Designing Deception in Adversarial Reinforcement Learning

Sanjiban Choudhury · Alok Kanti Deb · Jayanta Mukherjee

Received: date / Accepted: date

Abstract In an adversarial scenario, deceptions are powerful tools capable of earning time delayed rewards which an agent can use to circumvent the opponent’s counter attack. This paper illustrates deception as a complementary policy to direct objective satisfaction. In this paper, a framework for deceptions is defined to finally determine the number and nature of these actions. A minimal set of these actions ensures fast learning while being robust enough to confront any strong opponent. To satisfy the conditions of optimality with changing opponent policies, suitable judging criteria are provided. The focus of this work is primarily on application of Robot soccer. Benchmark examples of tactics involving a subset of players are implemented with deceptions, the results of which are compared to standard hand-coded solutions. This paper focuses on such contests with an aim to generate a library of tactics to deal with them seamlessly without having to customise the function approximator or learning algorithm to a great deal.

Keywords Reinforcement Learning · Action Space · SMDP · Robot Soccer

1 Introduction

Deception is defined in this context as a group of actions auxiliary to directly satisfying an objective. As the name indicates, these actions encode covert objectives, which in the long run can result in finding a weakness in the opponent’s policy.
In an episodic adversarial reinforcement learning scenario where the opponent’s policy is unknown, these actions have the property of carrying a high expected time-delayed reward as well as the ability to quickly explore the policy space. For example, in a situation involving the placement of defenders during a corner kick, deception can entice the attackers to carry out a tactic which is anticipated by the defending team. The balance between deception and objective pursuit, which depends mainly on the strength of the counter attack strategies of the opponent, becomes the goal of the learning algorithm.

Reinforcement Learning has increasingly gained importance as a powerful tool in problem solving. Recent work in fast policy search and policy gradient methods (Sutton et al, 2000; Peters and Schaal, 2008) have resulted in major applications (Bagnell and Schneider, 2001; Ng et al, 2004; Peters and Schaal, 2006; Kober et al, 2008). In the domain of Robot Soccer, the focus has been on batch learning algorithms (Riedmiller et al, 2009), based on fitted value iteration algorithms (Gordon et al, 1995) and fitted Q-iteration (Ernst et al, 2006). They have been used to learn cooperative behaviours in simulated league (Ma and Cameron, 2008), real robots (Asada et al, 1999) and humanoid robots (Ogino et al, 2004). The keep-away framework (Stone et al, 2005) is a significant development in application of learning tactics, as well as a popularized machine learning testbed. This has led to work on task decomposition and neuro-evolution of keepaway players (Whitson et al, 2003). Another major issue that has been addressed is the flexibility of state representation and action space choice. Several approaches have been used to design the state space (Dean et al, 1995; Dean and Robert, 1997) including the advantages of using $\epsilon$-reduction (Asadi and Huber, 2004) to partition state space and extend it for temporal abstraction.

The problem of large rigid action space has been addressed (Singh et al, 2004) by developing intrinsically motivated learning algorithms. This is an intuitive way to derive flexible hierarchical reusable skills which can be used for competent autonomy. However a middle ground is required between rigid primitive action spaces, and learning reusable skills so as to meet the demands of both flexibility as well as ease of learning. This calls for a restricted set of higher level actions which can facilitate efficient exploration of the policy space, confront competent responses from opponents and adapt to changing opponent policies. This paper defines a grammar for selecting such actions using knowledge of the domain as well as worst case assumptions of opponent’s response.

Deceptions together form a minimal closed network of options which are complementary. This nature allows them to contend with any given strong response of an opponent and quickly arriving at an optimal policy over a restricted space. This work attempts to produce results comparable to those of hand-coded policies, which can adapt easily to changes in opponent’s policies as well as achieve a certain level of independence from the specific algorithm or function approximator used. As a result a library of flexible tactics can be generated.

In the next section, a framework for episodic adversarial reinforcement learning has been presented along with supporting definitions. The scenario is divided into three major categories depending on the objectives. The higher level greedy option and the deception option are defined. Suitable judging criteria are presented to approximate optimal functions which otherwise heavily rely on an opponent’s policy. A proof regarding the number and nature of deceptions is presented, which allows a user to use basic knowledge of the domain to create a robust restricted
action space. In Sect. 3 a general learning algorithm is explained along with a conventional function approximator. In Sect. 4 the experiments are mainly focussed on RoboCup simulation. For every category, a benchmark example is implemented with deceptions and the results are compared to hand-coded policies. Finally the work is concluded in Sect. 5 along with the implications of the work in the future of RoboCup.

2 Preliminaries

In the finite Markov Decision Process framework, a learning agent interacts with an environment at some discrete, lowest level time scale. On each time step, t, the agent perceives the state of the environment, \( s_t \in S \), and on that basis chooses a primitive action, \( a_t \in A_s \). In response to each action, \( a_t \), the environment produces one step later a reward, \( r_{t+1} \), and a next state, \( s_{t+1} \). The environment’s transition dynamics can be modelled by one-step state transition probabilities

\[
p^{o}_{ss'} = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}
\]

and one-step expected rewards

\[
r^{a}_{s} = E\{r_{t+1} \mid s_t = s, a_t = a\}
\]

The agent’s objective is to learn a Markov policy, a mapping from states to probabilities of executing each available primitive action, \( \mu : S \times A \to [0, 1] \), that maximizes the expected discounted future reward from each state \( s \).

However the one-step primitive action \( a_t \) is insufficient to represent closed-loop policies for taking actions over a period of time. The term options (Sutton et al, 1999) has been used for the generalization of primitive actions to include temporally extended courses of action.

**Definition 1** An option, \( o \in O_{s_t} \), consists of the following components:

1. Probability of choosing an action given a state (a policy) \( \pi : S \times A \to [0, 1] \)
2. Probability of termination given a state \( \beta : S \to [0, 1] \)
3. an initiation set \( I \subseteq S \)

A Markov option executes by selecting an action, \( a_t \), according to the probability distribution \( \pi(s_t, \cdot) \), following which the environment transitions to the next state \( s_{t+1} \) where the option either terminates with probability \( \beta(s_{t+1}) \) or continues on. Thus for any option \( o \), let \( \xi(o, s, t) \) denote the event of \( o \) being initiated in state \( s \) at time \( t \). Then the reward for execution of \( o \) in \( s \in S \)

\[
r^{o}_{s} = E\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} \mid \xi(o, s, t)\}
\]

where \( (t+k) \) is the random time at which \( o \) terminates and \( \gamma \) is the discount factor of future rewards. The state transition model of an option is the combination of the likelihood of an option terminating at a state along with a measure of how delayed that outcome is relative to \( \gamma \).

\[
p^{o}_{ss'} = \sum_{k=1}^{\infty} p(s', k)\gamma^{k}
\]
for all \( s' \in S \), where \( p(s', k) \) is the probability that the option terminates in \( s' \) after \( k \) steps. The option-value function \( Q^\mu(s, o) \) is defined as the value of taking option \( o \) in state \( s \in S \) under policy \( \mu \):

\[
Q^\mu(s, o) = E\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid \xi(o, \mu, s, t)\}
\]

(5)

Semi-Markov options are options for which policies and termination conditions depend on events since the initiation of the option. These options can capture events like timeout, which maybe incorporated in deception.

