In Proceedings of 7th Workshop on Humanoid Soccer Robots,
IEEE-RAS International Conference on Humanoid Robots, Osaka, Japan, November 2012.

Dynamic Role Assignment using General Value Functions

Saminda Abeyruwan, Andreas Seekircher and Ubbo Visser
Department of Computer Science
University of Miami
1365 Memorial Drive, Coral Gables, FL, 33146 USA
{saminda,aseek,visser}@cs.miami.edu

***Abstract*—Collecting and maintaining accurate world knowledge in a dynamic, complex, competitive, and stochastic environment such as RoboCup 3D Soccer Simulation is a challenging task. Knowledge should be learned in real-time with time constraints. We use recently introduced Off-Policy Gradient Descent algorithms in Reinforcement Learning that illustrate learnable knowledge representations for dynamic role assignments. The results show that the agents have learned competitive policies against the top teams from RoboCup 2012 for three vs three, five vs five, and seven vs seven agents. We have explicitly used subset of agents to identify the dynamics and the semantics for which the agents learn to maximize their performance measures, and to gather knowledge about different objectives so that all agents participate effectively within the team.**

***Index Terms*—Dynamic Role Assignment Function, Reinforcement Learning, GQ(λ), Greedy-GQ(λ), Off-Policy Prediction and Control**

I. Introduction

The RoboCup 3D Soccer Simulation provides a dynamic, complex, competitive, and stochastic multi-agent environment for simulated agents to achieve goals. The simulated agents formalize their goals in two layers: 1) physical layers, where controls related to walking, kicking etc. are conducted; and 2) decision layers, where high level actions are taken to emerge behaviors. In this paper we investigate a mechanism suitable for decision layers to use recently introduced Off-Policy Gradient Decent Algorithms in Reinforcement Leaning (RL) that illustrate learnable knowledge representations to learn about *a dynamic role assignment function*.

In order to learn about an effective dynamic role assignment function, the agents need to consider the dynamics of agent-environment interactions. We consider these interactions as the agent's knowledge. If this knowledge is represented in a computational form (e.g., first-order predicate logic) an agent could infer many aspects about its interactions consistent with that knowledge. The knowledge representational forms show different degrees of computational complexities and expressiveness [1]. The computational requirements increase with the extension of expressiveness of the representational forms. Therefore, we need to identify and commit to a representational form which is scalable for on-line learning while preserving expressivity. A *human* soccer player knows a lot of information about the game before (s)he enters onto the field. This knowledge influences the outcome of the game to a great extent. In addition, human soccer players dynamically change their knowledge during games in order to achieve maximum rewards. Therefore, the knowledge of the human soccer player is to a certain extent either *predictive* or *goal-oriented*. Can a *robotic* soccer player collect and maintain predictive and goal-oriented knowledge? This is a challenging problem for agents with time constraints and limited computational resources.

We learn the role assignment function using a framework that is developed based on the concepts of Horde, the real-time learning methodology, to express knowledge using General Value Functions (GVFs) [1]. Similar to Horde's sub-agents, the agents in a team work are treated as independent RL sub-agents but the agents take actions based on their belief of the world model. The agents may have different world models due to noisy perceptions and communication delays. GVFs are constituted within the RL framework, and they are predictions or off-policy controls that are answers to questions. E.g., in order to make a prediction a question must be asked of the form "If I move in this formation, would I be in a position to score a goal?", or "What set of actions do I need to execute to block the progress of the opponent agent with the number 3?". The question addresses what to learn, and the problem of prediction or control is to learn value functions. An agent obtains its knowledge from information communicated back and forth between the agents and the agent-environment interaction experiences. GVFs are learned using recently developed Off-Policy Gradient Temporal Difference (OP-GTD) algorithms: a prediction question uses GQ(λ) [2], and a control or a goal-oriented question uses Greedy-GQ(λ) [3]. The OP-GTD algorithms possess a number of properties that are desirable for on-line learning within the RoboCup 3D environment: 1) off-policy updates; 2) linear function approximation; 2) no restrictions on the features used; 3) temporal-difference learning; 4) on-line and incremental; 5) linear in memory and per-time-step computation costs; and 6) convergent to a local optimum or equilibrium point [3], [4].

