Chapter 7

Diversity in Robot Soccer

This chapter describes research investigating specialization in learning robot soccer teams. Each agent is provided a common set of skills (motor schema-based behavioral assemblages) from which it builds a task-achieving strategy using reinforcement learning. The agents learn individually to activate particular behavioral assemblages given their current situation and a reward signal. This work was conducted following the methodology introduced in Chapter 3.

The experiments in JavaBots robot soccer simulations evaluate the agents in terms of *performance*, *policy convergence*, and *behavioral diversity*. As in foraging experiments (Chapter 6) the results show that in some cases robots will diversify by choosing heterogeneous behaviors. An interesting contrast with foraging results however is that diverse soccer teams perform better than homogeneous teams. The degree of diversification and the performance of the team depend on the reward structure. When the entire team is jointly rewarded or penalized (global reinforcement), teams tend towards heterogeneous behavior. When agents are provided feedback individually (local reinforcement), they converge to identical policies.

7.1 Task and performance metric

Robot soccer is an increasingly popular focus of robotics research [KAK+97]. It it is an attractive domain for multiagent investigations because a robot team's success against a strong opponent often requires some form of cooperation. Also, it is familiar to many audiences and it provides opportunities for diversity among the team members.

Figure 7.1: Simulated and real robot soccer: a match at the 1998 Robot World Cup in Paris (left) and a JavaBots simulation (right). Photograph courtesy Hiroaki Kitano.

The task is patterned after the official RoboCup rules for small-size league play [Com97]. Each team is composed of five robot players. Once play begins the teams attempt to push and/or kick the ball (an orange golf ball) into the opponent's goal. The game is played on a green field the size of a table tennis table. Boundaries are 10cm tall walls – the golf ball bounces back instead of going out-of-bounds. Goals are 50cm wide. When a goal is scored the ball is reset to the middle of the field and the players are re-positioned. Official RoboCup matches include two 10 minute halves. A photograph of a RoboCup soccer game is presented in Figure 7.1 (left).

In the Java-based soccer simulation used in this research (Figure 7.1, right) a robot's control system interacts with a well-defined sensor-actuator interface. The simulation proceeds in discrete steps. In each step the robots process their sensor data, then issue appropriate actuator commands. The simulation models physical interactions (robot, ball and wall collisions), sensors and motor-driven actuators.

When the ball is bumped by a robot it immediately accelerates and rolls away. The direction the ball rolls after being bumped varies randomly from -10 to +10 degrees off center. Rolling friction is modeled with constant deceleration after the bump. Dynamics are based on actual RoboCup robot performance [Sto98]. Each agent is provided the following sensors:

- **velocity sensor:** provides present heading and speed of the robot,
- **bump sensor:** returns a force vector in the direction of any bump,
- **ball position sensor:** provides an egocentric vector to the soccer ball,
- **defended goal sensor:** provides an egocentric vector back to the robot's own goal,
- **opponent goal sensor:** provides an egocentric vector the opponent's goal,
- **team sensor:** returns an array of egocentric vectors pointing to the robot's team members,
- **opponent sensor:** an array of egocentric vectors pointing to the robot's opponents,
- **score sensor:** indicates whether the team has just scored or was scored against,
- **robot ID:** a unique integer from 1 to the size of the team.

Robots are able to sense all information germane to the task. This approximates the sensor system available to many of the real robot teams competing at RoboCup; information is gathered by a video camera mounted above the playing field. Future revisions of the simulator may address challenges faced by autonomous robots without accurate global sensors, e.g. sensor noise, occlusion and field-of-view constraints.

The following actuator interface is provided to the control system:

- **set drive speed:** a real value from -1 to 1 is sent to the robot's drive motor, indicating how fast the robot should go.
- **set heading:** a real value from 0 to 2π is sent to the robot's steering actuator indicating the desired heading for the robot.
- **kick:** if the ball is near the robot's kick actuator it is immediately accelerated in the direction of the robot's heading.

Now consider the performance metric for soccer. How can we objectively evaluate a robot soccer team? In a human game the object is to have scored the most points when time runs out. It is only necessary to score one more point than the other team. Here, we take the stance that greater score differentials indicate better performance. Hence, the performance metric for robot teams is

$$P = S_{\text{us}} - S_{\text{them}} \tag{7.1}$$

where S_{us} and S_{them} are the scores of each team at the end of the game.

In terms of the taxonomy introduced in Chapter 4 this task and performance metric have the following characteristics:

- `TIME_LIM` because performance is measured over a fixed period (except for simplified soccer),
- `OBJECT_BASED` since performance is based on the location of an object, the ball,
- `COMP_EXT` because robots on the team compete for positive performance (goals) against an (external) opposing team,
- `COMP_INT` because robots on the team compete for goals amongst themselves,
- `MULTI_AGENT` since a single agent is unlikely to net a positive score differential against a multiagent opponent,
- `SENSOR_COMPLETE` since agents can sense all aspects of the environment germane to the task perfectly.