**Definition 2** Policies over options are defined as \( \mu : S \times O \to [0, 1] \), which selects an option, \( o \in O_s \), according to the probability distribution \( \mu(s, o) \).

A flat policy, \( flat(\mu) \), is defined as a fixed policy which does not change over time. The Bellman equations for general policies and options have been derived (Sutton et al, 1999) and it is shown (McGovern and Sutton, 1998; Parr and Russell, 1998; Drummond, 1998; Hauskrecht et al, 1998; Meuleau et al, 1998; Bradtke and Duff, 1995) that option-values converge to optimum values under the same conditions as conventional Markov learning (Parr, 1998).

**Definition 3** Adversarial reinforcement learning scenario is an episodic scenario involving an agent, which is implementing the learning algorithm, against an opponent. The term episodic implies that there exists either a terminal failure state \( s \in S_f \), or a terminal goal state \( s \in S_g \), or both. The role assumed is always that of the agent, while the adversary is the opponent. Both the agent as well as the opponent may have terminal goal or failure states. The function \( r : S \times O \times S \to \mathbb{R} \), where \( \mathbb{R} \) is the set of real numbers, denotes immediate rewards \( r(s, o, s') \) when option \( o \) is taken at \( s \) to transition to \( s' \). It is designed according to the objectives of the agent. Depending on the combination of goal and failure states, adversarial scenarios can be categorized into three cases, each with a different reward structure.

Case I: The aim of the agent is to avoid entering a terminal failure state while the aim of the opponent is to drive the agent towards it. Such cases can be represented with a reward structure for the agent:

\[
r(s, o, s') = \begin{cases} 
\tau(s, o, s') & \text{if } s \notin S_f \\
0 & \text{otherwise}
\end{cases}
\]

(6)

where \( \tau(s, o, s') \) is the number of timesteps elapsed while transitioning from \( s \) to \( s' \) by executing \( o \). For example, in a scenario involving midfielders keeping possession of the ball while attackers attempt to take it, the midfielders would be the agent while attackers the opponent. The terminal failure state would be an attacker possessing the ball.

Case II: The aim of the agent is to enter a terminal goal state while the aim of the opponent is to drive the agent away from it. Such cases can be represented with a reward structure for the agent:

\[
r(s, o, s') = \begin{cases} 
c & \text{if } s \in S_g \\
0 & \text{otherwise}
\end{cases}
\]

(7)
where $c > 0$. For example, attackers which are attempting to gain possession of the ball from midfielders would assume the role of an agent while the midfielders would be the opponent; the terminal goal state being gaining possession of the ball.

Case III: The aim of the agent is to avoid entering a terminal failure state while attempting to reach a terminal goal state. The aim of the opponent is to drive the agent towards the terminal failure state while preventing it from entering the terminal goal state. Such cases can be represented with a reward structure for the agent:

$$r(s, o, s') = \begin{cases} c & \text{if } s \in S_g \\ -d & \text{if } s \in S_f \\ 0 & \text{otherwise} \end{cases}$$

where $c > 0$ and $d > 0$. For example, in a scenario where a defender, the agent, has to tackle an attacker, the opponent, the terminal goal state would be a successful tackle. However, if the attacker outruns the defender considerably, it would lead to a terminal failure state.

These cases are illustrated in Fig. 1. For multiple agents, opponents, terminal goal and failure states, the cases can be generalized by considering a common goal or a collection of goals for agents belonging to the same team. A similar generalization can be done in case of failure. The scope of this paper is restricted to single agent learning, thus every agent individually executes the learning algorithm. For example, a scenario where agents together have 2 terminal goal states and 3 terminal failure states falls under the category of case III because the scenario involves both terminal goal and failure states. For such an example, when referring to a terminal failure state, all 3 states will be implied.

**Definition 4** *Higher level greedy option, $o_g \in O$ is an option chosen at every state $s \in S$ such that it maximizes expected reward against an inactive opponent not executing any option, $\mu_{\text{inactive}} : S \times O \rightarrow \{0\}$.*

$$o_g = \arg \max_{o \in O} E\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | flat(\pi), \forall s \in S\}$$

Fig. 1: (a) Case I (b) Case II (c) Case III
where $E\{\}$ is the expected value, $\gamma$ is the discount factor of future rewards and $\pi$ is the fixed closed loop policy of actions defining an option $o \in O$. The purpose of such an option is to accomplish the objective assuming that the opponent does not react. Inclusion of such an option in the final option space is important because an opponent can switch between being proactive and inactive. If the model of the environment is known along with the properties of an inactive opponent, then the higher level greedy option can be accurately designed, otherwise a reasonable estimate of Eq. 9 is adequate. The case-wise implication of a higher level greedy option is:

**Case I:** Option which leads to longest expected failure avoidance against an inactive opponent. For example, a goal keeper guarding the goal, in case of an inactive attacker, executes the higher level greedy option of staying at the projected entry point of a free moving ball. This option incurs the minimum risk of the ball entering the goal, thus maximizing the expected reward.

**Case II:** Option which leads to fastest expected goal achievement against an inactive opponent. For example, an attacker attempting to take the ball away from an inactive defender, executes an intercept option, where it reaches the ball as fast as possible and tackles the defender.

**Case III:** Option which leads to fastest expected goal achievement against an inactive opponent. Since opponent is inactive, the failure state cannot be approached by it. Hence the higher level greedy option is chosen as the goal satisfaction option without entering a failure state. For example, an attacker dribbling the ball towards the goal with an inactive defender obstructing its path, will execute the higher level greedy option of dribbling fast towards the goal while maintaining a safe minimal distance from the inactive defender.