In this paper we present a methodology and an implementation to learn about a dynamic role assignment function considering the dynamics of agent-environment interactions based on GVFs. The agents ask questions and approximate value functions answer to those questions. The agents independently learn about the role assignment functions in the presence of an adversary team. Based on the interactions, the agents may have to change their roles in order to continue in the formation and maximize rewards. There is a finite number of roles that an agent can commit to, and GVFs learned about the role assignment function. We conducted all our experiments in the RoboCup 3D Soccer Simulation League Environment. It is based on the general purpose multi-agent simulator SimSpark[1]. The robot agents in the simulation are modeled based on the Aldebaran Nao[2] robots. Each robot has 22 degrees of freedom. The agents communicate with the server through message passing and each agent is equipped with noise free joint perceptors and effectors. In addition to this, each agent has a noisy restricted vision cone of 120^o. Every simulation cycle is limited to 20 ms, where agents perceive noise free angular measurements of each joint and the agents stimulate the necessary joints by sending torque values to the simulation server. The vision information from the server is available every third cycle (60 ms), which provides spherical coordinates of the perceived objects. The agents also have the option of communicating with each other every other simulation cycle (40 ms) by broadcasting a 20 $bytes$ message. The simulation league competitions are currently conducted with 11 robots on each side (22 total).

The remainder of the paper is organized as follows: In Section II we briefly discuss about knowledge representation forms and existing role assignment formalisms. In Section III we introduce GVFs within the context of robotic soccer. In Section IV we formalize our mechanisms of dynamic role assignment functions within GVFs. In Section V we present the experiment results, and finally Section VI contains concluding remarks, and future work.

II. Related Work

One goal of multi-agent systems research is the investigation of the prospects of efficient cooperation among a set of agents in real-time environments. In our research we focus on the cooperation of a set of agents in a real-time robotic soccer simulation environment, where the agents learn about an optimal or a near-optimal role assignment function within a given formation using GVFs. This subtask is particularly challenging compared to other simulation leagues considering the limitations of physical, communication, behavioral, and crowd management rules. The role assignment is a part of the hierarchical machine learning paradigm [5], [6], where a formation defines the role space. Homogeneous agents can change roles flexibly within a formation to maximize a given reward function.

Recently RL has been successfully applied in learning the keep-away subtask in RoboCup 2D [7] and 3D [8] simulated leagues. Also, in other RoboCup leagues, such as the MidSize league, RL has been applied successfully to acquire competitive behaviors [9]. One of the noticeable impact on RL is reported by the Brainstormers team, the RoboCup 2D simulation league team, on learning different subtasks. A comprehensive analysis of a general batch RL framework for learning challenging and complex behaviors in robot soccer is reported in [10]. Despite convergence guarantees, Q(λ) [11] with linear function approximation has been used in role assignment in robot soccer [12] and faster learning is observed with the introduction of heuristically accelerated methods [13]. The dynamic role allocation framework based on dynamic programming is described in [14] for real-time soccer environments. The role assignment with this method is tightly coupled with the agent's low-level abilities and does not take the opponents into consideration. On the other hand, the proposed framework uses the knowledge of the opponent positions as well as other dynamics for the role assignment function.

Sutton et al. [1] have introduced a real-time learning architecture, Horde, for expressing knowledge using General Value Functions (GVFs). Our research is built on Horde to ask a set of questions such that the agents assign optimal or near-optimal roles within formations. In addition, following researches describe methods and components to build strategic agents: [15] describes a methodology to build a cognizant robot that possesses vast amount of situated, reversible and expressive knowledge, and [16] presents a methodology to "next" in real time predicting thousands of features of the world state. GVFs are successfully used (e.g., [17]) for switching and prediction tasks in assistive biomedical robots.

III. Learnable knowledge representation for Robotic Soccer

Recently, within the context of the RL framework [11], a knowledge representation language has been introduced that is expressive and learnable from sensorimotor data. This representation is directly usable for robotic soccer as agent-environment interactions are conducted through perceptors and actuators. In this approach, knowledge is represented as a large number of *approximate value functions* each with its *own policy*, *pseudo-reward function*, *pseudo-termination function*, and *pseudo-terminal-reward function* [1]. In continuous state spaces, approximate value functions are learned using function approximation and using more efficient off-policy learning algorithms. First, we briefly introduce some of the important concepts related to GVFs. The complete information about GVFs are available in [1]–[3], [18]. Second, we show its direct application to robotic soccer.