The first set of experiments in the investigation were conducted in slightly simplified soccer domain. The domain is simplified as follows: Teams are composed of four players instead of five. The goal spans the width of the field's boundary instead of a 50cm wide zone. Play is continuous; after a scoring event, the ball is immediately replaced to the center of the field without repositioning the agents. Another important difference in the simplified task is that there is no time limit imposed; play continues until a total of 10 points are scored (the simplified is not `TIME_LIM`). To distinguish between the two tasks the simplified version is referred to as *simplified soccer*, while the more complex task is *RoboCup soccer*.

7.2 Behavioral design

Behavior-based approaches are well suited for robot soccer since they excel in dynamic and uncertain environments. The robot behaviors described here are implemented in Clay (Chapter 3), an object-oriented recursive system for configuring robot behavior. Clay integrates primitive behaviors (motor schemas) using cooperative and competitive coordination operators. Both static and learning operators are available.

Experiments in soccer are conducted by engaging an *experimental* team against a fixed opponent *control* team in soccer contests. We begin by describing the control team's behavioral configuration.

Since the experimental team's performance is significantly impacted by the skill of its opponent, it is important to avoid variability in the control team's strategy to ensure consistent results. The control team will always follow a fixed policy against the teams under evaluation. The control team's design is based on the following observations. First, points are scored by bumping the ball across the opponent's goal. Second, robots must avoid bumping the ball in the wrong direction, lest they score against their own team. A reasonable approach then, is for the robot to first ensure it is behind the ball, then move towards it to bump it towards the opponent's goal. Alternately, a defensive robot may opt to remain in the backfield to block an opponent's scoring attempt.

Table 7.1: The control team's policy summarized as look-up tables. The 1 in each row indicates the behavioral assemblage selected by the robot for the perceived situation indicated on the left. The abbreviations for the assemblages are introduced in the text.

Control Team Forward

perceptual feature	**assemblage** *mtb*	*gbb*	*mtb f*
not behind_ball	0	1	0
behind_ball	1	0	0

Control Team Goalie

perceptual feature	**assemblage** *mtb*	*gbb*	*mtb f*
not behind_ball	0	1	0
behind_ball	0	0	1

Each robot selects from a set of behavioral assemblages to complete the task. The behaviors are sequenced to form a complete strategy. This style of behavior-based robot design, referred to as *temporal sequencing*, views an agent's strategy as a Finite State Automaton. Temporal sequencing is discussed in Chapter 3. The strategies may be equivalently viewed as lookup tables (Table 7.1). Here we focus on the lookup table view since it is also useful for discussing learned policies. The behavioral assemblages developed for these experiments, and the motor schemas activated are:

- *move_to_ball (mtb)*: The robot moves directly to the ball. A collision with the ball will propel it away from the robot. Individual motor schemas active in this assemblage include:

  - **move_to_kickspot**: high gain to draw the robot to a point one-half of a robot radius behind the ball. If the robot bumps the ball from that location, the ball is propelled in the direction of the opponent's goal.
  - **avoid_teammates**: gain sufficiently high to keep the robots on the team spread apart This schema was not activated in the simplified soccer experiments, but was found to be useful in later work.

- *get_behind_ball (gbb)*: The robot moves to a position between the ball and the defended goal while dodging the ball to avoid bouncing it in the wrong direction. Activated motor schemas are
  - **move_to_halfway_point**: high gain to draw the robot to a point halfway between the ball and the defended goal.
  - **swirl_ball**: a ball dodging vector with gain sufficiently high to keep the robots from colliding with the ball.
  - **avoid_teammates**: gain sufficiently high to keep the robots from colliding.

- *move_to_back_field (mtbf)*: The robot moves to the back third of the field while being simultaneously attracted to the ball. The robot will kick/bump the ball if it is comes within range. Active schemas include
  - **move_to_defended_goal**: high gain to draw the robot to the defended goal. A "dead zone" centered on the goal area permits the robot to roam freely if it is near the goal.
  - **move_to_kickspot**: gain sufficiently high to draw the robot to the ball if it is near the goal, but not high enough to pull the robot away from the goal.

The overall system is completed by sequencing the assemblages with a selector that activates an appropriate skill depending on the robot's situation. This is accomplished by combining a boolean perceptual feature, *behind_ball* with a selection operator. The selector picks one of the three assemblages for activation, depending on the current value of *behind_ball*.

The control team includes three "forwards" and one "goalie." The forwards and goalie are distinguished by the assemblage they activate when they find themselves behind the ball: the forwards move to the ball while the goalie remains in the backfield. Both types of player will try to get behind the ball when they find themselves in front of it.

