In Lakemeyer, Sklar, Sorrenti, Takahashi, editors, *RoboCup-2006: Robot Soccer World Cup X*, pp. 72--85, Springer Verlag, 2007.

Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study

Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-0233
{shivaram, yxliu, pstone}@cs.utexas.edu

Abstract. We present half field offense, a novel subtask of RoboCup simulated soccer, and pose it as a problem for reinforcement learning. In this task, an offense team attempts to outplay a defense team in order to shoot goals. Half field offense extends keepaway [11], a simpler subtask of RoboCup soccer in which one team must try to keep possession of the ball within a small rectangular region, and away from the opposing team. Both keepaway and half field offense have to cope with the usual problems of RoboCup soccer, such as a continuous state space, noisy actions, and multiple agents, but the latter is a significantly harder multiagent reinforcement learning problem because of sparse rewards, a larger state space, a richer action set, and the sheer complexity of the policy to be learned. We demonstrate that the algorithm that has been successful for keepaway is inadequate to scale to the more complex half field offense task, and present a new algorithm to address the aforementioned problems in multiagent reinforcement learning. The main feature of our algorithm is the use of inter-agent communication, which allows for more frequent and reliable learning updates. We show empirical results verifying that our algorithm registers significantly higher performance and faster learning than the earlier approach. We also assess the contribution of inter-agent communication by considering several variations of the basic learning method. This work is a step further in the ongoing challenge to learn complete team behavior for the RoboCup simulated soccer task.

1 Introduction

RoboCup simulated soccer [2, 4] has emerged as an excellent domain for researchers to test ideas in machine learning. Learning in the RoboCup soccer domain has to overcome several challenges, such as a continuous multi-dimensional state space, noisy sensing and actions, multiple agents (including adversaries), and the need to act in real-time. Machine learning techniques have been used in the past on a wide range of tasks in RoboCup soccer. For instance, the Brainstormers team [8, 9] uses reinforcement learning to train both individual behaviors and team strategies. Several researchers have focused on specific subtasks like goal-shooting [3, 6].

Keepaway is a subtask of RoboCup soccer that has recently been proposed by Stone *et al.* [11] as a testbed for reinforcement learning methods. In keepaway, a team of *keepers* tries to keep possession of the ball away from the opposing team of *takers* within a small rectangular region. The task is episodic, and each episode ends when the takers gain possession, or when the ball goes outside the region of play. The keepers seek to maximize the duration of the episode, and are rewarded based on the time elapsed after every action. Stone *et al.* [11] provide a Sarsa-based reinforcement learning method to learn keeper behavior at a high level of abstraction.

In this paper, we extend keepaway to a more complex task of RoboCup soccer, **half field offense**. This task is played on one half of the soccer field, much bigger than the typical keepaway region. There are also typically more players on both teams. In each episode, the *offense* team needs to score, which involves keeping possession of the ball, moving up the field, and shooting goals. The *defense* team tries to stop it from doing so. Since the task realistically models the offense scenario in soccer, a policy learned for half field offense can be integrated quite naturally into full-fledged RoboCup simulated soccer games.

Both keepaway and half field offense have to cope with the usual difficulties associated with RoboCup soccer: continuous state space, noisy actions, and multiple agents. But several factors contribute to making half field offense a much harder multiagent reinforcement learning problem than keepaway. Maintaining possession of the ball is the main objective in keepaway, but it is only a subtask in half field offense. In order to succeed in half field offense, the offense players not only have to keep possession, but must also learn to pass or dribble to forge ahead towards the goal, and shoot whenever an angle opens up. With a larger state space and a richer action set than keepaway, a successful half field offense policy is therefore quite complex. A factor that makes learning in half field offense even more difficult is that the success of the task is evaluated simply based on whether a goal is scored or not at the end of an episode. Since goal scoring episodes are rare initially, it becomes necessary that the learning algorithm make the most efficient use of such information.

The learning method proposed for keepaway [11] only achieves limited success on the more difficult half field offense task. We analyze this method and propose a new method that overcomes many of its shortcomings. While reinforcement learning is indeed constructed to accommodate delayed updates and learning complex policies, we show that the learning process on a complex multiagent task like half field offense can be expedited by making better design choices. In particular, our algorithm uses inter-player communication to speed up learning and achieve better performance. We introduce the half field offense task in Section 2 and the learning method in Section 3. Section 4 presents emprical results of the performance of our method on the half field offense task. Section 5 discusses related work, and we conclude in Section 6.