**Definition 5** *Deception option* is a set of options, $o_{d,i} \in O$ being the $i^{th}$ option in the set, complementary in nature to the higher level greedy option, together with which it forms the option space $O$ of the agent. For a given set of states $s \in S_{o_{d,i}}$, dependent on the opponent’s policy $\mu_{\text{opponent}}$, a final optimal policy $\mu_{\text{agent}}$ over the option space, the state option transition model $p^\mu_{os}$, a deception option $o_{d,j}$, when executed at time $t$ indicated by $\xi(o_{d,i}, s, t)$, should be able to do one of the following:

1. Result in an increased expected state-option value for the higher level greedy option $o_g$ taken at time $(t+k)$ under the final optimal policy $\mu_{\text{agent}}$, as compared to the previous state-option value at time $t$.

\[
E(Q^{\mu_{\text{agent}}}(s_{t+k}, o_g) \mid \xi(o_{d,i}, s, t)) = \sum_{s' \in S} p^\mu_{ss'} Q^{\mu_{\text{agent}}}(s', o_g) > Q^{\mu_{\text{agent}}}(s, o_g) \quad (10)
\]

2. Result in an increased expected state-option value for a deception option $o_{d,j}, j \neq i$ taken under the final optimal policy $\mu_{\text{agent}}$, as compared to the previous state-option value at time $t$.

\[
E(Q^{\mu_{\text{agent}}}(s_{t+k}, o_{d,j}) \mid \xi(o_{d,i}, s, t)) = \sum_{s' \in S} p^\mu_{ss'} Q^{\mu_{\text{agent}}}(s', o_{d,j}) > Q^{\mu_{\text{agent}}}(s, o_{d,j}) \quad (11)
\]
This implies that successful deceptions are those which balance strengthening the greedy options and strengthening other deceptions. For example, an attacker approaching a goal keeper with the ball may execute deception options of the nature of charging the goal keeper, or the left or right post. Such options may increase the option-value of kicking to the clearest region by drawing out the goal keeper from the goal or biasing it to one particular side. The deceptions may even increase each other’s option value - charging the left post may result in an increased guard towards the right post which may reinforce charging the right post as a deception and so on until the attacker has a clear line to shoot.

Choosing a successful deception option depends on the knowledge of both an opponent’s policy as well as the final optimal policy, neither of which is available. Given a learning algorithm (McGovern and Sutton, 1998), and a suitably restrictive option space containing possible successful deceptions, an optimal policy can be arrived at such that Eq. 10 and Eq. 11 are approximately satisfied. Hence the task translates to choosing a restrictive deception option space, such that the learning algorithm is able to reinforce the successful deceptions. Thus the criteria that the set of deception options must satisfy are:

1. The set must contain possible successful deception options against all strong opponents of interest.
2. The set must be as restrictive as possible to speed up learning.
3. In cases where the model is unknown, Eq. 10 and Eq. 11 must be approximated by suitable criteria for judgement.

To meet these criteria, a theorem is established later on in the section which acts as a guideline for choosing deception options.

**Definition 6** *Counter of an option, \( \kappa(o) \), \( o \in O \), is the most effective option that can be executed by an opposing team, i.e., it yields the minimum expected option value of the subsequent state option pair.*

\[
\kappa(o) = \arg \min_{o \in O} E(Q^{\mu_{agent}}(s_{t+k}, o) \mid \xi(o_{opp}, s, t)), \forall s \in S
\]  

(12)

where \( \mu_{agent} \) is the policy being followed by the agent.

For example, the counter of a pass from players maintaining possession is blocking the line of the pass. The counter by a single attacker when a defender is trying to intercept the ball, is dribbling around it. It is seen that counters are context dependent, i.e. the counter to an intercept may be a pass if there are one or more teammates or dribbling away otherwise. In this context, a counter is not designed but is rather considered to be a best case response from the opponent. This consideration is used to improve the worst case scenario.

**Definition 7** *Perfectly aggressive opponent* is an opponent which executes any policy \( \mu_{agg} \) over the counter of the higher level greedy option \( \kappa(o_g) \).

**Definition 8** *Perfectly reactive opponent* is an opponent which executes any policy \( \mu_{react} \) over a set of options containing the counters of all deception options \( \bigcup_{o_d,i \in O} \kappa(o_{d,i}) \).
Definition 9  Weakness of an option $o \in O$ is a set of states, $S_{weak}$, for which if an agent executes the option and the opponent executes $\kappa(o)$, the probability of entering a failure state is greater than a threshold.

$$P(s \in S_{weak}, o_{agent} = o, o_{opp} = \kappa(o), s' \in S_f) > P_{thresh}$$ (13)

In Case I, if states are represented in polar form $(r, \theta_1, \theta_2, \cdots)$ as shown in Fig. 2 where $r$ is the distance from failure state, $r = \min ||s - s_f||, s_f \in S_f$ and $\theta_i$ is an angular variable expressing the weakness of deception option $o_{d,i}$, then

1. Region of weakness $R_w$, a region with radius $r_w$ is defined as the weakness of the higher level greedy option $o_g$.

$$P(|r| < r_w, o_{agent} = o_g, o_{opp} = \kappa(o_g), |r'| = 0) > P_{thresh}$$ (14)

where $r' = \min ||s' - s_f||, s_f \in S_f$.

2. Sector of weakness $\Phi_i$, a sector in dimension $\theta_i$ having bounds $[\phi_{i,1}, \phi_{i,2}]$ is defined as the weakness of deception option $o_{d,i}$.

$$P(\phi_{i,1} \leq \theta_i \leq \phi_{i,2}, o_{agent} = o_{d,i}, o_{opp} = \kappa(o_{d,i}), |r'| = 0) > P_{thresh}$$ (15)

For example if midfielders are trying to keep possession of the ball from attackers, then $r$ would represent the euclidean distance of the nearest attacker to the ball. For the deception option $o_{d,i}$ corresponding to a pass to closest teammate, $\theta_i$ would be the angle between the attacker, the possessor of the ball and the teammate receiving the pass.

For case II, weakness is always associated with the opponent, which is subjected to the same conditions as an agent in case I. For case III, when aiming at a goal state, the agent has the same definition as that of an agent in case II. To incorporate the failure state, the polar representation $(r, \rho, \gamma_1, \gamma_2, \cdots, \theta_1, \theta_2, \cdots)$ is used as shown in Fig. 3 where $\rho = \arctan(\eta d), d = \min ||s - s_f||, s_f \in S_f$, $\eta$ is a scaling constant and $\gamma_i$ contains the weakness of deception option $o_{d,i}$ associated
with avoiding the failure state. With reference to case I, \( r \) is the distance to goal and \( \theta_i \) contains the weakness of the counter to the agent’s deception used to reach the goal. Thus an incorporation to the sector of weakness is made by the inclusion of \( \Psi_i \), a sector in dimension \( \gamma_i \) having bounds \([\psi_{i,1}, \psi_{i,2}]\) which is defined as the weakness of deception option \( o_{d,i} \).

\[
P(\psi_{i,1} \leq \gamma_i \leq \psi_{i,2}, o_{agent} = o_{d,i}, o_{opp} = \kappa(o_{d,i}), \rho = 0) > P_{thresh} \quad (16)
\]

To incorporate \( n \) sectors of weakness for an option, the option can be repeated \( n \) times.

**Definition 10** A local subgoal is defined as a nonterminal goal for agents defending a failure state. If this subgoal exists, then

\[
P(r’ > r \mid o_{agent} = o_{sg}) > P_{thresh}, s \in S_{subgoal}, o_{sg} \in O \quad (17)
\]

A local subgoal for an agent is a local subfailure for the opponent and vice-versa.