7.3 Design of learning strategies

To isolate the impact of learning on performance, the learning teams were developed using the same behavioral assemblages and perceptual features as the control team. This approach ensures that **the performance of a learning team versus the control team is due only to differences in policy.**

Clay includes both fixed (non-learning) and learning coordination operators. The control team's configuration uses a fixed selector for coordination. Learning is introduced by replacing the fixed mechanism with a learning selector. A Q-learning module is embedded in the learning selector [WD92].

The Q-learner automatically tracks previous perceptions and rewards to refine its policy. At each step, the learning module is provided the current reward and perceptual state. It learns over time to select the best assemblage given the situation.

7.3.1 Reinforcement functions for soccer

The policy an agent learns is likely to depend on the reward function used to train it. One objective of this research is to discover how *local* versus *global* reinforcement impacts the diversity and performance of learning teams. Global reinforcement refers to the case where a single reinforcement signal is simultaneously delivered to all agents, while with local reinforcement each agent is rewarded individually. To that end, we consider two reinforcement functions for learning soccer robots. Assuming the game proceeds in discrete steps, the global reinforcement function at timestep t is:

$$R_{\text{global}}(t) = \begin{cases} 1 & \text{if the team scored at } t-1, \\ -1 & \text{if the opponent scored at } t-1, \\ 0 & \text{otherwise.} \end{cases}$$

This function will reward all team members when any one of them scores. Thus a goalie will be rewarded when a forward scores, and the forward will be punished when the goalie misses a block. Observe that the global reinforcement function and the performance metric (Equation 8.1) are related:

$$P \quad = \quad \sum_{t=0}^{t=N} R_{\text{global}}(t)$$

where N is the number of steps in the game. R_{global} is a performance-based reward. A close correlation between reward function and performance metric is helpful, since reinforcement learning mechanisms seek to maximize their reward. In terms of the taxonomy presented in Chapter 4 R_{global} is an `INTERNAL_SOURCE`, `PERFORMANCE`, `DELAYED`, `DISCRETE` and `GLOBAL` reward function. Now, consider a local function where each agent is rewarded individually:

$$R_{\text{local}}(t) = \begin{cases} 1 & \text{if the agent was closest to the ball} \\ & \text{when its team scored at } t-1, \\ -1 & \text{if the agent was closest to the ball} \\ & \text{when the opposing team scored at } t-1, \\ 0 & \text{otherwise.} \end{cases}$$

Even though global information is required to implement this reward function, in this context `LOCAL` refers to the fact that the reward is based on the individual's performance, not the entire teams'. This function will reward the agent that scores and punish an agent that allows an opponent to score. There may not be much benefit, in terms of reward, for a robot to serve a defensive role in this model since it would receive frequent negative but no positive rewards. In terms of the reward taxonomy, R_{local} is classified the same as R_{global}, except its locality is `LOCAL` rather than `GLOBAL`. The R_{local} reward is `INTERNAL_SOURCE`, `PERFORMANCE`, `DELAYED`, `DISCRETE` and `LOCAL`.

A potential problem with the R_{local} function is the implicit assumption that the agent closest to the ball is the one responsible for a scoring event. It may be that the closest robot just happened to be near the goal while another agent kicked the ball for a score from a distance. To address this, a separate reward function, based on time since the ball was touched was investigated:

$$R_{\text{touch}}(t) = \begin{cases} \gamma^{t_{\text{touch}}} & \text{if the team score at } t-1, \\ -\gamma^{t_{\text{touch}}} & \text{if the opponent scores at } t-1, \\ 0 & \text{otherwise.} \end{cases}$$

t_{touch} is time in milliseconds since the agent last touched the ball. γ is a parameter set to values between 0 and 1 that indicates how quickly a potential reward should decay after the ball is touched. Note that R_{touch} can be written in terms of R_{global}

$$R_{\text{touch}}(t) = R_{\text{global}}(t)\gamma^{t_{\text{touch}}}$$

If $\gamma = 1$, $R_{\text{touch}} = R_{\text{global}}$. As γ is reduced towards 0 the reward becomes progressively more agent-centered or local. The R_{touch} reward is `INTERNAL_SOURCE`, `PERFORMANCE`, `DELAYED`, `CONTINUOUS` and `COMB_LOCALITY`.

7.4 Performance with local and global rewards

The first set of experiments were conducted in the simplified soccer task using the R_{local} and R_{global} reward functions. Experimental data were gathered by simulating thousands of soccer games and monitoring robot performance. The learning robots are evaluated on three criteria: task performance (score), policy convergence, and diversity of behavior.

For each trial, the learning robots were initialized with all Q-values set to zero. A series of 100 10-point games were played. Information on policy convergence and score was recorded after each game. The robots retain their learning set between games. An experiment is composed of 10 runs, or a total of 1000 10-point games. Each run uses the same initial parameters but different pseudo-random number seeds.