2 Half Field Offense Task Description

Half field offense is an extension of the keepaway task [11] in RoboCup simulated soccer. In half field offense, an offense team of m players has to outsmart the defense team of n players, including a goalie, to score a goal. Typically $n \geq m$. The task is played over one half of the soccer field, and begins near the half field line with the ball close to one of the offense players. The offense team tries to maintain possession (keep the ball close enough for it to be kicked), move up the field, and score. The defense team tries to take the ball away from the offense team. The task is episodic, and an episode ends when one of three events occurs: 1. A goal is scored, 2. The ball is out of bounds, or 3. A defender gets possession of the ball (including the goalie catching the ball). Fig. 1 shows a screen-shot from a half field offense task, where four players are on the offense team and five players including a goalie are on the defense team. We denote this version of the task 4v5 half field offense, and discuss it in Section 2.2.

In principle, it is possible to frame half field offense as a learning problem for either the offense team or the defense team (or both), but here we only focus on learning by the offense team. The objective is to increase the goal-scoring performance of the offense team, while the defense team follows a fixed strategy. A similar approach is also adopted for keepaway [11].

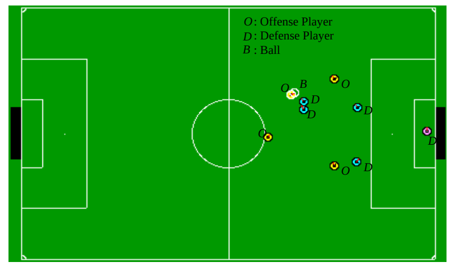


Fig. 1: Half field offense game in progress.

The offense team player who possesses the ball (and is hence closest to it) is required to take one of the following actions[1]:

- **Pass**k. This action involves a direct kick to the teammate that is the k-th closest to the ball, where $k = 2, 3, \ldots, m$. The representation used is indexical, since it is based on distances to the teammates, and not their actual jersey numbers.
- **Dribble**. In order to encourage the offense player with possession to dribble towards the goal, a cone is constructed with the player at its vertex and its axis passing through the center of the goal. The player takes a small kick within this cone in a direction that maximizes its distance to the closest defense player also inside the cone. The half angle of the cone is small (15°) when it is far away from the goal and opponents, but is progressively increased (up to 75°) as it gets closer to the goal or opponents. Thus it is encouraged to forge ahead towards the goal whenever possible, but has room to move away from the defense players should they get too close.
- **Shoot**. By taking this action the player kicks the ball towards the goal in the direction bisecting the widest open angle available between the goalie, other defenders, and the goalposts.

When no offense player has possession of the ball, the one closest to the ball seeks to reach it by dashing directly towards it (**GetBall**). Offense players other than the one closest to the ball always try to to maintain a formation in order to take the attack forward (**GetOpen**). More precisely, any player from the offense team is constrained to behave as follows.

if I have possession of the ball **then**
  Execute some action from $\{\mathbf{Pass}2, \ldots, \mathbf{Pass}m, \mathbf{Dribble}, \mathbf{Shoot}\}$
else if I am the closest offense player to the ball **then**
  GetBall (Intercept the ball).
else
  GetOpen (Move to the position prescribed by the formation, see Section 2.2).

[1] They are in fact high-level skills, and are better described by the term "options," which are themselves composed of low-level actions over extended time periods. They nevertheless play the role of actions in the sense of reinforcement learning. We simply refer to them as actions for simplicity. For a more detailed discussion, see [11].

Therefore, the behavior of any offense player is fixed except when it has possession of the ball. Deciding which action to take when in possession of the ball precisely constitutes the learning problem, as is also the case in keepaway [11]. In principle, the player who has possession can benefit from a larger action set than the one we have described, but these actions are enough to achieve quite a high level of performance. More importantly, they allow us to focus on learning high-level strategies.

2.1 State Representation

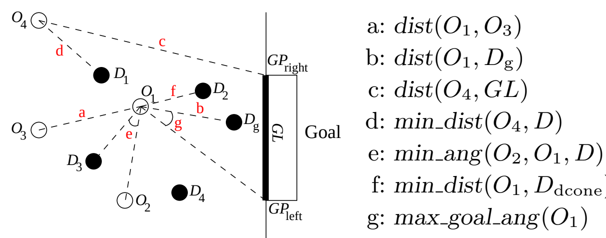


Fig. 2: Sample state variables for 4v5 half field offense.