**Definition 11** A fair game is defined as a scenario where both teams have a sufficient number and nature of options without handicapping each other, such that the average reward over an episode depends primarily on their policies. To ensure a fair game both teams must have the same number of goals or local subgoals, as well as the same number of failures or local subfailures.

\[
N(S_g, agent) + N(S_{sg}, agent) = N(S_g, opponent) + N(S_{sg}, opponent) \\
N(S_f, agent) + N(S_{sf}, agent) = N(S_f, opponent) + N(S_{sf}, opponent) \quad (18)
\]

where \( N(S) \) is the number of distinct categories in \( S \).

Equal number of goals or local subgoals, and failures or local subfailures ensures that a game is balanced from the perspective of the objective of both teams. This definition broadly combines Case I and II under the category of Case III. Hence,
the strength of a team is determined by the balance between aiming for a goal or local subgoal and avoiding failure or local subfailure. Thus a set of deceptions must be able to engage the opponent in a fair game.

**Lemma 1** Let \( N \) be the number of terminal failure states and \( M \) be the number of terminal goal states for an agent facing an opponent. Then the necessary conditions for a fair game are:

i. If \( N > M \), \((N - M)\) extra agents are required

ii. If \( N = M \), the game is fair

iii. If \( N < M \), \((M - N)\) opponents are required

**Proof (Sketch)** According to Eq. 18, the number of goals or local subgoals, and failures and local subfailures must be equal for both teams. For agents defending a failure state, every additional agent can create a local subgoal, thus creating a local subfailure for the opponent. Similarly for agents pursuing a goal, every additional opponent can engage an agent, thus reducing the subgoals by 1.

For \( N > M \), number of required subgoals

\[ N(S_{sg}, \text{agent}) = (N(S_{g}, \text{opponent}) + N(S_{sg}, \text{opponent})) - N(S_{g}, \text{agent}) \]

\[ = N + 0 - M = N - M \]

Therefore if \((N - M)\) extra agents are added, this requirement is met. To check the equality of failure and subfailures:

\[ N(S_{sf}, \text{agent}) + N(S_{f}, \text{agent}) = 0 + N = N \]

\[ N(S_{sf}, \text{opponent}) + N(S_{f}, \text{opponent}) = N - M + M = N \]

For \( N < M \), the situation is reversed as the opponents have a handicap. Thus similarly \((M - N)\) opponents are required.

In the scenario of \( N = M \),

\[ N(S_{g}, \text{agent}) + N(S_{sg}, \text{agent}) = M + 0 = M \]

\[ N(S_{g}, \text{opponent}) + N(S_{sg}, \text{opponent}) = N + 0 = N \]

\[ N(S_{f}, \text{agent}) + N(S_{sf}, \text{agent}) = N + 0 = N \]

\[ N(S_{f}, \text{opponent}) + N(S_{sf}, \text{opponent}) = M + 0 = M \]

**Lemma 2** All opponents of interest lie on the line joining a perfectly aggressive opponent to a perfectly reactive opponent, i.e,

\[ \mu_{opp} = \alpha \cdot \mu_{agg} + (1 - \alpha) \cdot \mu_{reactive} \]

\[ \alpha : S \rightarrow \{0, 1\} \]  \hspace{1cm} (19)

**Proof (Sketch)** Let \( \mu_{opp} \) be the policy of the opponent over a set of options \( O \). Let \( s \in S \) and \( o_{opp} \in O_s \) such that \( \mu_{opp}(s, o_{opp}) \neq 0 \). Performance of this option against \( o \) executed by the agent increases as the value of the agent’s option decreases. Thus performance is expressed as:

\[ -E(Q^{o_{agg}}(s_{t+k}, o) \mid \xi(o_{opp}, s, t)) \]  \hspace{1cm} (20)
where $Q^\mu(s,o)$ is the option value for the agent and $\mu_{agent}$ is the policy followed by the agent.

If agent is executing a higher level greedy option $o_g$, then an opponent of interest will execute an option $o_{opp}$
\[
o_{opp} = \arg \max_{o' \in O} -E(Q^\mu_{agent}(s_{t+k}, o_g) \mid \xi(o', s, t))
\]
\[
= \arg \min_{o' \in O} E(Q^\mu_{agent}(s_{t+k}, o_g) \mid \xi(o', s, t))
\]
\[
= \kappa(o_g)
\]

Hence such opponents can be represented by a policy over $\kappa(o_g)$.

If an agent is executing deception option $o_{d,i}$, then an opponent of interest will execute an option $o_{opp}$
\[
o_{opp} = \arg \max_{o' \in O} -E(Q^\mu_{agent}(s_{t+k}, o_{d,i}) \mid \xi(o', s, t))
\]
\[
= \arg \min_{o' \in O} E(Q^\mu_{agent}(s_{t+k}, o_{d,i}) \mid \xi(o', s, t))
\]
\[
= \kappa(o_{d,i})
\]

These opponents can be represented by a policy over $\bigcup_{o_{d,i} \in O} \kappa(o_{d,i})$.

Any policy over $\kappa(o_g)$ is represented as $\mu_{agg}$ while any policy over $\bigcup \kappa(o_{d,i})$ is represented as $\mu_{react}$. Let $S_{agg}$ be a set for which $\mu_{opp}(s, \kappa(o_g)) \neq 0$. Then $\alpha(s) = 1, \forall s \in S_{agg}$ and $\alpha(s) = 0$ otherwise.

**Theorem 1** To engage the opponent in a fair game, deception should be chosen such that the opponent cannot approach an agent’s failure or local subfailure while simultaneously blocking the agent’s goal or local subgoal.

**Corollary 1** Deception can be chosen as the minimal set of options such that
1. Weakness of $o_g$ can be exited or weakness area of $\kappa(o_g)$ can be entered.
2. $\exists o \in O$ such that weakness area of $o_{d,i}$ can be exited or weakness area of $\kappa(o_{d,i})$ can be approached.

**Proof (Sketch)** From definition 11, in a fair game scenario both teams have equal number of goals or local subgoals and failures or local subfailures. From lemma 2, an opponent of interest lies on the line joining a perfectly reactive opponent and a perfectly aggressive opponent. Thus the set of options of interest executed by the opponent is $\bigcup \kappa(o), o \in O$.

Case I: As mentioned in definition 9, $(r, \theta_1, \theta_2, \cdots)$ is the polar representation of the state space. Let the deception option for entering the subgoal be $o_{d,i}$. If $\not\exists o \in O$ to exit $\Phi_i$, while opponent enters $R_w$, then for any $\mu_{agent}$, the failure state will be reached. Since this is independent of the agent’s policy, a fair game can no longer exist as shown in Fig. 4.