7.4.1 Task performance

Performance is measured as the difference between the learning team's score and the opponent's score (Equation 8.1). A negative value indicates the team lost the game, while positive values indicate the team won the game. When rewarded using the global reinforcement signal R_{global}, the learning teams out-score the control team by an average of six points to four, yielding a performance of 2.0. The average includes the initial phase of training. When trained using the local reward R_{local}, the learning teams lose by an average of four points to six, or a performance of -2.0. In these soccer experiments, **teams trained using global reinforcement perform best.**

7.4.2 Learning rate

Learning rate is evaluated by checking for policy convergence. Convergence is tracked by monitoring how frequently an agent's policy changes. Consider a robot that may have been following a policy of moving to the ball when behind it, but due to a recent reinforcement it switches to the *get_behind_ball* assemblage instead. Switches like this are tracked as policy changes.

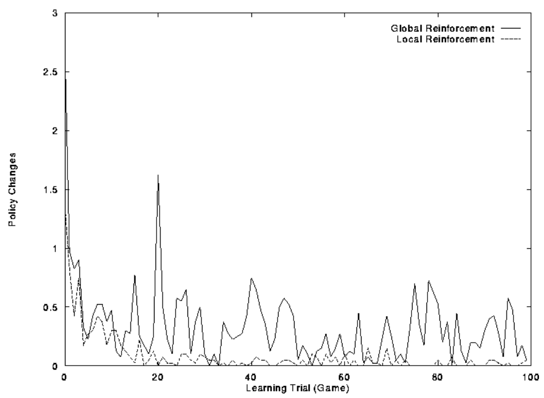

Figure 7.2: Policy convergence measured as average number of policy changes per trial for teams using local and global rewards.

The data, plotted in Figure 7.2, shows good convergence for robots using local rewards. The average number of changes per game drops to 0.05 after 100 games. An individual simulation to 1000 games using the local reward resulted in convergence to zero. The number of policy changes for robots using R_{global} initially decreases, but does not converge in the first 100 games. The average number of policy changes is 0.25 per game after 100 games. In these experiments **teams using local rewards show better policy convergence properties than teams using global rewards.**

Table 7.2: The nine soccer robot policies possible for the learning agents discussed in the text. Each policy is composed of one row for each of the two possible perceptual states (not behind ball or behind ball). The position of the 1 in a row indicates which assemblage is activated for that policy in that situation. The policies of the goalie and forward robots introduced earlier (Figure 2) are in bold.

	mtb	*gbb*	*mtbf*	*mtb*	*gbb*	*mtbf*	*mtb*	*gbb*	*mtbf*
not bb	0	0	1	0	0	1	0	0	1
bb	0	0	1	0	1	0	1	0	0
not bb	**0**	**1**	**0**	0	1	0	**0**	**1**	**0**
bb	**0**	**0**	**1**	0	1	0	**1**	**0**	**0**
not bb	1	0	0	1	0	0	1	0	0
bb	0	0	1	0	1	0	1	0	0

7.4.3 Diversity

After the training phase, robots are evaluated for behavioral diversity by examining their policies. The teams are classified as hetero- or homogeneous depending on whether the robot's policies are the same. Altogether there are 9 possible policies

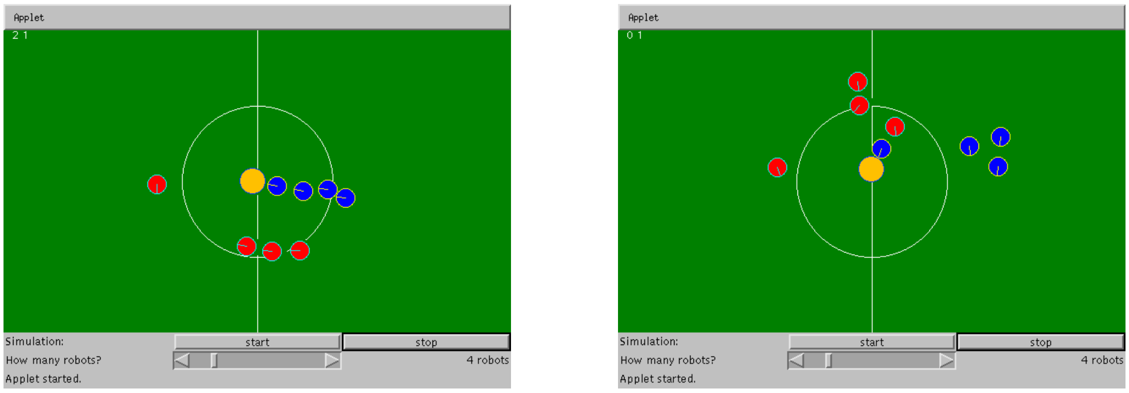

Figure 7.3: Examples of homo- and heterogeneous learning soccer teams. In both cases the learning team (dark) defends the goal on the right. The agents try to propel the ball across the opponent's goal by bumping it. A homogeneous team (left image) has converged to four identical behaviors which in this case causes them to group together as they move towards the ball. A heterogeneous team (right) has settled on diverse policies which spread them apart into the forward and middle of the field.

for the learning agents since for each of the two perceptual states, they may select one of three assemblages. Table 7.2 summarizes the possible policies. Based on these nine policies, there are a total of 6561 possible 4 robot teams.