In RoboCup simulated soccer [2], the server provides the players sensory information at regular intervals of time. Players typically process the low-level information thus obtained to maintain estimates of the positions and velocities of the players and the ball. For our task, we define the state using a set of variables involving distances and angles between players, which can be derived from information about their positions. These are listed below. The offense players are numbered according to their distances to the ball using an indexical representation. The offense player with the ball is always denoted O_1. Its teammates are $O_2, O_3, \ldots, O_m$. The defense players are also numbered according their distances to the ball; they are $D_1, D_2, \ldots, D_n$. The goalie, which may be any of the D_i, is additionally denoted D_g.

- $dist(O_1, O_i), i = 2, 3, \ldots, m$. The distances from O_1 to its teammates.
- $dist(O_1, D_{\text{g}})$. The distance from O_1 to the goalie on the defense team.
- $dist(O_i, GL), i = 1, 2, \ldots, m$. For the offense player, the distance to the segment of the goal line GL between the goalposts.
- $min\_dist(O_i, D), i = 1, 2, \ldots, m$. For each offense player, the closest distance to any opponent, that is, $min\_dist(O_i, D) = \min_{j=1,\ldots,n} dist(O_i, D_j)$.
- $min\_ang(O_i, O_1, D), i = 2, 3, \ldots, m$. For offense players other than O_1, the smallest angle $\angle O_i O_1 D$ among all D, where D is a defense player, that is, $min\_ang(O_i, O_1, D) = \min_{j=1,\ldots,n} ang(O_i, O_1, D_j)$.
- $min\_dist(O_1, D_{\text{dcone}})$. The distance from O_1 to the closest defender in the dribble cone or dcone. The dribble cone is a cone with half angle 60° with its vertex at O_1 and axis passing through the center of the goal. D_{dcone} is the set of defenders in the dribble cone.
- $max\_goal\_ang(O_1)$. The maximum angle with the vertex at O_1, formed by rays connecting O_1 and goalposts or defense players that are in the goal cone, which is the triangle formed by O_1 and the two goalposts GP_{left} and GP_{right}.

We adopt this set of state variables expecting them to be of direct relevance to the actions, although they are neither independent nor complete. We expect $dist(O_1, GL)$, $max\_goal\_ang(O_1)$, and $dist(O_1, D_{\text{g}})$ to directly affect **Shoot**; $min\_dist(O_1, D)$ and $min\_dist(O_1, D_{\text{dcone}})$ to affect **Dribble**; and the other variables to affect **Pass**k. As in keepaway [11], the indexical representation based on distances is expected to help the players generalize better. We arrived at this set of state variables through experimentation, but did not expend much time optimizing the set. We note that the set of state variables is independent of the number of defense players, and has a linear relation with the number of offense players, therefore scaling to versions of the task with large numbers of players. The 4v5 version of the task uses 17 state variables.

2.2 4v5 Half Field Offense and Benchmark Policies

4v5 half field offense (see Fig. 1) is a version of the task involving four offense players and five defense players, including the goalie. We use this version of the task for all our experiments. In 4v5, the offense player wiith the ball must choose an action from the set $\{\mathbf{Pass}2, \mathbf{Pass}3, \mathbf{Pass}4, \mathbf{Dribble}, \mathbf{Shoot}\}$, while the other offense players, following a fixed strategy, stay in an arc formation. The defense players also follow a static policy. Due to space limitations, the complete behaviors of the offense and defense players on the 4v5 task are specified on a supplementary web site[2]. The web site also lists examples of policies (including **Random**, in which actions are chosen randomly, and **Handcoded**, a policy we have manually engineered) for the 4v5 task and videos of their execution.

3 Reinforcement Learning Algorithm

Since half field offense is modeled as an extension of keepaway [11], they share the same basic learning framework. In fact, the learning method that has been most successful on keepaway [11] can be directly used for learning half field offense. But while with keepaway the learning curve obtained using this method typically levels off after just 15 hours of simulated time, we find that with half field offense it continues to gradually rise even after 50 hours. Furthermore, when we visualize the execution of the policy thus learned,[2] it is quite apparent that it is sub-optimal. In order to ascertain whether indeed the learning can be improved, we analyze the task and the learning method in detail. We then proceed to introduce a new learning approach using inter-agent communication, which significantly improves the learning rate and the resulting performance. In this section, we explain how the reinforcement learning method is set up, the problems faced by multiagent reinforcement learning on this task, and how we handle them using explicit inter-agent communication.