The interpretation of the approximate value function as a knowledge representation language grounded on information from perceptors and actuators is defined as: knowledge expressed as an *approximate value function* is *true or accurate*, if its numerical values matches those of the mathematically defined *value function* it is approximating. Therefore, a value function asks a *question*, and an approximate value function is the *answer* to that question. Based on prior interpretation, the standard RL framework extends to represent learnable knowledge as follows. In the standard RL framework [11], let the agent and the world interact in discrete time steps $t = 1, 2, 3, \ldots$. The agent senses the state at each time step, $S_t \in \mathcal{S}$, and selects an action $A_t \in \mathcal{A}$. One time step later

[1]http://simspark.sourceforge.net/

[2]http://www.aldebaran-robotics.com/eng/

```
Initialize $w_0$ to 0, and $\theta_0$ arbitrary.
Choose proper (small) positive values for $\alpha_\theta$, $\alpha_w$, and set values for $\gamma(.) \in (0,1]$, $\lambda(.) \in [0,1]$.
repeat
    Initialize $e = 0$.
    Take $A_t$ from $S_t$ according to $\pi_b$, and arrive at $S_{t+1}$.
    Observe sample, $(S_t, A_t, S_{t+1})$ at time step $t$ (with their corresponding state-action feature vectors), where $\hat{\phi}_{t+1} = \phi(S_{t+1}, A^*_{t+1})$, $A^*_{t+1} = \text{argmax}_b\, \theta_t^{\mathrm{T}} \phi(S_{t+1}, b)$.
    for each observed sample do
        $\delta_t \leftarrow r(S_{t+1}) + (1-\gamma(S_{t+1}))z(S_{t+1}) + \gamma(S_{t+1})\theta_t^{\mathrm{T}}\hat{\phi}_{t+1} - \theta_t^{\mathrm{T}}\phi_t$.
        If $A_t = A^*_t$, then $\rho_t \leftarrow \frac{1}{\pi_b(A^*_t|S_t)}$; otherwise $\rho_t \rightarrow 0$.
        $e_t \leftarrow I_t\phi_t + \gamma(S_t)\lambda(S_t)\rho_t e_{t-1}$.
        $\theta_{t+1} \leftarrow \theta_t + \alpha_\theta[\delta_t e_t - \gamma(S_{t+1})(1-\lambda(S_{t+1}))(w_t^{\mathrm{T}} e_t)\hat{\phi}_{t+1}]$.
        $w_{t+1} \leftarrow w_t + \alpha_w[\delta_t e_t - (w_t^{\mathrm{T}}\phi_t)\phi_t)]$.
    end for
until each episode.
```

Fig. 1: Greedy-GQ(λ) with linear function approximation for General Value Function learning [18].

the agent receives a scalar reward $R_{t+1} \in \Re$ and senses the state $S_{t+1} \in \mathcal{S}$. The rewards are generated according to the *reward function* $r : S_{t+1} \rightarrow \Re$. The objective of the standard RL framework is to learn the stochastic action-selection *policy* $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ that gives the probability of selecting each action in each state, $\pi(s,a) = \mathcal{P}(A_t = a|S_t = s)$, such that the agent maximizes rewards summed over the time steps. The standard RL framework extends to include a *terminal-reward-function*, $z : \mathcal{S} \rightarrow R$, where $z(s)$ is the terminal reward received when the termination occurs in state s. In the RL framework, $\gamma \in [0,1)$ is used to discount delayed rewards. Another interpretation of the discounting factor is a constant probability of $1-\gamma$ termination of arrival to a state with zero terminal-reward. This factor is generalized to a *termination function* $\gamma : \mathcal{S} \rightarrow [0,1]$, where $1-\gamma(s)$ is the probability of termination at state s and a terminal reward $z(s)$ would be generated. Let G_t be the complete return from state S_t at time t, then the sum of the rewards (transient plus terminal) until termination at time T is: $G_t = \sum_{k=t+1}^{T} r(S_k) + z(S_T)$. The *action-value function* is $Q^\pi(s,a) = \mathrm{E}(G_t|S_t = s, A_t = a, A_{t+1:T-1} \sim \pi, T \sim \gamma)$, where $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \Re$. This is the expected return for a trajectory started from state s and action a, and selecting actions according to the policy π until termination occurs with γ. We approximate the action-value function with $\hat{Q} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$. Therefore, the action-value function is a precise grounded question, while the approximate action-value function offers the numerical answer.