Case II: The polar representation $(r, \theta_1, \theta_2, \cdots)$ represents the region of weakness of the opponent, in the scenario of an agent in Case I, with an option space $\bigcup \kappa(o), o \in O$. Let the counter option $o_{opp,i} = \kappa(o_{d,i})$. The weakness sector for this counter option is represented by $\Phi_i$. If $\not\exists o \in O$ to enter $\Phi_i$, while the opponent blocks $R_w$, then for any $\mu_{agent}$, the opponent will continue sustenance by entering the subgoal state, while the goal state will not be reached as shown in Fig. 5.
Fig. 4: (a) Deception options should be able to exit weakness in Case I (b) Possible state trajectory incase (a) is not met.

Fig. 5: (a) Deception options should be able to enter weakness of deception of opponent while it blocks the goal in Case II (b) Possible state trajectory incase (a) is not met.

Case III: The polar representation is \((r, \rho, \gamma_1, \gamma_2, \cdots, \theta_1, \theta_2, \cdots)\). If \(\not\exists o \in O\) to enter \(R_w\), while opponent enters \(\Psi\), then for any \(\mu_{agent}\), the failure state will be reached as shown in Fig. 6a. If \(\not\exists o \in O\) to enter \(\Phi_i\), while opponent blocks \(R_w\), then for any \(\mu_{agent}\), the opponent will continue sustenance by entering the subgoal, while the goal state will not be reached as shown in Fig. 7a.

Thus the theorem is proved by contradiction.
3 Reinforcement Learning Framework

The learning framework used in this paper is single agent SMDP learning. Thus the SMDP version of Sarsa($\lambda$) is used with linear tile coding approximator (Albus, 1981; Rummery and Niranjan, 1994; Sutton and Barto, 1998).
3.1 Sarsa(\(\lambda\))

Sarsa(\(\lambda\)) is an on-policy learning, which means that an agent will interact with the environment, thus updating its policy based on the actions it takes. At the same time, the agent updates its policy by changing \(Q(s, a)\). The basic form of Sarsa(\(\lambda\)) with eligibility traces (Sutton and Barto, 1998; Stone et al, 2005) is

\[
\begin{align*}
\text{Initialize } Q(s, a) &\text{ arbitrarily and } e(s, a) = 0 \text{ for all } s, a \\
\text{Repeat (for each episode):} & \\
& \quad \text{Initialize } s \\
& \quad \text{Choose } a \text{ from } s \text{ using policy derived from } Q \\
& \quad \text{Repeat (for each step of episode):} \\
& \quad \quad \text{Take action } a, \text{ observe reward } r, \ s' \\
& \quad \quad \text{Choose } a' \text{ from } s' \text{ using policy derived from } Q \\
& \quad \quad \delta \leftarrow r + \gamma Q(s', a') - Q(s, a) \\
& \quad \quad e(s, a) \leftarrow e(s, a) + 1 \\
& \quad \text{For all } s, a: \\
& \quad \quad Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a) \\
& \quad \quad e(s, a) \leftarrow \gamma \lambda e(s, a) \\
& \quad \quad s \leftarrow s'; a \leftarrow a' \\
& \quad \text{until } s \text{ is terminal}
\end{align*}
\]

where \(\alpha\) is the learning rate and \(\gamma\) is the discount factor weight on future rewards. The values stored by \(e(s, a)\) are known as eligibility traces which represent the credit the past action choices receive for current rewards. The parameter \(\lambda\) determines the proportion of the credit. The policy followed is \(\epsilon\) – greedy, where the probability of taking a random action is \(\epsilon\) and the remaining probability is for taking the current optimal action. A balance between exploration and exploitation is thus maintained. Implementation of this algorithm in the SMDP context of RoboCup Simulation is presented by Stone et al (2005).

3.2 Function Approximation

The function approximator most commonly used in the application is tile-coding (Albus, 1981). Tile coding is a method of discretization of the continuous state variables by taking an arbitrary group of such variables and laying infinite axis-parallel tilings over them. A particular state can be contained by many such tiles which make up a feature set. The tiles vote on the actions individually thus determining the action values. Even though the state variables are bounded, the number of tiles can potentially be very high to ensure accuracy of approximation. Such sparse data sets are handled using open-addressed hash-coding, storing only non-zero elements.

4 Experimental Setup and Results

The experiments were run on the RoboCup Soccer Simulator Server version 11.1.0 using the C development environment created by (Stone et al, 2005). The primitive actions used have been implemented by the CM-United 99 team (Stone et al, 2000).
For every case, a suitable benchmark example was taken and deception options were constructed according to the guidelines presented in the paper. The final performance was compared to that of hand-coded policies as well as specialised learning frameworks to judge the generic nature of deception options. Comparisons were made in terms of final result over the limitations of the learning algorithm and the function approximator, as well as the time taken to learn the policies.

4.1 Case I: Keepaway

Keepaway (Stone et al, 2005) is a subtask of RoboCup soccer, in which the team of agents, known as the keepers, is trying to maintain possession of the ball while the team of opponents, the takers, is trying to gain possession. A snapshot of a keepaway game in progress is shown in Fig. 8 - there are 3 keepers and 2 takers playing in a 20m x 20m area. The keepers are numbered $K_i$, $K_1$ being the keeper nearest to the ball while the others are in increasing order of distance from $K_1$. The takers are numbered as $T_i$ in increasing order of distance from $K_1$. Since the agents have a failure state, the number of keepers is 1 more than the number of takers for a fair game.

Each episode begins with the trainer placing the ball in a designated starting position with players semi-randomly deployed in the field. The takers are placed in one corner while the keepers are placed in the other corners. The ball is placed near one of the keepers. The trainer manages the play and ends it when the takers gain possession or the ball goes out of bounds.

In this scenario, $S_f$ is defined as the state in which the ball is in the possession of the taker, i.e, in the taker’s kickable area. The reward structure of the agent is chosen as follows:

$$r_{t+1}(s_t, a_t, s_{t+1}) = \begin{cases} 
\tau_{t+1} - \tau_t & \text{if } s \notin S_f \\
0 & \text{otherwise}
\end{cases}$$
<table>
<thead>
<tr>
<th>Option</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>HoldBall</td>
</tr>
<tr>
<td>Deception</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Block, Mark</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Policy</th>
<th>Episode duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always Hold</td>
<td>2.9s ± 1.0s</td>
</tr>
<tr>
<td>Random</td>
<td>5.3s ± 1.8s</td>
</tr>
<tr>
<td>Hand-coded</td>
<td>13.3s ± 8.3s</td>
</tr>
<tr>
<td>Learnt (Benchmark)</td>
<td>15.69s ± 2.81s</td>
</tr>
<tr>
<td>Learnt (using Deception)</td>
<td><strong>13.82s ± 2.17s</strong></td>
</tr>
</tbody>
</table>

where \( t \) denotes SMDP timestep and \( \tau \) denotes simulator time.