3.1 Basic Setup

As in keepaway [11], the reinforcement learning problem is modeled as a semi-Markov decision process [1], where decisions are taken at unequal intervals of time, and only by the player with the ball. Each agent uses a function approximator to represent an action-value function or Q-function that maps state and action pairs (s, a) to real numbers, which are its current estimates of the expected long term reward of taking action a from state s. Each agent updates its Q-function using

Game Scenario	Notation	Reward
Goal	*goal_reward*	1.0
Ball with some offense player	*offense_reward*	0
Ball caught by goalie	*catch_reward*	-0.1
Ball out of bounds	*out_reward*	-0.1
Ball with some defense player	*defense_reward*	-0.2

Table 1: Definition of rewards

[2] `http://www.cs.utexas.edu/~AustinVilla/sim/halffieldoffense/index.html`

the Sarsa learning method. The main difference between the method from [11] and the one we propose in Section 3.3 is in how each agent obtains the experience, and how frequently updates are made.

Rewards for the reinforcement learning problem are defined in Table 1. A positive reward of 1.0 is given for an action that results in a goal, while small negative rewards are given when the episode ends unsuccessfully. It is conceivable to give small positive rewards (say 0.01) for successful passes, but we found that a zero reward was just as effective. Negative rewards are provided at the end of unsuccessful episodes to encourage keeping the ball in play. The ratios between different rewards can have a significant impact on the learning process [3]. We informally tested out different values and found this particular assignment effective. Since the task is episodic, we do not use discounting.

3.2 Difficulty of Multiagent Learning

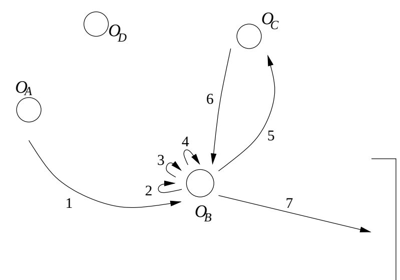


Fig. 3: Example episode: The numbers indicate the sequence of actions.

In the method used for keepaway [11], each agent learns independently. The only points in time when it receives rewards and makes updates to its Q-function are when it has possession of the ball, or when the episode ends. The reward itself is the length of the duration between when the action was taken and when possession was regained or the episode ended. Clearly, it is easy to apply this scheme to half field offense, using the reward structure specified in Table 1. We illustrate this method by tracing through a typical episode from half field offense (depicted in Fig. 3). The episode begins with O_A in possession of the ball.[3] O_A passes the ball to O_B, who takes three dribble actions before passing it to O_C. O_C then passes it back to O_B, who shoots the ball into the goal. O_D does not participate in this episode. Using the learning method from [11], O_A will receive *goal_reward* for its pass action (1); O_B will receive *offense_reward* for its dribble actions (2, 3, and 4) and pass action (5), and *goal_reward* for its shoot action (7); while O_C will get *goal_reward* for its pass action (6). O_A and and O_C will only make their learning updates at the end of the episode, while O_B will also make intermediate updates whenever it regains possession. O_D does not make any learning update in this episode.

We find shortcomings in the method from [11] that we have just described. First, consider another episode that differs from the above example only in that the final **Shoot** action results in the ball being caught by the goalie instead of finding the goal. In this case also, O_A, O_B, and O_C make corresponding updates to their Q-functions, but the reward used for the updates is *catch_reward*. Even though the reason for the failure to score on this episode is only perhaps a slightly flawed **Shoot** action, the reward assigned to O_A for its successful pass to O_B (and indeed actions 6 and 7) now becomes drastically different (negative instead of positive). This case illustrates that it is more desirable for O_A to update its Q-function for the pass to O_B right after O_B receives the ball, since the update then would be based on the Q-value of O_B in its current state, and the reward for a successful pass. While

[3] We use subscripts A, B, C, D to indicate fixed players since numerical subscripts indicate players according to how close they are to the ball.

using the method from [11], the update is only based on how the episode ends. This can lead to a higher variance for updates to the Q-function, especially since the task is stochastic. The problem is that there is a long temporal delay between the execution of an action by a player and the corresponding learning update; because of this delay, the assigned reward and the next resulting state can change drastically. It is not too harmful in keepaway, since the rewards are themselves the time elapsed between the events, and do not change based on the next state. But in half field offense, even slight changes in the middle of an episode can lead to very different outcomes and very different rewards.