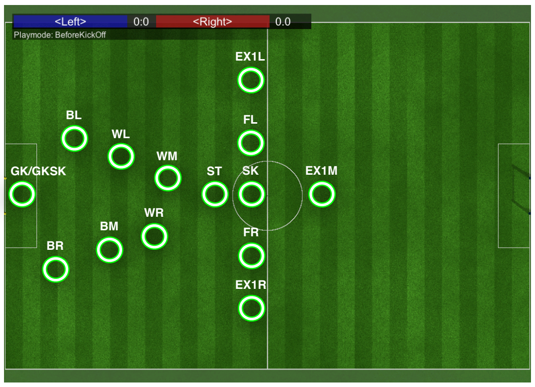
GVFs are defined over four functions: $\pi, \gamma, r,$ and z. r and z act as pseudo-reward and pseudo-terminal-reward functions respectively. γ is also in pseudo form as well. But γ is more substantive than reward functions as termination interrupts the normal flow of state transitions. In pseudo termination, the standard termination is omitted. In robotic soccer, the base problem can be defined as the time until a goal is scored by either the home or the opponent team. We can consider a pseudo-termination has occurred when the striker is changed. GVF is defined as: $Q^{\pi,\gamma,r,z}(s,a) = \mathrm{E}(G_t|S_t = s, A_t = a, A_{t+1:T-1} \sim \pi, T \sim \gamma)$. The four functions, $\pi, \gamma, r,$ and z, are the *question functions* to GVFs, which in return defines the general value function's semantics. RL agents learn an approximate action-value function, $\hat{Q}$, using the four auxiliary functions π, γ, r and z. We assume that the state space is continuous and the action space is discrete. We approximate the action-value function using a linear function approximator. We use a feature extractor $\phi : S_t \times A_t \rightarrow \{0,1\}^n$ built on tile coding [11] to generate feature vectors from state variables and actions. This is a sparse vector with a constant number of "1" features, hence, a constant norm. In addition, tile coding has the key advantage of real-time learning and to implement computationally efficient algorithms to learn approximate value functions. In linear function approximation, there exists a weight vector, $\theta \in \Re^n$, to be learned. Therefore, the approximate GVFs are defined as $\hat{Q}(s,a,\theta) = \theta^{\mathrm{T}}\phi(s,a)$ such that $\hat{Q} : \mathcal{S} \times \mathcal{A} \times \Re^n \rightarrow \Re$. Weights are learned using gradient-descent temporal-difference algorithms [18]. These algorithms learn stable and efficiently using linear function approximation from *off-policy* experiences. Off-policy experiences are generated from a *behavior policy*, π_b, that is different from the policy being learned about named as *target policy*, π. Therefore, one could learn multiple target policies from the same behavior policy.

We are interested in finding optimal policies for role assignment and henceforth we use Greedy-GQ(λ) [3], [18] algorithm for control[3]. We use linear function approximation for continuous state space, and discrete actions are used within options. The question functions are defined by: 1) $\pi : S_t \times A_t \rightarrow [0,1]$ (target policy is greedy w.r.t learned value function); 2) $\gamma : S_t \rightarrow [0,1]$ (termination function); 3) $r : S_{t+1} \rightarrow \Re$ (transient reward function); and 4) $z : S_{t+1} \rightarrow \Re$ (terminal reward function). The answer functions are defined by: 1) $\pi_b : S_t \times A_t \rightarrow [0,1]$ (behavior policy); 2) $I_t : S_t \times A_t \rightarrow [0,1]$ (interest function); 3) $\phi : S_t \times A_t \rightarrow \Re^n$ (feature-vector function); and 4) $\lambda : S_t \rightarrow [0,1]$ (eligibility-trace decay-rate function).