Two example teams, one homogeneous, the other heterogeneous, are illustrated in Figure 7.3. All members of the team on the left have converged to identical policies. In fact, *all* robots in the 10 locally-reinforced teams converged to the same "forward" policy used by the control team (Table 7.1). All 10 teams converged to fully homogeneous behavior.

In contrast, all of the 10 globally-reinforced teams diversify to heterogeneous behavior. In all cases, the agents settle on one of three particular policies. All the teams include one robot that converges to the same "forward" policy used by the control team; they also include at least one agent that follows the same policy as the control team's "goalie." The other robots learn a policy of always selecting the *get_behind_ball* assemblage, no matter the situation (for convenience this policy is referred to as a "mid-back"). In cases where the team had not fully converged, investigation reveals that the changes are due to one agent alternating between the "goalie" and "mid-back" policies. In summary, the globally-reinforced teams always converged to one "forward," one or two "mid-backs" and one or two "goalies."

To quantify the varying degree of diversity in these teams, social entropy (presented in Chapter 5) is used as a measure of behavioral heterogeneity. Social entropy, inspired by Shannon's Information Entropy [Sha49], evaluates the diversity of a robot society based on the number of behavioral castes it includes and the relative size of each. $H(\mathcal{R})$, the social entropy of the robot society $\mathcal{R}$, ranges from a minimum of zero, when all agents are identical, to a maximum when each robot forms a different caste. The maximum entropy for a team of four soccer robots is 2.0. $H(\mathcal{R}) = 0$ for the homogeneous teams trained using local reinforcement and $H(\mathcal{R}) = 1.5$ for the heterogeneous teams.

7.5 Performance using R_{touch}

Another set of experiments were conducted in the RoboCup soccer task using the R_{touch} reward function for learning. As in the previous experiments, data were gathered by simulating thousands of soccer games and monitoring task performance (score difference), policy convergence, and diversity of behavior. To investigate how the value of γ in R_{touch} impacts performance and diversity, simulations were run as γ was swept from 0.1 and 1.0 in steps of 0.1. For each value of γ, 10 runs of 100 trials were conducted. For each run the simulator was initialized with a different random number seed. Each trial is two simulated minute game. The shorter time was used so that policy changes and performace could be evaluated with a finer resolution than a ten minute trial would permit.

At the beginning of each run the learning robots were initialized with a random policy; Q-values were set to random values between -1 and 1. Next a series of 100 two minute trials are conducted with information on policy convergence and score recorded after each trial. The robots retain their learning set between trials. Each run uses the same initial parameters but different random number seeds.

7.5.1 Task performance

Performance is the difference between the learning team's score and the fixed opponent's score at the end of each trial (Equation 8.1). For each value of γ between 0.1 and 1.0, average performance is computed from the results of 10 experimental runs. These results are plotted in Figure 7.4. In all cases, **the teams trained using R_{touch} out perform the pre-programmed control team**. From the graph it is apparent that **performance improves as γ increases, to a maximum when** $\gamma = 1.0$. When $\gamma = 1.0$ the learning teams out-score the control team by an average

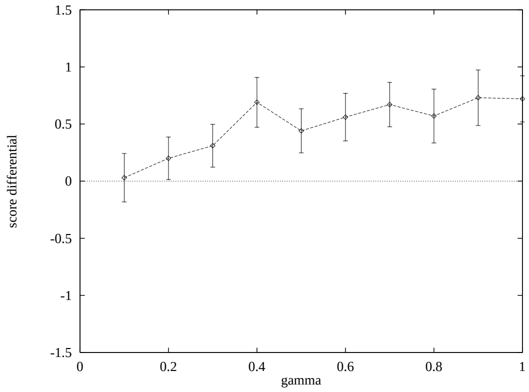

Figure 7.4: Score differentials for teams using the R_{touch} reward function as γ is swept from 0.1 to 1.0. Error bars show 95% confidence intervals. Positive numbers indicate the experimental team is winning on average.

of 0.72 points per two minute trial. This result supports the earlier experiments involving R_{local} and R_{global} that indicated performance was best with global rewards (recall that when $\gamma = 1.0$, $R_{\text{touch}} = R_{\text{global}}$).