Another evident shortcoming is the case of player O_D. Since each player learns solely based on its own experiences in the method from [11], O_D, which is not involved in this episode, does not update its Q-function even once during this episode. Since the players are homogeneous, it seems conceivable that the players' experiences can be shared. For instance, O_D should be able to learn, based on O_B's experience, that **Shoot** action taken from close to the goal is likely to receive high reward. In fact, even among O_A, O_B, and O_C, only O_B records the information that the **Shoot** action resulted in a goal; O_A and O_C are only able to make updates to their respective **Pass** actions. Surely, they will also benefit by making O_B's update for the **Shoot** action. Sharing experiences can be particularly useful early in training, when successful episodes are rare. We next describe our learning method, which uses inter-agent communication to overcome the shortcomings described above.

3.3 Agent Communication

In the solution we propose, inter-agent communication is used to facilitate information sharing among the agents, and to enable frequent and more reliable updates. The protocol followed is similar to one used by Tan [12] for learning in an artificial predator-prey domain, where agents are able to communicate their experiences to their partners.

Number	Sender	State	Action	Reward
1	O_A	s_1	**Pass**3 (to O_B)	*offense_reward*
2	O_B	s_2	**Dribble**	*offense_reward*
3	O_B	s_3	**Dribble**	*offense_reward*
4	O_B	s_4	**Dribble**	*offense_reward*
5	O_B	s_5	**Pass**2 (to O_C)	*offense_reward*
6	O_C	s_6	**Pass**2 (to O_B)	*offense_reward*
7	O_B	s_7	**Shoot**	*goal_reward*

Table 2: Messages broadcast during the example episode

Since every action leads either to some offense player getting possession or the end of the episode, in our scheme, the appropriate reward for that action is provided as soon as one of these events occurs. Thus, in our example, O_A is given *offense_reward* as soon as its pass (action 1) to O_B succeeds, instead of having to wait until the end of the episode to receive *goal_reward*. As soon as it receives *offense_reward*, O_A broadcasts a message (see Table 2) to its teammates, describing the state in which it was when it took the pass action (s_1), the pass action itself (**Pass**3), and the reward received (*offense_reward*). In general, every time a player takes action a in state s and receives reward r, it broadcasts a message of the form (s, a, r) to the team. Since a Sarsa update is completely specified by a (s, a, r, s', a') tuple, each player records the messages received and makes an update as soon as enough information is available for it. Thus, when O_B broad-

Algorithm 1 Reinforcement Learning with Communication

```
Initializations;
for all episode do
  s ← NULL;
  repeat
    // acting
    if I have possession of the ball then
      s ← getCurrentStateFromEnvironment();
      Choose action a using Q-function and ϵ-greedy selection;
      Execute action a;
      r ← waitForRewardFromEnvironment();
      broadcast(s, a, r);
    else
      if I am the closest offense player to the ball then
        GetBall;
      else
        GetOpen;
    // learning
    if I receive message (s_m, a_m, r_m) then
      if s is NULL then
        s, a, r ← s_m, a_m, r_m;
      else
        s′, a′, r′ ← s_m, a_m, r_m;
        Perform Sarsa update based on (s, a, r, s′, a′):
            Q(s, a) ← Q(s, a) + α (r + γQ(s′, a′) − Q(s, a));
        s, a, r ← s′, a′, r′;
  until episode ends;
```

casts message 2 (see Table 2), *all* the players make a Sarsa update using the tuple $(s_1, \textbf{Pass}3, \textit{offense\_reward}, s_2, \textbf{Dribble}.)$. It is quite clear that in this scheme, the SMDP step is designed to last only until some teammate gets possession (unless the episode ends before that), therefore keeping the updates more reliable. At the same time, communication permits each player to make an update for every action that has been taken by any of the offense players during the episode. In fact, since all the players begin with the same initial Q-function and make the same updates, we can expect that their action-value functions will always be alike, thereby reducing an essentially distributed problem to one of centralized control (imagine a single "virtual" agent who resides at all times inside the player who currently has possession of the ball). However, in practice, the message passing is not completely reliable, so a small number of updates get missed.