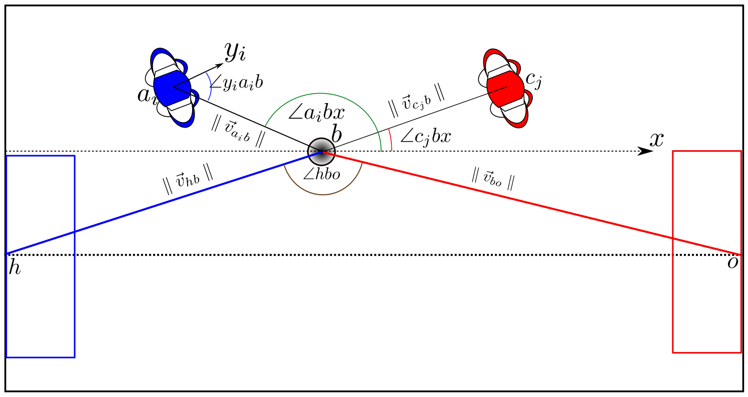
IV. Dynamic role assignment

A *role* is a specification of an internal or an external behavior of an agent. In the soccer domain, roles select behaviors of agents based on different reference criteria: the agent close to ball becomes the striker. Given a role space, $\mathcal{R} = \{r_1, \ldots, r_n\}$, of size n, the collaboration among $m \leq n$ agents, $\mathcal{A} = \{a_1, \ldots, a_m\}$, is obtained through *formations*. The ole space consists of active and reactive roles. For example, the striker is an active role and the defender could be a reactive role. Given a reactive role, there is a function, $R \mapsto T$, that maps roles to target positions, T, on the field. These target positions are calculated with respect to a reference pose (e.g., ball position) and other auxiliary criteria such as crowd management rules. A role assignment function, $R \mapsto A$, provides a mapping from role space to agent space,

[3]An implementation of Algorithm 1: https://github.com/samindaa/RLLib



(a) Primary formation, [19]



(b) State variables representation.

Fig. 2: State variable representation and the primary formation.

while maximizing some reward function. The role assignment function can be static or dynamic. Static role assignments often provide inferior performance in robot soccer [14]. Therefore, we learn a dynamic role assignment function within the RL framework using off-policy control.

A. Target Positions with the Primary Formation

Within our framework, an agent can choose one role among 13 roles. These roles are part of a primary formation, and an agent calculates the respective target positions according to its belief of the absolute ball position and the rule imposed by the 3D soccer simulation server. We have labeled the role space in order to describe the behaviors associated with them. Figure 2a shows the target positions for the role space before the kickoff state. The agent closest to the ball takes the striker role (SK), which is the only active role. Lets assume that the agent's belief of the absolute ball position is given by (x_b, y_b). Forward left (FL) and forward right (FR) target positions are offset by $(x_b, y_b) \pm (0, 2)$. The extended forward left (EX1L) and extended forward right ((EX1R)) target positions are offset by $(x_b, y_b) \pm (0, 4)$. The stopper (ST) position is given by $(x_b - 2.0, y_b)$. The extended middle (EX1M) position is used as a blocking position and it is calculated based on the closest opponent to the current agent. The other target positions, wing left (WL), wing right (WR), wing middle (WM), back left (BL), back right (BR), and back middle (BM) are calculated with respect to the vector from the middle of the home goal to the ball and offset by a factor which increases close to the home goal. When the ball is within the reach of goal keeper, the (GK) role is changed to goal keeper striker (GKSK) role. We slightly change the positions when the ball is near the side lines, home goal, and opponent goal. These adjustments are made in order to keep the target positions inside the field. We allow target positions to be overlapped. The dynamic role assignment function may assign the same role during the learning period. In order to avoid position conflicts an offset is added; the feedback provides negative rewards for such situations.

