2-Keepaway audience-scenario-specific value-based argumentation framework* (VSAF) is a tuple $(SAF, V, val, Valpref_{Sta})$ where:

1. $SAF = (Sta, Arg^*, AF = (Arg, Att))$ is a 3-2-Keepaway scenario-specific argumentation framework;
2. $V = \{IT, TK, MK\}$ is the set of values;
3. $val = \{O2 \mapsto IT, O3 \mapsto IT, F2 \mapsto TK, F3 \mapsto TK, H \mapsto MK\}$ is a function which maps from Arg^* to V;
4. $Valpref_{Sta} \subseteq V \times V$ is a preference relation (transitive, irreflexive and asymmetric), reflecting the value preferences in state *Sta*.

Based on the preference of values in scenarios, unsuccessful attacks can be eliminated [1] and simplified abstract argumentation frameworks derived from a VSAF, adapting the method reviewed in Section 2. Take the scenario in Figure 1. Suppose that in this situation $Valpref_{Sta}$ is given be $IT >_v TK >_v MK$; then the attack relationships $(F3, O2)$, $(H, O2)$, $(H, F3)$ and $(H, F2)$ are eliminated. The simplified argumentation framework is shown in Figure 2 (ii). The unique preferred extension of this simplified framework is $\{O2, F2\}$, in which there is an argument promoting the highest value, namely *IT*, and where both arguments support the action **PassBall(2)ThenReceive**. Recall that, according to the analysis in Section 3.1, **PassBall(2)ThenReceive** is the most encouraged action according to the domain knowledge. In fact, we can prove that, in a simplified argumentation framework, the highest value will be promoted by an argument in every preferred extension and each preferred extension corresponds to one action.

Theorem 2. Let $AF^- = (Arg, Att^-)$ be a simplified argumentation framework derived from $VSAF = (SAF, V, val, Valpref_{Sta})$. Let $V_{Sta} = \{v \mid v \in V, \exists A \in Arg \text{ s.t. } A \mapsto v\}$. Define $v_{max} \in V_{Sta}$ such that $\forall v \in V_{Sta}, (v, v_{max}) \notin Valpref_{Sta}$. For any preferred extension S of AF^-, $\exists A \in S$ s.t. $A \mapsto v_{max}$.

Proof. We prove this theorem by contradiction. Suppose S is a preferred extension and $\nexists A \in S$ such that $A \mapsto v_{max}$. Assume $B \notin S$ and $B \mapsto v_{max}$. Then for any $C \in S$, $(B, C) \in Att^-$ and $(C, B) \notin Att^-$. As a result, S is not admissible, which contradicts the assumption that S is a preferred extension. □

Theorem 3. Let $AF^- = (Arg, Att^-)$ be a simplified argumentation framework derived from $(SAF, V, val, Valpref_{Sta})$. If S is a preferred extension of AF^-, then for some $k \in \{1, 2, 3\}$, $S = Arg_k$.

Proof. Suppose S is a preferred extension. We can prove this theorem by proving $S \subseteq Arg_k$ and $Arg_k \subseteq S$ for some $k \in \{1, 2, 3\}$. Note that by eliminating unsuccessful attacks, some of the symmetrical attacks are replaced by one-side attacks. So in Arg, for any two arguments in different categories, there still exists some attack relationships between them, either symmetrical or one-sided. Since S is conflict-free, S is a subset of a specific Arg_k for some $k \in \{1, 2, 3\}$. To show $Arg_k \subseteq S$, we first show that S is non-empty. Theorem 2 shows that each preferred extension contains an argument that promotes the highest value. Therefore all the preferred extensions of AF^- are non-empty. Now we prove $Arg_k \subseteq S$ by contradiction. Suppose $\exists B \in Arg_k$ and $B \notin S$. Because $S \cup \{B\} \subseteq Arg_k$, $S \cup \{B\}$ is conflict-free. Assume that $\exists C \in Arg_j, j \neq k$ such that $(C, B) \in Att^-$. Since S is a non-empty preferred extension and $S \subseteq Arg_k$, S attacks C. So $S \cup \{B\}$ is admissible. But this contradicts with the assumption that S is a preferred extension. Therefore, the assumption is incorrect and $Arg_k \subseteq S$. □

As a consequence of these results, the rational actions determined argumentatively are the actions promoting the highest (viable) values. This justifies the choice of numerical values discussed next.