7.5.2 Learning rate

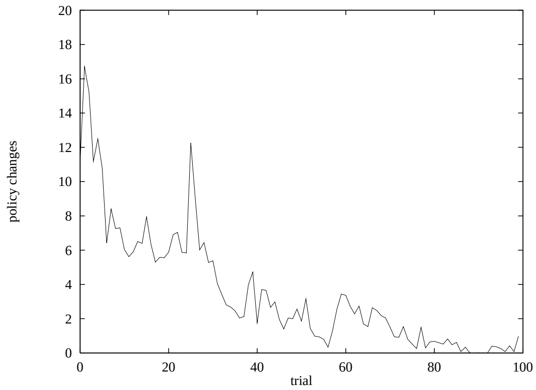

Figure 7.5: Policy changes versus trial number for teams using the R_{touch} reward function ($\gamma = 1.0$).

As in the first set of experiments, learning rate is measured as policy convergence and is tracked by monitoring how frequently an agent's policy changes. The average number of policy changes per trial for 10 runs is plotted in Figure 7.5. This graph is for teams trained using R_{touch} with $\gamma = 1$. Convergence rates are similar for other values of γ as well.

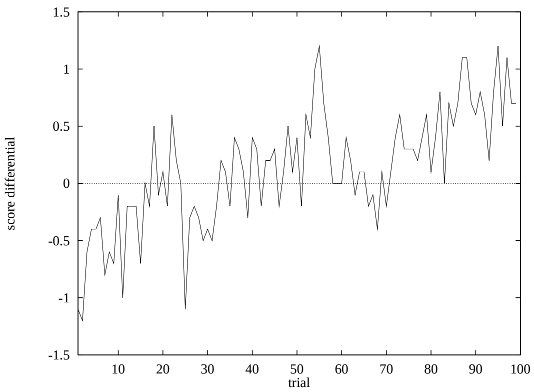

Figure 7.6: Score versus trial number for teams trained using R_{touch} ($\gamma = 1.0$).

Learning rate can also be evaluated by monitoring performance over time. Average performance for teams trained using R_{touch} with $\gamma = 1$ is plotted versus trial number in Figure 7.6. In early trials, performance is negative, but it improves throughout the run, leveling off near 0.7.

Agents trained using R_{touch} show good convergence properties.

7.5.3 Diversity

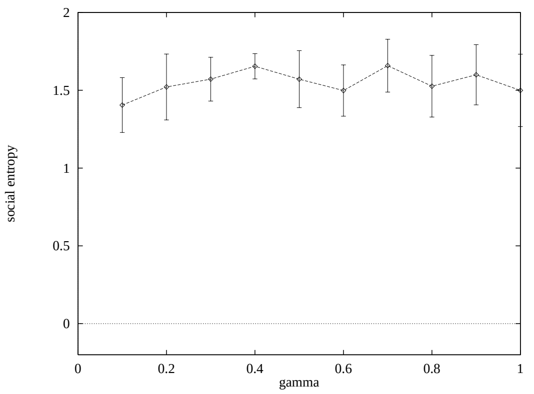

Figure 7.7: Hierarchic social entropy for soccer teams trained using R_{touch} as γ varies from 0.1 to 1.0. Error bars show 95% confidence intervals.

Diversity in is measured after the learning phase is complete using hierarchic social entropy (Chapter 5). For teams of five robots, entropy can range from a minimum of 0.0 (all agents are identical) to 2.32 (all agents are different). The graph in Figure 7.7 plots diversity for learning soccer teams as γ is swept from 0.1 to 1.0. Measured diversity is approximately 1.5 for all values of γ. The data indicate that

diversity is not impacted by γ in robot teams trained using R_{touch}.

Recall that when $\gamma = 1.0$, R_{touch} returns a reward equivalent to R_{global}. Note that even when γ is set to a small value R_{touch} will always return some positive or negative reward to all agents whenever a scoring event occurs. In this regard, R_{touch} is a global reward function, regardless of γ. This may be why diversity is apparent in the soccer teams across all values of γ. It is also interesting to note that the level of diversity in these learning teams (1.5) is the same as that found in the experiments in the simplified soccer domain with fewer robots.

7.6 Performance versus human-designed teams

The results reported up to this point show that a simulated robot team can learn a winning soccer strategy against a fixed control team. The learning teams are provided the same behavioral assemblages as the fixed opponent so that any difference in performance is due to the sequencing strategies the agents learn, not the behavioral assemblages themselves. This experimental approach leaves open the possibility that

1. the strategy utilized by the fixed opponent team may be poor, and finding a way to beat it is easy, or

2. the behavioral assemblages may be too simple and could never be utilized in a really successful robot soccer strategy.

Either of these possibilities would reduce the significance of the results.