Algorithm 1 provides the pseudocode of the algorithm the learning players implement. Each player stores its current action value function using a function approximator. When in possession of the ball, it decides which action to take based on an ϵ-greedy action selection scheme. After executing a and receiving a reward r, the player broadcasts the triple (s, a, r) to the team. Players not in possession of the ball simply follow the static policy. Each player uses the first message that is received during an episode to initialize values for the triple (s, a, r), and on every subsequent message (s', a', r') makes a learning update using the saved and received information.

We implement the inter-player communication using a "trainer," an independent agent that can communicate with all the players. The player broadcasting an (s, a, r) message actually sends it to the trainer, who then sends it to all the players. To be consistent, we assume that even to make an update corresponding to its own action, a player first sends a message to the trainer, and makes the update only on receiving it back from the trainer. The trainer sends a special (s, a, r) message to the players when the end of an episode is reached, so that they may make a final update for that episode and start afresh for the next. We use Sarsa(0) as our reinforcement learning algorithm, along with CMAC tile coding for function approximation (as in [11]). The CMACs comprise 32 tilings for each feature. Distance features have tile widths of $3m$, while the tile width for angle features is fixed at $10°$. We use $\alpha = 0.125$, $\gamma = 1.0$, and $\epsilon = 0.01$.

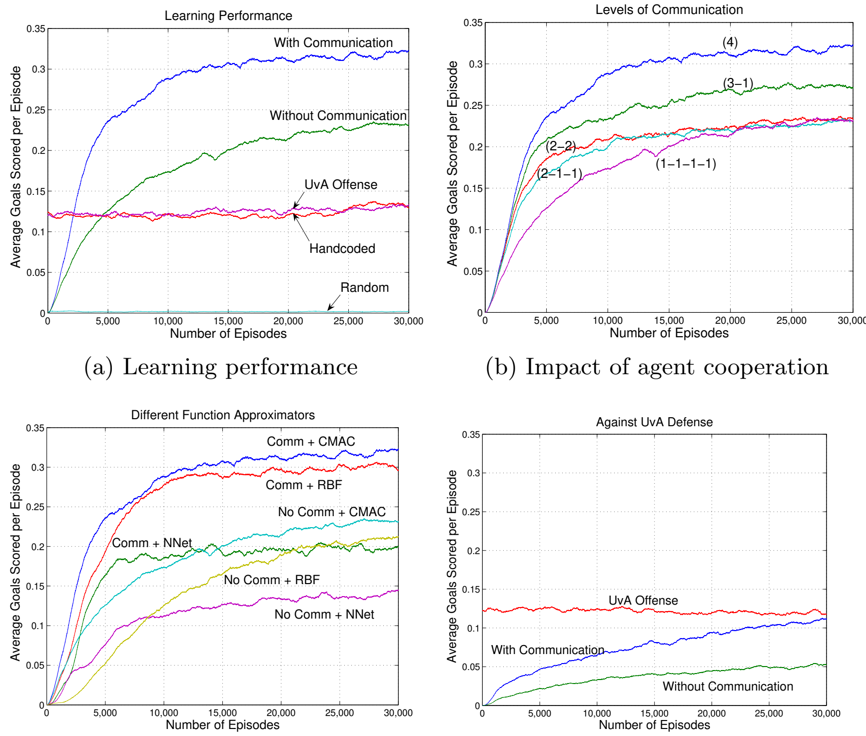
4 Experimental Results and Discussion

In this section we present performance results of our learning algorithm. The graphs depict learning curves with the y-axis showing the fraction of successful episodes, and the x-axis the number of training episodes. The learning curves are smoothed using a sliding window of 1000 episodes. Each curve is an average of at least 30 independent runs. We have performed t-tests every $5,000$ episodes comparing the values of the curves, and we report the levels of significance for important comparisons.

To focus on learning while still perserving a high level of complexity in our experiments with the 4v5 half field offense task, we have modified a couple of RoboCup simulated soccer defaults. While the RoboCup default only allows players to "see" within a $90°$ cone, we allow for $360°$ vision, which removes hidden state, but still retains sensor noise. Also, we do not enforce the offsides rule in our task, even though our players get offsides only occasionally. These changes are enforced in all our experiments, in order to make meaningful comparisons between different offense team policies.