3.3 Rankings and potential values of actions

Recall that our aim in constructing the argumentation frameworks is to get $\Phi(s, a)$ (Section 2.3). Learning performance can be very sensitive to the $\Phi(s, a)$ values, so they should reflect our domain knowledge accurately. Since the domain knowledge is essentially based on the idea of promoting higher values in specific scenarios, assigning numbers to the values is the most direct method to transform abstract domain knowledge into numerical values. Specifically, by assigning values with numbers, VSAF can be extended to *3-2-Keepaway audience-scenario-specific numerical-value-based argumentation framework*, which is formally defined as follows:

Definition 3. A *3-2-Keepaway audience-scenario-specific numerical-value-based argumentation framework* (NVSAF) is a tuple $NVSAF = (VSAF, G)$ such that:

- $VSAF = (SAF, V, val, Valpref_{Sta})$ is a 3-2-Keepaway audience-scenario-specific value-based argumentation framework
- G is a function on *VSAF* such that $G : Sta \times V \to \mathbb{R}$.

For example, given the scenario *Sta* showed in Figure 1, $G(Sta, MK) = R_{MK}$, $G(Sta, IT) = R_{IT}$ and $G(Sta, TK) = R_{TK}$ where $R_{MK}, R_{IT}, R_{TK} \in \mathbb{R}$. Because **HoldBall()** is supported by argument *H*, which promotes value *MK*, we have $\Phi(Sta, \textbf{HoldBall()}) = R_{MK}$. Similarly, $\Phi(Sta, \textbf{PassBall(2)ThenReceive}) = R_{IT} + R_{TK}$ and $\Phi(Sta, \textbf{PassBall(3)ThenReceive}) = R_{TK}$. It should be emphasised that $G(Sta, V)$ values should be chosen so as to be consistent with the ordering of values in $Valpref_{Sta}$. For example, in the scenario showed in Figure 1, we have supposed that $IT >_v TK >_v MK$. As a result, we should have $R_{IT} > R_{TK} > R_{MK}$. In specific states, actions will be supported by a number of arguments which promote different values. So we can simply compute the $\Phi(Sta, a)$ values by summing up the numerical values of abstract values that corresponds to this action a. Formally, in state Sta, the potential value of action a is: $\Phi(Sta, a) = \sum_{arg \in Arg_i} G(Sta, val(arg))$ where $i = 1$ if a is **HoldBall()**, and $i = 2, 3$ if a is **PassBall(i)ThenReceive**.

4 IMPLEMENTATION RESULTS

In this section we deploy the techniques of Section 3 within the RoboCup Soccer simulation platform (see `www.robocup.org`). We first give the parameters we use for the experimentation (Section 4.1) and then discuss the empirical results (Section 4.2).

4.1 Threshold and parameter values

The parameters of numerical values used in this paper are proposed on the basis of observation, runtime data statistics and earlier results. As all these numbers are the result of a limited number of experiments, we make no claims that they are optimal, and most likely there is considerable room for improvement.

For the threshold values for determining *far* and *open*, we set $L = 10$ and $A = 15$. We presume a risk-avoidance attitude, always ranking *IT* higher than *TK*. Thus, $Valpref_{Sta}$ can be defined as:

$$Valpref_{Sta} = \begin{cases} MK > IT > TK & \text{iff} \quad K_1 \text{ is } safe \\ IT > MK > TK & \text{iff} \quad K_1 \text{ is } under\ threat \\ IT > TK > MK & \text{iff} \quad K_1 \text{ is } in\ danger \end{cases}$$