B. Roles to RL Action Mapping

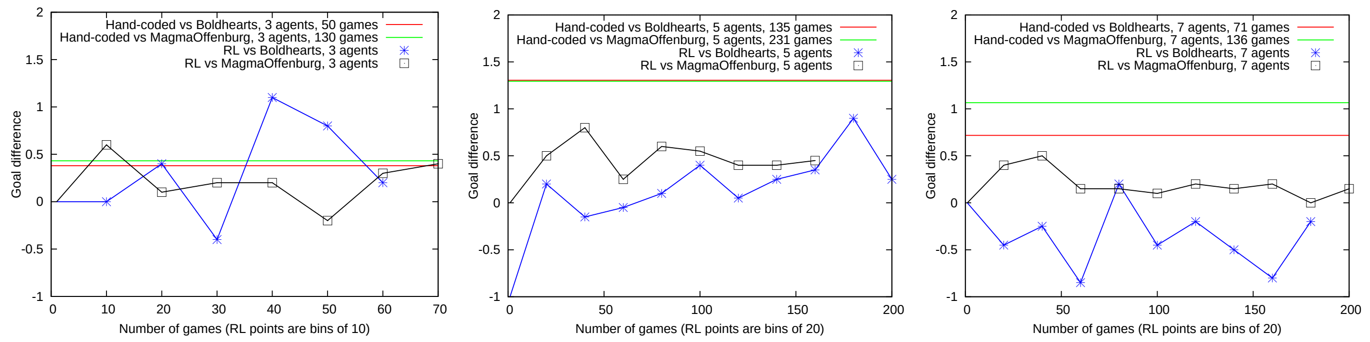
The agent closest to the ball becomes the striker, and only one agent is allowed to become the striker. The other agents except the goalie are allowed to choose from twelve roles. We map the available roles to discrete actions of the RL algorithm. In order to use Algorithm 1 (Fig. 1), an agent must formulate a question function using a value function, and the answer function provides the solution as an approximate value function. All the agents formulate the same question: *What is my role in this formation in order to maximize future rewards?* All agents learn independently according to the question, while collaboratively aiding each other to maximize their future reward. We make the assumption that the agents do not communicate their current role. Therefore, at a specific step, multiple agents may commit to the same role. We discourage this condition by modifying the question as *What is my role in this formation in order to maximize future rewards, while maintaining a completely different role from all teammates in all time steps?*

C. State Variables Representation

Figure 2b shows the schematic diagram of the state variable representation. All points and vectors in Figure 2b are defined with respect to a global coordinate system. h is the middle point of the home goal, while o is the middle point of the opponent goal. b is the ball position. $\|.\|$ represents the vector length, while $\angle pqr$ represents the angle among three points $p,\ q,$ and r pivoted at q. a_i represents the self-localized point of the $i = 1, \ldots, 11$ teammate agent. y_i is some point in the direction of the robot orientation of teammate agents. $c_j,\ j = 1, \ldots, 11$, represents the mid-point of the tracked opponent agent. x represents a point on a vector parallel to unit vector e_x. Using these labels, we define the state variables as:

$$\{\| \vec{v}_{hb} \|, \| \vec{v}_{bo} \|, \angle hbo, \{\| \vec{v}_{a_i b} \|, \angle y_i a_i b, \angle a_i bx\}_{i=n_{start}}^{n_{end}}, \\ \{\| \vec{v}_{c_j b} \|, \angle c_j bx, \}_{j=1}^{m_{max}}\}.$$

n_{start} is the teammate starting id and n_{end} the ending id. m_{max} is the number of opponents considered. Angles are normalized to $[-\frac{\pi}{2}, \frac{\pi}{2}]$.



(a) (b) (c)

Fig. 3: Goal difference in games with (a) three; (b) five; and (c) seven agents per team.

D. GVFs Question and Answer Functions

There are twelve actions available in each state. We have left out the striker role from the action set. The agent nearest to the ball becomes the striker. All agents communicate their belief to other agents. Based on their belief, all agents calculate a cost function and assign the closest agent as the striker. We have formulated a cost function based on relative distance to the ball, angle of the agent, number of teammates and opponents within a region near the ball, and whether the agents are active. In our formulation, there is a natural termination condition; scoring goals. With respect to the striker role assignment procedure, we define a pseudo-termination condition. When an agent becomes a striker, a pseudo-termination occurs, and the striker agent does not participate in the learning process unless it chooses another role. We define the question and answer functions as follows:

- *Question functions:* 1) π = greedy w.r.t $\hat{Q}$; 2) $\gamma(.) = 0.8$; 3) $r(.) =$ a) the change of x value of the absolute ball position; b) a small negative reward of 0.01 for each cycle; c) a negative reward of 5 is given to all agents within a radius of 1.5 meters; 4) $z(.) =$ a) $+100$ for scoring against opponent; b) -100 for opponent scoring; and 5) time step $= 2$ seconds.
- *Answer functions:* 1) $\pi_b = \epsilon$-greedy w.r.t target state-action function; 2) $\epsilon = 0.05$; 3) $I_t(.) = 1$; 4) $\phi(.,.) =$ a) we use tile coding to formulate the feature vector. $n_{start} = 2$ and $n_{end} = 3, 5, 7$. $m_{max} = 3, 5, 7$. Therefore, there are $18, 28, 30$ state variables. b) state variable is independently tiled with 16 tilings with approximately each with $\frac{1}{16}$ generalization. Therefore, there are $288 + 1,448 + 1,608 + 1$ active tiles (i.e., tiles with feature 1) hashed to a binary vector dimension $10^6 + 1$. The bias feature is always active; and 5) $\lambda(.) = 0.8$.
- Parameters:
1) $\| \theta \| = \| \mathbf{w} \| = 10^6 + 1$; 2) $\| \mathbf{e} \| = 2000$ (efficient trace implementation); 3) $\alpha_\theta = \frac{1}{289}, \frac{1}{449}, \frac{1}{609}$; and 4) $\alpha_w = 0.001 \times \alpha_\theta$.