Fig. 4(a) plots the performance of our learning algorithm. Our learning algorithm using inter-agent communication achieves a long term success ratio of 32%, while one where the agents learn independently (as in keepaway [11]) only manages to register 23%. Beyond 5000 episodes, their order is significant ($p < 10^{-8}$). Clearly, the gain from using communication is substantial. This is particularly apparent when we compare it to the performance recorded by other static policies. Within 2000 episodes of training, our algorithm is able to learn a more successful policy than the **Handcoded** policy mentioned in Section 2.2. When we visualize the execution of the learned policy,[2] it is noticeably different from the **Handcoded**, suggesting that learning is able to capture behavior that is non-intuitive for humans to describe. Fig. 4(a) also plots the performance of the **Random** policy, which succeeds less than 1% of the time, thereby confirming that extended sequences of the right actions are required to score goals. The other curve in the graph shows the performance achieved by a set of four offense players (numbers 6, 7, 8, and 9) from the UvA 2003 RoboCup team [5], which won the RoboCup simulated league championship that year. The comparison between our players and the UvA offenders is not completely fair, because they have not been tuned specifically for the half field offense task. But the fact that our players are able to learn a policy that performs at least twice as well as the UvA players in this setting still gives some insight into the effectiveness of the policy they learn.

In order to get a clearer understanding of the impact of communication, we ran a set of experiments in which only subsets of players communicate among themselves, while some learn independently (as in keepaway [11]). Fig. 4(b) plots learning curves from experiments in which all four players communicate their updates (4), only three of them communicate while one learns independently (3-1), all players communicate but each only with a partner (2-2), two of them communicate while



(a) Learning performance (b) Impact of agent cooperation

(c) Different function approximators (d) Against UvA defenders

Fig. 4: Experimental results

two learn independently (2-1-1), and they are all independent (1-1-1-1). At $5,000$ and $10,000$ episodes of training, the order specified above (except between 2-2 and 2-1-1) is significant with $p < 10^{-3}$, suggesting that increased communication results in a faster learning rate. After $30,000$ episodes of training, the full-communication curve (4) remains ahead of all the others, with $p < 10^{-8}$. It is clear, therefore, that communication does make a significant difference to the performance. Through informal experimentation, we verified that communication results in a faster learning rate for the keepaway task too, though the final policy it learns does not perform significantly better than one learned with no communication [11].

While we use CMACs for function approximation in most of our experiments, we ran an additional set of experiments using different function approximators to see how this change affects the performance. The other function approximators we used are neural networks (NNets) and radial basis functions (RBFs). They have also been in the past used for learning keepaway [10]. Fig. 4(c) plots the performance obtained by using these function approximators both in the full-communication case and the no-communication case. In both cases, the order CMAC > RBF > NNet is preserved beyond $10,000$ episodes of training (In keepaway the RBFs perform slightly better than the CMACs). But more importantly, we find that for each of the function approximators, significantly higher performance is achieved by the communication-based algorithm.

In order to verify to what extent our learning algorithm was robust to changes in the task, we ran a variation in which the offense team plays against a set of defense players from the UvA 2003 team [5] (players 5, 6, 2, 3, and 1). These players offer a far more defensive strategy than ours, and position themselves strategically to block passes between the offense players. After $30,000$ episodes of learning, our players show an 11% success rate, which almost matches the performance achieved by the set of UvA offense players against this opposition. In fact, the learning curve still seems to be rising at this point. The UvA offense performs better than the learned policy, however with a low confidence ($p < 0.2356$). Therefore our learning method is able to achieve a high level of performance against a world-class opposition, despite having only been trained with a limited action set. Surely, the main reason for its success is inter-agent communication, as the no-communication algorithm only manages to achieve a success rate of 5% in the same number of episodes.