Here we define that when $Min_{j \in 1,2} dist(K_1, T_j) > 10$, K_1 is *safe*; when $5 < Min_{j \in 1,2} dist(K_1, T_j) \leqslant 10$, K_1 is *under threat*; when $0 < Min_{j \in 1,2} dist(K_1, T_j) \leqslant 5$, K_1 is *in danger*. Furthermore, we define the function F in *NVSAF* as: [2]

$$G(Sta, \{MK, IT, TK\}) = \begin{cases} \{40, 20, 10\} & \text{iff} \quad K_1 \text{ is } safe \\ \{10, 20, 5\} & \text{iff} \quad K_1 \text{ is } under\ threat \\ \{0, 25, 5\} & \text{iff} \quad K_1 \text{ is } in\ danger \end{cases}$$

Here, the numbers are set according to the following principles:

- **HoldBall()** should be encouraged when K_1 is safe. According to the analysis in Section 3.1, the reward of **HoldBall()** in the original system is too low. As a result, the abstract value *MK*, promoted by **HoldBall()**, should be assigned higher numerical values.
- *IT* should be assigned higher numerical values. As described in Section 3.1, the value *IT* is out of consideration in the existing rewarding system. In other words, the rewards of *IT* will only come from the potential values. So *IT* should have higher values.
- The magnitude of these numerical values should be comparable with the $Q(s, a)$ values in the original SMDP algorithm. If the numerical values here are much larger than the original Q values, the effect of the original rewarding rules will be 'submerged' and, as a result, the convergence process will take a longer time; otherwise, the shaping reward will quickly be 'submerged' and the learning performance will not be significantly improved.

4.2 Empirical results

The parameters in the SMDP have the same values as described in [12], while the values of other parameters have been described in the previous sections. Here we employ the look-ahead advice [15] as the reward shaping technique, and the $\Phi(s, a)$ values are computed according to the method described in Section 3.3. The learning curve of the standard SMDP algorithm and the argumentation-based SMDP algorithm have been shown in Figure 3. After the first few hours

[2] Here $G(Sta, V)$ is a compact representation of $G(Sta, v), \forall v \in V$

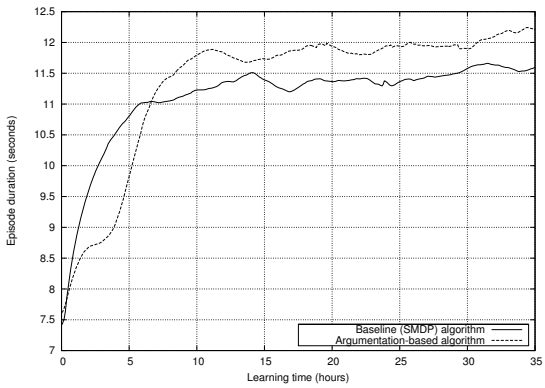

Figure 3. Performances of the standard SMDP algorithm and the argumentation-based algorithm. Each curve represents the average performance over 10 independent trials.

of adjustment, the argumentation-based approach increases faster than the standard algorithm, and becomes constantly better than the standard algorithm after about 7 hours of learning. Note that each learning trial will stop after almost 35 hours of training because the RoboCup Soccer simulation platform becomes unstable then.

5 RELATED WORK

There has been some research on improving the performance of RL by using high-level domain knowledge. Marthi [6] proposed *abstract MDP* to find an approximation of the true shaping functions by solving a simpler abstract problem under the instructions of prior knowledge. Grzes and Kudenko [5] used the high-level STRIPS operator knowledge in reward shaping to search for an optimal policy and showed that the STRIPS-based reward shaping converges faster than the abstract MDP approach. However, in both of these approaches, a goal state is required, which is unavailable in the Keepaway game. Bianchi et al. [10] proposed the Heuristically Accelerated RL method to integrate heuristic information into RL. However, in their approach, for a proportion $1 - \epsilon$ (see Section 2.2), the action which is chosen by the heuristics will be executed, and for the other ϵ proportions, a random action will be chosen. As ϵ is often very small, in most occasions, the heuristically chosen actions will be executed and, as a result, in order to improve the performance of RL, fine-grained and accurate heuristics are needed. But in complex applications such as Keepaway, precise heuristics can hardly be obtained.

There is research on improving machine learning by argumentation. Mozina et al. [7] proposed argumentation based machine learning, which combines arguments with the original examples of CN2 algorithm [2] to form argumented examples. The use of arguments significantly improves the performance of CN2. However, the relationships between different arguments are not taken into account in their technique, which restricts the effect argumentation has. Also, the machine learning technique considered, CN2, is supervised.