Since the scope of the paper is restricted to learning for a single agent, learning was initiated only when a keeper possessed the ball. While constructing the minimal option space, the higher level greedy option \( o_g \) was designed according to Eq. 9. Thus \( o_g \) was selected as the option HoldBall which is holding the ball at the current position while minimizing tackle probabilities. The counter \( \kappa(o_g) \) is the Intercept option. According to Corollary 1, \( o_{d,i} \) was selected as a Pass to nearest teammate \( K_{i+1} \). The counter \( \kappa(o_{d,i}) \) is Block if receiver is nearby and Mark if the receiver is further back. Also, keepers which are not in possession of the ball execute the GetOpen option to move to a less crowded location. Thus while \( o_{d,i} \) can exit \( R_w \), the combination of HoldBall and GetOpen can exit \( \Phi_i \). There are also cases when state is in \( \Phi_1 \), but not in \( \Phi_2 \). In such cases the alternative Pass option also exits the sector of weakness, the exit being sharp jumps as learning resumes only after the Pass is completed. Table 1 summarises these options.

The state variables chosen were:

- \( \text{dist}(K_1, C), \text{dist}(K_2, C), \text{dist}(K_3, C) \)
- \( \text{dist}(T_1, C), \text{dist}(T_2, C) \)
- \( \text{dist}(K_1, K_2), \text{dist}(K_1, K_3) \)
- \( \text{dist}(K_1, T_1), \text{dist}(K_1, T_2) \)
- \( \min(\text{dist}(K_2, T_1), \text{dist}(K_2, T_2)) \)
- \( \min(\text{dist}(K_3, T_1), \text{dist}(K_3, T_2)) \)
- \( \min(\text{ang}(K_2, K_1, T_1), \text{ang}(K_2, K_1, T_2)) \)
- \( \min(\text{ang}(K_3, K_1, T_1), \text{ang}(K_3, K_1, T_2)) \)

The takers implemented a hand-coded policy (Stone et al, 2005). The Sarsa(\( \lambda \)) algorithm used the following parameters: \( \alpha = 3e-4, \epsilon = 0.01 \) and \( \lambda = 0 \). The CMAC function approximator used 32 tilings for every variable, using a total of 416 tilings at a resolution of 3.0m and 10 degrees.

In Fig. 9 the learning curve of the keepers using the deception option space is shown. The graph levels at 11 seconds after approximately 8 hours of training, following which the time gradually reaches 13.5 seconds after 100 hours of training. Table 2 compares this time with other benchmark times. The time is comparable to that of a carefully hand-tuned policy but with much less deviation. This is
Fig. 9: The episode duration of keepaway with training hours (the graph is passed through a LPF with a sliding window of 1000 episodes). The final mean time is approximately 13.5 s.

Table 3: Robustness of Learnt Policy to various Taker Policies

<table>
<thead>
<tr>
<th>Taker Policy</th>
<th>Episode duration</th>
<th>Deviation from benchmark time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always Intercept</td>
<td>13.8s</td>
<td>+0s</td>
</tr>
<tr>
<td>Policy 1</td>
<td>14.8s</td>
<td>+1.0s</td>
</tr>
<tr>
<td>Policy 2</td>
<td>11.9s</td>
<td>−1.9s</td>
</tr>
<tr>
<td>Policy 3</td>
<td>9.7s</td>
<td>−4.1s</td>
</tr>
<tr>
<td>Policy 4</td>
<td>15.1s</td>
<td>+1.3s</td>
</tr>
</tbody>
</table>

Table 3 shows the performance of the framework against various taker policies using the learnt weights against the Always Intercept policy of takers as the initial weights. The difference between the strongest taker policy, Policy 3, and the Always Intercept policy is within the acceptable margins, thus indicating that the algorithm is able to exploit the weakness of any opponent.

In Fig. 10 the state space trajectory of a keepaway episode is shown to investigate the sector of weakness of the Pass option. The sector of weakness $\Phi_i$ in the dimension $\text{ang}(K_i, K_1, T)$ for both Pass options is chosen as $[0, 15]$ degrees for an acceptable $P_{\text{thresh}}$. Whenever takers enter the sector of weakness, the keepers exit it either using the HoldBall and GetOpen combination (shown by a slow moving trajectory exiting the weakness) or by using the alternative Pass (shown by a long line segment).

To observe the balance between the higher level greedy function and deception, a subscene of a keepaway episode is shown in Fig. 11. The maximum option-value of $K_1$ is shown as a function of the position of $T_1$. In Fig. 11a, both takers approach $K_1$. If the passing zone to $K_3$ is occupied within a threshold distance, a pass is indicative of the fact that the algorithm is a very effective substitute to laboriously hand-tuned algorithms.
preferred to $K_2$. This threshold distance increases in Fig. 11b as $T_2$ comes closer to $K_1$. It is also seen that an attack from the direction opposite to the other keepers leads to a higher probability of a pass. As either taker nears $K_1$ in Fig. 11c, any occupancy along the $K_1K_3$ line indicates an imminent tackle and thus a pass is made. After the pass in Fig. 11d, the situation resets to a scenario similar to Fig. 11a, with a pass to $K_3$ as a more viable option. It is noted that threshold distance for passing is much less than in previous scenes as the risk of this pass is higher.
4.2 Case II: Takeaway

Takeaway is a subtask of RoboCup soccer, in which the team of agents, known as the *takers*, is trying to gain possession of the ball while the team of opponents, the *keepers*, is trying to maintain possession. The environment and associated rules are exactly the same as in keepaway, except that the learning algorithm is executed by the takers. Since the agents have a goal state, the number of takers is 1 less than the number of keepers for a fair game.

In this scenario, $S_g$ is defined as the state in which the ball is in the possession of the taker, i.e., in the taker’s kickable area. The reward structure of the agent is chosen as follows:

$$r_{t+1}(s_t, o_t, s_{t+1}) = \begin{cases} 
1 & \text{if } s \in S_g \\
0 & \text{otherwise} 
\end{cases}$$

where $t$ denotes SMDP timestep.