5 Related Work

Half field offense is a natural extension of the keepaway task introduced by Stone *et al.* [11]. It is a much harder problem to learn than keepaway, and we have shown that inter-agent communication can be effectively coupled with the algorithm that has so far been successful for keepway [11] to boost its performance significantly. Geipel [3] and Maclin *et al.* [6] have in the past applied reinforcement learning techniques to goal-shooting scenarios, but these have typically involved fewer players and a smaller field than 4v5 half field offense. The Brainstormers RoboCup team [8, 9] has consistently applied reinforcement learning techniques to train different aspects of team behavior. They use reinforcement learning to learn high-level skills called "moves" in terms of low-level actions, and use these as primitives for learning high-level tactical behavior for attacking players [9]. For function approximation, they use neural networks, inputs to which are low-level state information. They also formulate simulated soccer as a Multi-agent Markov Decision Process (MMDP) [8] and discuss different models of agents based on their action sets and coordination. While their focus has been to develop a general architecture for learning team behavior, we, in this paper, address the specific problem of learning high-level behavior by the offense player with ball possession. For this reason, we use predefined high-level skills like **Pass**k and **Shoot**. Since we mainly use CMACs, which cannot represent arbitrary non-linear functions, we design our state features to be at a high level of abstraction, in order to facilitate better generalization.

Multiagent reinforcement learning with inter-agent communication has been studied in the past. Whitehead [13] describes a *Learn-by-Watching* method similar to ours and obtains theoretical bounds for the speedup in learning by multiple Q-learning agents. Tan [12] empirically evaluates the effect of inter-agent communication on a much simpler problem than ours, a predator-prey scenario within a 10x10 discrete grid. The predators can cooperate by sharing their sensations, learning episodes, and whole policies. Of these, sharing episodes involves communicating extended series of state-action-reward messages between the predators, and has a direct correspondence with the method we have employed. Again, communication is is shown to be beneficial to learning. Mataric [7] uses reinforcement learning to train real robots on a box-pushing task. Communication is mainly used to share their sensations in order to form a complete state of the world, unlike in our algorithm, where communication directly impacts updates made to the agents' action-value functions.

6 Conclusions and Future Work

In this paper, we have introduced half field offense, a novel subtask of RoboCup simulated soccer. It extends an earlier benchmark problem for reinforcement learning, keepaway [11]. Half field offense presents significant challenges as a multiagent reinforcement learning problem. We have analyzed the learning algorithm that has been most successful for keepaway [11], and scaled it to meet the demands of our more complex task. The main feature of our new algorithm is the use of inter-agent communication, which allows for more frequent and reliable learning updates. We have presented empirical results suggesting that the use of inter-agent communication can increase the learning rate and the resulting performance significantly.

Learning half field offense is a step further in the ongoing challenge to learn complete team behavior for RoboCup simulated soccer. In this work we have only focused on learning the behavior of the offense player who has possession of the ball. It is in principle possible to pose as learning problems the behaviors of the other offense players, as well as the defense team. Also, high-level skills like **Pass***k* and **Shoot**, which we have directly used for learning here, may themselves be learned in terms of low-level actions like *turn* and *kick*. These are avenues for future research.

Acknowledgements

This research was supported in part by NSF CISE Research Infrastructure Grant EIA-0303609, NSF CAREER award IIS-0237699, and DARPA grant HR0011-04-1-0035.

References

1. S. J. Bradtke and M. O. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In *Advances in Neural Information Processing Systems 7 (NIPS-94)*, 1995.
2. M. Chen, E. Foroughi, F. Heintz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, and X. Yin. *Users Manual: RoboCup Soccer Server — for Soccer Server Version 7.07 and Later.* The RoboCup Federation, August 2002.
3. M. M. Geipel. Informed and advice-taking reinforcement learning for simulated robot soccer. Master's thesis, Fakultät für Informatik, Forschungs- und Lehreinheit Informatik IX, Technische Universität München, 2005.
4. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, 1997.
5. J. R. Kok, N. Vlassis, and F. C. A. Groen. UvA Trilearn 2003 team description. In *Proceedings CD RoboCup 2003*, 2003.
6. R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.
7. M. J. Matarić. Using communication to reduce locality in distributed multi-agent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):357–369, 1998.
8. A. Merke and M. Riedmiller. Karlsruhe Brainstormers — a reinforcement learning approach to robotic soccer II. In *RoboCup-2001: Robot Soccer World Cup V*, 2001.
9. M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, and R. Ehrmann. Karlsruhe Brainstormers — a reinforcement learning approach to robotic soccer. In *RoboCup-00: Robot Soccer World Cup IV*, 2000.
10. P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu. Keepaway soccer: From machine learning testbed to benchmark. In *RoboCup-2005: Robot Soccer World Cup IX*, 2006.
11. P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.
12. M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning (ICML-93)*, pages 330–337, 1993.
13. S. D. Whitehead. A complexity analysis of cooperative mechanisms in reinforcement learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 607–613, 1991.