6 CONCLUSION

We described a novel approach to reinforcement learning (RL) that draws on concepts from argumentation theory. We use argumentation frameworks to represent the domain knowledge systematically and explicitly, so that by identifying the preferred extensions of the argumentation frameworks, domain experts and algorithm designers can verify their ideas and integrate heuristic instructions into RL.

We implemented ABRL in the SMDP algorithm and conducted experiments on Keepaway games. We showed that the best actions and highest values can be identified by obtaining preferred extensions of the argumentation frameworks. Based on these instructions, domain knowledge can be turned into numerical values and incorporated into the original SMDP algorithm. Experiment results show that by using the argumentation-based approach, the optimal performance can be significantly improved.

There are four main issues about ABRL that still need to be further researched. First of all, how to extend the argumentation frameworks to integrate more accurate and detailed heuristic instructions into the RL algorithms. In this paper, we use only five arguments. This greatly restricts the effect the arguments may have on the learning process. Secondly, in our current work, we obtain the preferred extensions in VSAFs by hand. In large scale systems where many arguments may be employed, this will be extremely time-consuming or even impossible. So in the long term, computing the preferred extensions automatically will be essential. Thirdly, we need a method to assign numbers to values automatically. At present, the numerical values are proposed by domain experts. Inevitably, some errors may be contained in these hand-tuned numbers. Since the strength of each argument can be identified by analysing the structure of the argumentation frameworks, we may be able to assign numbers automatically so that the domain knowledge can be reflected by those numbers more accurately. The last issue is how to extend ABRL to multi-agent learning scenarios. Currently, our work focuses on the learning process of an individual agent. However, as a powerful tool to handle negotiation problems between agents, argumentation may have a greater effect in the multi-agent learning scenario.

REFERENCES

[1] T. Bench-Capon and K. Atkinson, 'Abstract argumentation and values', in *Argumentation in Artificial Intelligence*, eds., L. Rahwan and G. Simari, Springer, (2009).
[2] P. Clark and T. Niblett, 'The CN2 induction algorithm', *Machine Learning*, **3**, 261–283, (1989).
[3] S. Devlin, M. Grzes, and D. Kudenko, 'Multi-agent, reward shaping for robocup keepaway (extended abstract)', in *Proc. AAMAS*, (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] M. Grzes and D. Kudenko, 'Plan-based reward shaping for reinforcement learning', in *Proc. 4th International IEEE Conference 'Intelligent Systems'*, (2008).
[6] B. Marthi, 'Automatic shaping and decomposition of reward shaping', in *Proc. of ICML*, (2007).
[7] M. Mozina, J. Zabkar, and I. Bratko, 'Argument based machine learning', *Artificial Intelligence*, **171**, 922–937, (2007).
[8] A. Ng, D. Harada, and S. Russell, 'Policy invariance under reward transformations: theory and application to reward shaping', in *Proc. ICML*, (1999).
[9] J. Randlov and P. Alstrom, 'Learning to drive a bicycle using reinforcement learning and shaping', in *Proc. ICML*, (1998).
[10] R.Bianchi, C. Ribeiro, and A. Costa, 'Accelerated autonumous learning by using heuristic selection of actions', *J. of Heuristics*, **14**, 135–168, (2008).
[11] P. Stone and R. Sutton, 'Keepaway soccer: a machine learning testbed', in *RoboCup 2001*, LNAI 2377, Springer, (2002).
[12] P. Stone, R. Sutton, and G. Kuhlmann, 'Reinforcement learning for robocup soccer keepaway', *Adaptive Behavior*, **13**, 165–188, (2005).
[13] R. Sutton and A. Barto, *Reinforcement Learning*, MIT Press, 1998.
[14] N. Tanaka and S. Arai, 'Teamwork formation for keepaway in robotics soccer (reinforcement learning approach)', in *Proc. PRIMA*, (2006).
[15] E. Wiewiora, G. Cottrell, and C. Elkan, 'Principled methods for advising reinforcement learning agents', in *Proc. ICML*, (2003).