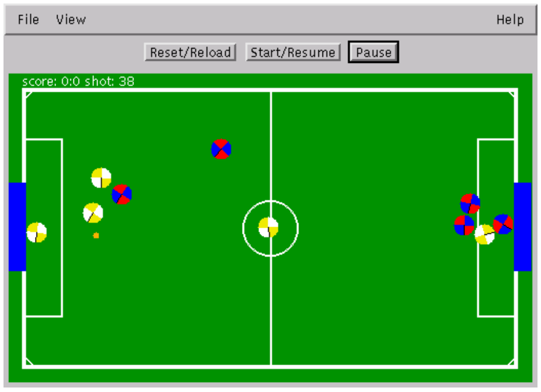

Figure 7.8: Example team trained using R_{touch} in trials against **DTeam**. The dark colored learning agents defend the goal on the right. Three agents have converged to a defensive role while two play offensive positions.

To address this, learning soccer teams were tested in experiments against teams developed by others. Students in classes[1] taught in the College of Computing at Georgia Tech were assigned the task of developing a multiagent robot soccer team using the JavaBots system. The students were provided the same fixed opponent team used in the earlier experiments as a "straw-man" for testing their own teams. Since students' grades were linked to how well their teams performed, it can be assumed they did their best to develop effective strategies. Approximately 16 teams were developed by students in these classes. The best three were chosen for evaluation here. All three of these teams used fixed strategies, they do not learn:

- **BriSpec** designed by Brian McNamara. The members of this team play three different positions. One player always remains at the back of the field; it aligns itself between the ball and the goal. Three players play mid-field positions; they stay behind the ball and attempt to spread out from one another. The remaining agent stays in front of the ball, in expectation of a pass.
- **DTeam** by David H. Johnson. The players on this team diversify to fill four specialized roles. Two of the players exploit weaknesses in the simulator dyanmics and soccer rules as follows: One player always moves to block the opponent's goalie. The simulated dynamics are such that one player cannot push another, so the blocking is usually effective. Another player always waits near the center of the field for the ball to appear. This behavior exploits the simulation's deadlock prevention scheme; the ball is repositioned to the center of the field when 60 simulated seconds elapse with no score. This player often is the first to get the ball when it is repositioned. The remaining three players serve as a goalie and two forwards. DTeam was programmed using Clay.
- **Kechze** by Kent Lyons, Christopher Jurney and Zellyn Hunter. Kechze is similar to DTeam in that it exploits the ball re-positioning rule of the simulation. This team is rule-based however (instead of using motor schemas as DTeam does). Like the other teams, Kechze players fill specialized roles. But Kechze has an important refinement that enables it to improve its performance: the agent assigned to wait in the middle of the field for the ball does not do so until a certain time elapses. This delay is in recognition of the simulator's timed ball relocation scheme. The Kechze team gains full use of that player for more time. In contrast, the player that fills this role on DTeam always moves directly to the center position and does not contribute to play until the ball is re-located.

The R_{touch} reward function with $\gamma = 1.0$ was used to train agents in learning trials against these teams. For each of the three experiments, the learning agents were initialized with a random policy (Q-values were set to random values between

[1]CS 7100 taught in Fall 1997 by Irfan Essa and CS 4324 taught in Spring 1998 by Chris Atkeson.

-1 and 1). Next, a series of 200 two minute trials were played between the learning teams and each opponent team.

Overall performance is evaluated as the average score difference in the last 10 trials of each experimental run. Plots of performance versus learning trial for each of the three opponent teams are provided in Figures 7.9 and 7.10. In each case, the learning teams converge to a winning strategy with a positive (winning) score differential. An example team trained versus **DTeam** is illustrated in Figure 7.8. In these experiments **learning soccer teams using the behaviors developed in this work out perform the best human-designed strategies**.

Table 7.3 summarizes these results, as well as the other experiments examined in this chapter. The only losing learning teams were those programmed to use the R_local reward. All other teams converged to winning and relatively diverse strategies.

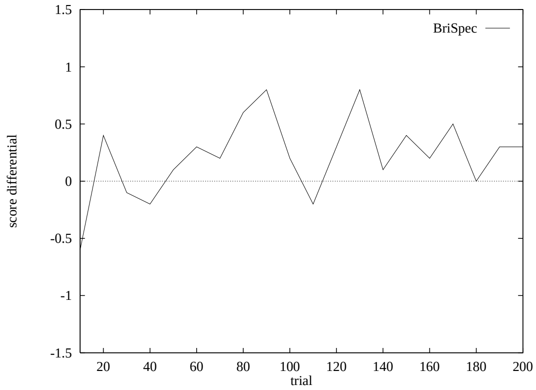

Figure 7.9: Performance versus trial number for games against the **BriSpec** team. Each point is the average performance of the learning team over 10 trials.