V. Experiments

We conducted games against the teams Boldhearts and MagmaOffenburg, both semi-finalists of the RoboCup 3D Soccer Simulation competition in Mexico 2012[4]. We conducted knowledge learning according to the configuration given in subsection IV-D.

The first experiments were done using a team size of five with the RL agents against Boldhearts. After 140 games our RL agent increased the chance to win from 30% to 50%. This number does not increase more in the next games, but after 260 games the number of lost games (initially 35%) is reduced to 15%. In the further experiments we used the goal difference to compare the performance of the RL agent. Figure 3 shows the average goal differences that the hand-tuned role assignment and the RL agents archive in games against Boldhearts and MagmaOffenburg using different team sizes. With only three agents per team the RL agent only needs 40 games to learn a policy that outperforms the hand-coded role selection (Fig. 3a). Also with five agents per team, the learning agent is able to increase the goal difference against both opponents (Fig. 3b). However, it does not reach the performance of the manually tuned role selection. Nevertheless considering the amount of time spent for fine-tuning the hand-coded role selection, these results are promising. Furthermore, the outcome of the games depends a lot on the underlying skills of the agents, such as walking or dribbling. These skills are noisy, thus the results need to be averaged over many games (std. deviations in Fig. 3 have been between 0.5 and 1.3).

The results in Figure 3c show a bigger gap between RL and the hand-coded agent. However, using seven agents the goal difference is generally decreased, since the defense is easily improved by increasing the number of agents. Also the hand-coded role selection results in a smaller goal difference. Furthermore, considering seven agents in each team the state space is already increased significantly. Only 200 games seem to be not sufficient to learn a good policy. Sometimes the RL agents reach a positive goal difference, but it stays below the hand-coded role selection. In Section VI, we discuss some of the reasons for this inferior performances for the team size seven. Even though the RL agent did not perform well considering only the goal difference, it has learned a

[4]The published binary of the team UTAustinVilla showed unexpected behaviors in our tests and is therefore omitted.

moderately satisfactory policy. After 180 games the amount of games won is increased slightly from initially 10% to approximately 20%.

VI. Conclusions and Future Work

We designed and experimented RL agents that learned to assign roles in order to maximize expected future rewards. All the agents in the team ask the question "What is my role in this formation in order to maximize future rewards, while maintaining a completely different role from all teammates in all time steps?". This is a goal-oriented question. We use Greedy-GQ(λ) to learn experientially grounded knowledge encoded in GVFs. Dynamic role assignment function is abstracted from all other low-level components such as walking engine, obstacle avoidance, object tracking etc. If the role assignment function selects a passive role and assigns a target location, the lower-layers handle this request. If the lower-layers fail to comply to this request, for example being reactive, this feedback is not provided to the role assignment function. If this information needs to be included; it should become a part of the state representation, and the reward signal should be modified accordingly. The target positions for passive roles are created w.r.t the absolute ball location and the rules imposed by the 3D soccer simulation league. When the ball moves relatively fast, the target locations change faster. We have given positive rewards only for the forward ball movements. In order to reinforce more agents within an area close to the ball, we need to provide appropriate rewards. These are part of reward shaping [20]. Reward shaping should be handled carefully as the agents may learn sub-optimal policies not contributing to the overall goal.