Learning was initiated for both takers. The higher level greedy option $o_g$ was selected as Intercept which means directly heading towards the ball to retrieve it for which $\kappa(o_g)$ is the Pass option. The deception $o_{d,i}$ was selected as Block the line to teammate $K_{i+1}$, and Mark the teammate $K_{i+1}$. For both deceptions, $\kappa(o_{d,i})$ is HoldBall as it minimizes the probability of losing the ball and hence the expected reward of $o_{d,i}$. The options terminate based on an adequate timeout, to ensure that the effect on execution of the option is reflected. Both the Block
Table 4: Designed option space for Takeaway

<table>
<thead>
<tr>
<th>Option</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>Intercept</td>
</tr>
<tr>
<td></td>
<td>PassK2, PassK3</td>
</tr>
<tr>
<td>Deception</td>
<td>Block</td>
</tr>
<tr>
<td></td>
<td>HoldBall</td>
</tr>
<tr>
<td>Mark</td>
<td>HoldBall</td>
</tr>
</tbody>
</table>

Fig. 12: The episode duration of takeaway with training hours (the graph is passed through an averaging filter with a sliding window of 40 episodes). The final mean time is approximately 7.91 s

and the Mark functions retrieve the ball if it enters a threshold proximity. The reason for inclusion of both Block and Mark is that the Block function obstructs the passing line maintaining the same distance to the ball. If the radial distance needs to be increased to increase the probability of intercepting a pass, the option Mark assists in doing so. Thus while $o_g$ can enter $R_w$, the combination of Block and Mark can enter $\Phi_i$, the sector of weakness of $opp,i = \kappa(o_g)$.

The state variables chosen were:

- $\text{dist}(K_1, K_2), \text{dist}(K_1, K_3)$
- $\text{dist}(K_1, T_1), \text{dist}(K_1, T_2)$
- $\min(\text{dist}(K_2, T_1), \text{dist}(K_2, T_2))$
- $\min(\text{dist}(K_3, T_1), \text{dist}(K_3, T_2))$
- $\min(\text{ang}(K_2, K_1, T_1), \text{ang}(K_2, K_1, T_2))$
- $\min(\text{ang}(K_3, K_1, T_1), \text{ang}(K_3, K_1, T_2))$

The Sarsa($\lambda$) algorithm used the following parameters: $\alpha = 3e - 4$, $\epsilon = 0.05$ and $\lambda = 0.5$. The CMAC function approximator used 32 tilings for every variable, using a total of 256 tilings at a resolution of 3.0m and 10 degrees.

In Fig. 12 the learning curve of takers against a fixed keepaway policy (as learnt in Case I) is displayed. After approximately 2 hours of learning the graph settles at
Fig. 13: State space trajectory of a takeaway episode which lasted 8.25s.

Table 5: Benchmark and Learnt times for Takeaway

<table>
<thead>
<tr>
<th>Policy</th>
<th>Episode duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>14.53s ± 0.74s</td>
</tr>
<tr>
<td>Hand-coded (Unrestricted State Variables)</td>
<td>5.44s ± 0.9s</td>
</tr>
<tr>
<td>Hand-coded (Restricted State Variables)</td>
<td>7.26s ± 1.31s</td>
</tr>
<tr>
<td>Learnt</td>
<td>7.91s ± 0.81s</td>
</tr>
</tbody>
</table>
Fig. 14: Left column: Maximum option value of $T_1$ as a function of position of $T_1$. Right Column: Maximum option value of $T_2$ as a function of position of $T_2$, in a takeaway subscene
around 8.5s and decreases slowly after. The relative speed of learning this task as compared to keepaway is indicative of the difficulty of the two cases. Table 5 shows that the hand-coded policy with no restrictions had a more improved time as it was specific for a 3 vs 2 case. The more general hand-coded policy with fine tuned weights resulted in an almost similar time to the learnt policy. The performance of the takers implementing deception against varied Keeper policies as shown in Table 6 indicates that the weaknesses of these policies are exploited.

The sector of weakness of the opponent is examined again in Fig. 13. Compared to Fig. 10, the state trajectory spent a higher percentage of time in the weakness area. This is due to the deception options invoked by the taker. The trajectory also exits the area of weakness less frequently using a Pass, which indicates that either area of weakness is occupied as the agents approach the goal.

In Fig. 14 a subscene of a takeaway episode is captured, where the balance between deception option and higher level greedy option is illustrated. In Fig. 14a, $T_1$ is divided between blocking $K_1K_2$ and marking $K_2$, where marking is preferred if both passing zones are free. Otherwise, blocking of $K_1K_2$ is preferred because it leads to a higher probability of making $K_1$ pass the ball, while marking maintains a considerable distance from $K_1$. Since the line $K_1K_3$ in Fig. 14b is already occupied, $T_2$ proceeds primarily to block $K_1K_2$. In Fig. 14c, $T_1$ is driven towards the passing cone between $K_1$ and $K_2$ which indicates that it will intercept along the cone. Deception is initiated in Fig. 14d to force a pass along the line $K_1K_3$. $T_2$ is driven to block $K_1K_2$. In Fig. 14e, $T_1$ continues to balance blocking the line $K_1K_2$ and going to intercept while $T_2$ is driven back to block $K_1K_3$ in Fig. 14f. As both pass zones are covered in Fig. 14g, $T_1$ is at the threshold distance to make an interception. When the pass is finally made in Fig. 14h, $T_2$ intercepts it.

4.3 Case III: Aggressive Defense

The Aggressive Defense tactic, known as ADB, is used by a defender to prevent an attacker from moving ahead by interrupting its motion or stealing the ball. The term "aggressive" is used to denote that the tactic leads to a confrontation between the attacker and the defender as opposed to the defender falling back towards his goal. This tactic was implemented by Riedmiller et al (2009) using batch learning with a neural network function approximator and a model of the environment; however, for the purpose of this experiment the Sarsa($\lambda$) algorithm and tile-coding will be used. The difficulty in implementing the tactic is greater than that of the other cases, as the scenario is heavily dependent on the opponent, implying that the options selected will have to be more generic. Hand-coded behaviours may be
too specialised, and do not have the required flexibility to exploit the weakness of the opponent. The confrontations between defender and attacker may be critical to the outcome of the game.

The environment is as follows:

1. The focus region is a bounded $20m \times 20m$ square. If this zone is exited, the episode stops.
2. The goal direction in this case is on the left side (direction in which the attacker is moving).

The episode is deemed successful if the defender acquires the ball. The defender has failed if the opponent retaining the ball position has over-run the agent and escaped the bounding box in the direction of the goal. Besides the clear success and failure states, semi-success and semi-failure states are introduced from a practical perspective. In the theory discussed, these states are same as terminal states, but with diminished magnitude of rewards. If the attacker kicks away the ball because of interference with its plan, or because of a panic kick to prevent imminent interception, the state is terminal as the ball leaves the bounding box and it is termed a semi-success, $S_{sg}$. In the event of a time-out, the episode is deemed a semi-failure, $S_{sf}$ as the defender was not successful in interfering. The reward structure of the agent was chosen as follows:

$$r_{t+1}(s_t, a_t, s_{t+1}) = \begin{cases} 
8 & \text{if } s \in S_{g} \\
1 & \text{if } s \in S_{sg} \\
-5 & \text{if } s \in S_{f} \\
-1 & \text{if } s \in S_{sf}
\end{cases}$$

where $t$ denotes SMDP timestep.