7.7 Discussion and summary

The relative benefits of three different reinforcement functions for robot soccer teams have been evaluated in terms of team performance, learning rate, and social entropy in the resulting team. The three reward functions, R_{local}, R_{global} and R_{touch} were evaluated on learning teams as they engaged a fixed opponent team and three other human-designed teams in thousands of trials. The primary results are

- individual learning robots will, in many cases, automatically diversify to fill different roles on a team;

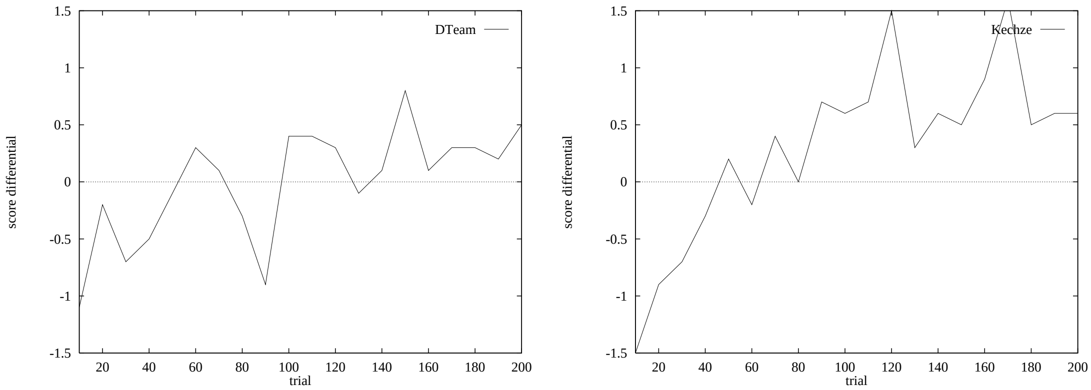

Figure 7.10: Performance versus trial number for games against two human-designed teams: **DTeam** (left) and **Kechze** (right). Each point on each curve represents the average performance of a learning team over 10 trials.

Table 7.3: Performance and diversity results from robot soccer experiments. Except in the case of agents trained using R_{local} all teams converge to winning strategies.

reward function	opponent	domain	performance	social entropy
R_{local}	fixed	simplified	-2.0	0.0
R_{global}	fixed	simplified	2.0	1.5
R_{touch}, $\gamma = 0.1$	fixed	RoboCup	0.0	1.4
R_{touch}, $\gamma = 1.0$	fixed	RoboCup	0.7	1.5
R_{touch}, $\gamma = 1.0$	BriSpec	RoboCup	0.3	1.7
R_{touch}, $\gamma = 1.0$	DTeam	RoboCup	0.5	1.9
R_{touch}, $\gamma = 1.0$	Kechze	RoboCup	0.6	1.2

- after a training period, teams of learning robots out perform the best human-designed teams;
- global reinforcement leads to better performance and greater diversity, but slow policy convergence for robot teams;
- local reinforcement leads to poorer performance and more homogeneous behavior, but faster policy convergence.

The performance of teams using R_{local} and R_{global} for learning in a simplified soccer domain show that local rewards provide quicker learning, while global reinforcement leads to better performance and greater diversity. Also, the globally-reinforced teams perform significantly better than the pre-programmed control team. The locally-reinforced teams converge to "greedy" behaviors that maximize their individual reward, but lead to poor team performance. This may suggest that defensive play is important in soccer but there is no incentive for a robot to fill a defensive role. With the local reward strategy a goalie would be "punished" every time the opponent scores and never receive a positive reinforcement. Quick convergence in the locally-reinforced teams is due to the close relationship between an individual agent's actions and the rewards it receives with local reinforcement strategies.

Additional experiments were conducted in the RoboCup task using the R_{touch} reward function. This function provides a reward based on time since the robot last touched the ball. If a goal is scored and the agent touched the ball recently, its reward is greater than than if it touched the ball further in the past. A parameter of the reward function, γ sets the rate at which the reward decays. Rewards using R_{touch} reward function with $\gamma = 1.0$ are identical to those generated by the R_{global} function. Experiments conducted by sweeping γ from 0.1 to 1.0, show that performance is best with $\gamma = 1.0$. Diversity is not impacted by the value of γ; all teams using R_{touch} converged behavioral diversities of approximately 1.5 (the same as teams using global reinforcement). This result is probably due to the fact that, no matter what value γ is set to, all robots receive some non-zero reward at every scoring event – hence the reward always has a global nature.

In all of these experiments a fixed opponent team was configured from the same behaviors available to the learning teams. This approach was utilized to ensure that differences in performance were due to a team's policy or learning strategy and not the behaviors from which it selects. This leaves open the possibility however, that the fixed opponent is easy to beat, thus the learning systems are not adequately challenged. Experiments against three human-developed soccer teams were conducted to address this. In all three cases **learning teams out-performed the human-developed soccer teams**.