The experimental evidences show that agents are learning competitive role assignment functions for defending and attacking. We have to emphasize that the behavior policy is ϵ-greedy with a relatively small exploration than uniformly distributed as used in [1]. The main reason for this decision is that when an adversary is present with the intention of maximizing it objectives, practically the learning agent may have to run for a long period to observe positive samples. Therefore, we have used the off-policy Greedy-GQ(λ) algorithms for learning goal-oriented GVFs within on-policy control setting. Our hypothesis is that with the improvements of the functionalities of lower-layers, the role assignment function would find better policies for the given question and answer functions. Our next step is to let RL agent learn policies against other RoboCup 3D soccer simulation league teams. Beside the role assignment, we also contributed with testing off-policy learning in high-dimensional state spaces in a competitive adversarial environment. We have conducted experiments with three, five, and seven agents per team. The full game consists of eleven agents, and the next step is to extend learning to consider all agents, and to include methods that select informative state variables and features.

Since, we learned from off-policy experiences, we can save the tuples, $(S_t, A_t, S_{t+1}, r(S_{t+1}), \rho_t, z(S_{t+1}))$, and learn the policy off-line. The Greedy-GQ(λ) learns a deterministic greedy policy. This may not be suitable for complex and dynamic environments such as the RoboCup 3D soccer simulation environment. Therefore, we need to resolve to actor-critic algorithms such as Off-PAC [21]. In the future, we plant to investigate these methods to obtain competitive policies.

References

[1] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *AAMAS*, 2011, pp. 761–768.

[2] H. R. Maei and R. S. Sutton, "GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces," *Proceedings of the 3d Conference on Artificial General Intelligence AGI10*, pp. 1–6, 2010.

[3] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton, "Toward off-policy learning control with function approximation," in *ICML*, 2010, pp. 719–726.

[4] R. S. Sutton, C. Szepesvári, and H. R. Maei, "A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation," in *NIPS*, 2008, pp. 1609–1616.

[5] P. Stone and M. Veloso, "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artificial Intelligence*, vol. 110, no. 2, pp. 241–273, June 1999.

[6] ——, "Layered learning," in *Proceedings of the Eleventh European Conference on Machine Learning*. Springer Verlag, 1999, pp. 369–381.

[7] P. Stone, R. S. Sutton, and G. Kuhlmann, "Reinforcement learning for RoboCup-soccer keepaway," *Adaptive Behavior*, vol. 13, no. 3, pp. 165–188, 2005.

[8] A. Seekircher, S. Abeyruwan, and U. Visser, "Accurate ball tracking with extended kalman filters as a prerequisite for a high-level behavior with reinforcement learning," in *The 6th Workshop on Humanoid Soccer Robots at Humanoid Conference, Bled (Slovenia)*, 2011.

[9] T. Gabel, S. Lange, M. Lauer, and M. Riedmiller, "Bridging the gap: Learning in the robocup simulation and midsize league," in *In Proceedings of the 7th Portuguese Conference on Automatic Control (Controlo)*, 2006.

[10] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Auton. Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[12] H. Köse, "Q-learning based market-driven multi-agent collaboration in robot soccer," *TAINN*, pp. 219–228, 2004.

[13] J. A. Gurzoni, F. Tonidandel, and R. A. C. Bianchi, "Market-based dynamic task allocation using heuristically accelerated reinforcement learning," in *EPIA*, 2011, pp. 365–376.

[14] P. MacAlpine, D. Urieli, S. Barrett, S. Kalyanakrishnan, F. Barrera, A. Lopez-Mobilia, N. Ştiurcă, V. Vu, and P. Stone, "UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition," in *Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, June 2012.

[15] T. Degris and J. Modayily, "Scaling-up knowledge for a cognizant robot," *In notes of the AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI*, 2012.

[16] J. Modayil, A. White, and R. S. Sutton, "Multi-timescale nexting in a reinforcement learning robot," *CoRR*, vol. abs/1112.1133, 2011.

[17] P. Pilarski, M. Dawson, T. Degris, J. Carey, and R. Sutton, "Dynamic switching and real-time machine learning for improved human control of assistive biomedical robots," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS EMBS International Conference on*, june 2012, pp. 296 –302.

[18] H. R. Maei, "Gradient temporal-difference learning algorithms," Ph.D. dissertation.

[19] J. Stoecker and U. Visser, in *RoboCup*, ser. Lecture Notes in Computer Science, T. Rfer, N. M. Mayer, J. Savage, and U. Saranli, Eds., vol. 7416. Springer, 2011, pp. 282–293.

[20] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ser. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287.

[21] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," *CoRR*, vol. abs/1205.4839, 2012.