The learning set as shown in Fig. 15, consists of two semi circles along which the attacker can lie, with the defender in the middle. The radius of the semi-circle facing the opponent’s side is $5m$, while that of the agent’s side is $3m$. The ball
is placed randomly in the opponent’s kickable area with zero velocity, while the initial velocities of both agent and opponent are chosen randomly.

The state space was kept relatively small consisting of essential variables:

1. Distance between defender to attacker
2. Distance between attacker and left boundary
3. Angle between goal attacker and defender
4. Angle between attacker orientation and goal
5. Angle between ball attacker and defender

The higher level greedy option was chosen as Intercept. The counter $\kappa(\omega)$ can then be one of four symmetrical options - Dribble angled left, Dribble angled right, Sidestep left and Sidestep right - or in certain states a Panic Kick. Sidestep refers to temporarily lobbing the ball around the opponent and intercepting it behind him. The area of weakness of these actions can be entered by executing a Headfake - Dash to 4m left or Dash to 4m right. The counter of these options is mirroring the Headfake. If the opponent executes a Dash to goal, the deception Dash to 4m front is chosen to enter $R_w$ of the opponent. The counter of this option is either Dash to goal or Dribbling angled left or right, depending on the relative importance of reaching the goal to losing the ball. The weakness of Dash to 4m front option cannot be exited as the attacker nears the goal, hence the intercept option can enter the weakness of Dash to 4m front, satisfying Theorem 1. The most restrictive option space is shown in table 7.

The Sarsa($\lambda$) algorithm used the following parameters: $\alpha = 3e - 4$, $\epsilon = 0.05$ and $\lambda = 0.5$. The CMAC function approximator used 32 tilings for every variable, using a total of 160 tilings at a resolution of 3.0m and 10 degrees.

In Fig. 16 this attacker follows a much simpler policy than that of the WrightEagle team (Shi et al, 2009) against which Team Brainstormers (Riedmiller et al,
Fig. 17: State space trajectory of a couple of ADB episodes.

Table 7: Designed option space for ADB

<table>
<thead>
<tr>
<th>Option</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy Intercept</td>
<td>Dribble angled left/right, Sidestep, Panic Kick</td>
</tr>
<tr>
<td>Deception Dash</td>
<td>Dash to goal, Dribble angled left/right</td>
</tr>
<tr>
<td>Deception Dash</td>
<td>Dribble angled left</td>
</tr>
<tr>
<td>Deception Dash</td>
<td>Dribble angled right</td>
</tr>
</tbody>
</table>
Table 8: Benchmark and Learnt times for ADB

<table>
<thead>
<tr>
<th>Policy</th>
<th>Success</th>
<th>SemiSuccess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-coded (Brainstormers vs WrightEagle)</td>
<td>38.1%</td>
<td>14.8%</td>
</tr>
<tr>
<td>Learnt Policy (Brainstormers vs WrightEagle)</td>
<td>62.3%</td>
<td>26.7%</td>
</tr>
<tr>
<td>Learnt Policy (Deception vs Hand-coded attacker)</td>
<td>32.37%</td>
<td>28.88%</td>
</tr>
</tbody>
</table>

Table 9: Robustness of Learnt Policy to various Attacker Policies

<table>
<thead>
<tr>
<th>Attacker Policy</th>
<th>Total Success</th>
<th>Deviation from benchmark success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training policy</td>
<td>61.25%</td>
<td>+0%</td>
</tr>
<tr>
<td>Policy 1</td>
<td>79.56%</td>
<td>+18.31%</td>
</tr>
<tr>
<td>Policy 2</td>
<td>59.66%</td>
<td>−1.59%</td>
</tr>
<tr>
<td>Policy 3</td>
<td>56.12%</td>
<td>−5.13%</td>
</tr>
</tbody>
</table>

2009) competes. However, given the disadvantage of not having a model as well as not having a neural-fitted iteration scheme, the performances of the learning agent can be compared as shown in Table 8. Table 9 shows that the total success is above 50% for various hand-coded attacker policies.

To investigate the sector weakness, the state space trajectory of an ADB episode is shown in Fig. 17. The sector of weakness is arrived at by using combinations of deception, while the trajectory in Fig. 17b converges to goal faster than to failure. This indicates that the counter of deception’s weakness is entered by the agent.

In Fig. 18a the main decision of the defender is between dashing to the front or to the right of the attacker. This is to balance between blocking the attacker’s path and pushing it away. When the defender is sufficiently near the attacker’s right as in Fig. 18b, a Dash to the left is invoked as a deception to move further behind the attacker while moving forward. In Fig. 18c as the attacker resumes its path towards the goal, the defender initially dashes to the front of the attacker and proceeds to intercept in Fig. 18d.

5 Conclusion and Future Work

This paper presents a framework for creating a restrictive option space in episodic adversarial SMDP to arrive at an optimal policy in a quick efficient manner using certain options known as deceptions. Since these optimal equations are dependent on the opponent’s policy as well as the agent’s final policy, suitable judging criteria are so established to ensure robustness of choice. The task of the learning algorithm is shown to use these options to estimate the weakness of an opponent’s policy and exploit it. Deception options are established as options which compensate for internal weakness, infiltrate any possible weakness of the opponent and are powerful tools for traversing the policy space effectively during the learning process.

With the focus on RoboCup tactics, the examples implemented are benchmark problems so the performance of the deception theory can be compared to other approaches. The results indicate that within a very short time, the policies arrived
at are comparable to well-tuned hand-coded policies. The deceptions are not able to match the performance of other approaches which have a model or a more powerful, specifically suited, function approximator, however it performs consistently across all scenarios. As illustrated with each example, the time spent by a user to create a framework for every policy is minimal as the guidelines are used to determine the deceptions with ease. Thus the objective of the generation of a library of tactics, without a large deal of case-specific tuning, is achieved.

The future scope of this work is to study the transient behaviour of two competing agents, each having a deception framework. To gain an advantage over an opponent, anticipation and detection of deception will be required. It is also important to investigate whether a steady compromise between competing policies will be reached or an oscillation will be sustained. Separate modules which detect these deceptions can strengthen decision making ability thus reducing the time-period of these oscillations.

Deceptions in the current framework constitute a fixed minimal set which can generate a very good result subjected to the limitations of the function approximator and the convergence of the algorithm. Once an opponent’s weakness is estimated, to further improve performance, using these deceptions as a seed, new options can be generated and evaluated, each of which can better exploit the weakness of the opponent. This can result in a great deal of flexibility and improved performance.
Acknowledgements  We would like to thank Sponsored Research and Industrial Consultancy (SRIC), IIT Kharagpur for funding the RoboCup project; Patrick Riley, Manuela Veloso and the CMUnited-99 team for the creation of basic agent behaviours that were used as a basis for our agents; Peter Stone, Richard Sutton and Satinder Singh for the Keepaway framework; Richard Sutton for the tile coding software.

References


