Integrating Learning with Motor Schema-Based Control for a Robot Soccer Team

Tucker Balch

Mobile Robot Laboratory
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-208 USA

Abstract. This paper describes a reinforcement learning-based strategy developed for Robocup simulator league competition. In overview: 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. The entire team is jointly rewarded or penalized when they score or are scored against (global reinforcement). This approach provides for diversity in individual behavior.

1 Introduction

Motor schemas are an important example of behavior-based robot control. The motor schema paradigm is the central method in use at the Georgia Tech Mobile Robot Laboratory, and is the platform for this research. Motor schemas are the reactive component of Arkin's Autonomous Robot Architecture (AuRA) [1]. AuRA's design integrates deliberative planning at a top level with behavior-based motor control at the bottom. The lower levels, concerned with executing the reactive behaviors are incorporated in this research.

Individual motor schemas, or primitive behaviors, express separate goals or constraints for a task. As an example, important schemas for a navigational task would include **avoid_obstacles** and **move_to_goal**. Since schemas are independent, they can run concurrently, providing parallelism and speed. Sensor input is processed by perceptual schemas embedded in the motor behaviors. Perceptual processing is minimal and provides just the information pertinent to the motor schema. For instance, a **find_obstacles** perceptual schema which provides a list of sensed obstacles is embedded in the **avoid_obstacles** motor schema. Motor schemas may be grouped to form more complex, emergent behaviors. Groups of behaviors are referred to as *behavioral assemblages*. One way behavioral assemblages may be used in solving complex tasks is to develop an assemblage for each sub-task and to execute the assemblages in an appropriate sequence. The resulting task-solving strategy can be represented as a Finite State Automaton (FSA). The technique is referred to as *temporal sequencing* [1].

Even though behavior-based approaches, like the motor schema paradigm are robust for many tasks and environments, they are not necessarily adaptive.

When feedback regarding success in a task is available, *reinforcement learning* can shift the burden of behavior refinement from the designer to the robots operating autonomously in their environment. For some simple tasks, given a sufficiently long trial, agents are even able to develop optimal policies [5]. Rather than attempting to design an optimal system from the start, the designer imbues his robots with adaptability. The robots strive continuously to improve their performance; finding suitable behaviors automatically as they interact with the environment. For these reasons reinforcement learning is becoming pervasive in mobile robot research.

Q-learning is a type of reinforcement-learning in which the value of taking each possible action in each situation is represented as a utility function, $Q(s,a)$. Where s is the state or situation and a is a possible action. If the function is properly computed, an agent can act optimally simply by looking up the best-valued action for any situation. The problem is to find the $Q(s,a)$ that provides an optimal policy. Watkins provides an algorithm for determining $Q(s,a)$ that converges to optimal [8].

The Java-based soccer control system described here (Figure 1) was originally developed for research using a simulator developed at Georgia Tech [4]. The system is being converted for operation with the Soccer Server simulation system developed by Noda [6].

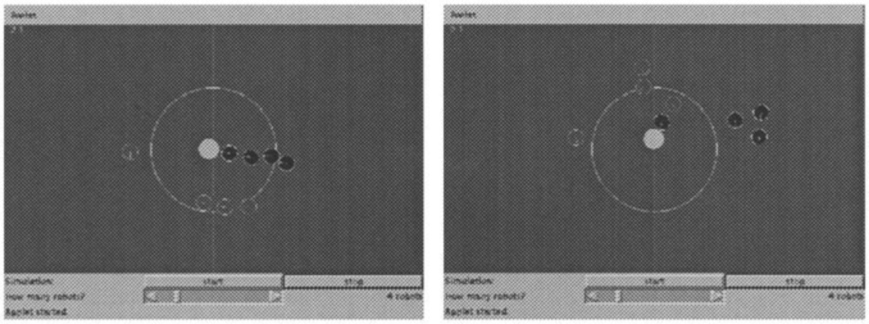

Fig. 1. Examples of homo- and heterogeneous learning soccer teams. In both cases the learning team (dark) defends the goal on the right. The agents try to propel the ball across the opponent's goal by bumping it. A homogeneous team (left image) has converged to four identical behaviors which in this case cause 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 middle and back of the field.

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 (it is best to humiliate the opponent!). Hence, the performance metric for robot teams is

$$P = S_{\text{us}} - S_{\text{them}} \tag{1}$$

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

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. Rolling friction is modeled with constant deceleration after the bump. Each agent is provided the following synthetic 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.
- **Team sensor:** returns an array of egocentric vectors pointing to the robot's team members.
- **Enemy 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.

The ball position, robot ID and defended goal sensors are used in the experimental robots examined here. At present, the sensors are perfect. Future revisions of the simulator may address real-world issues like noise, sensor 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.

The sensor and actuator interface closely parallels those available on commercial robots. An eventual goal is to verify this work by porting the system to four Nomadic Technologies Nomad 150 robots in Georgia Tech's Mobile Robot Laboratory.

2 Behaviors for Soccer

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, 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. The system is outlined at a high level here. For more detail the reader is referred to [2].

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

Control Team Forward

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

Control Team Goalie

Fig. 2. The control team's strategy viewed 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.

Experiments 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 will be 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 two 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.

To implement this design, each robot is provided a set of behavioral assemblages for soccer. Each assemblage can be viewed as a distinct "skill" which, when sequenced with other assemblages forms 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. The strategies may be equivalently viewed as lookup tables (Figure 2). This paper will focus on the lookup table representation since it is also useful for discussing learned policies. The behavioral assemblages developed for these experiments are:

- *move_to_ball (mtb)*: The robot moves directly to the ball. A collision with the ball will propel it away from the robot.
- *get_behind_ball (gbb)*: The robot moves to a position between the ball and the defended goal while dodging the ball.
- *move_to_back_field (mtbf)*: The robot moves to the back third of the field while being simultaneously attracted to the ball.

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

The 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 (*mtb*) while the goalie remains in the backfield (*mtbf*). Both types of player will try to get behind the ball (*gbb*) when they find themselves in front of it.

3 Learning Soccer

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, thus: **the relative performance of a learning team versus the control team is due only to learning.**

Recall that Clay (the system used for configuring the robots) 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 [8] module is embedded in the learning selector. It is acknowledged that other types of reinforcement learning approaches are also appropriate for this system. Q-learning was selected arbitrarily for this initial study. Future investigations may be undertaken to evaluate the impact of learning type on robotic systems.

At each step, the learning module is provided the current reward and perceptual state, it returns an integer indicating which assemblage the selector should activate. The Q-learner automatically tracks previous perceptions and rewards to refine its policy.

The policy an agent learns depends directly 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:

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

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 1) are related by:

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

where N is the number of steps in the game. A close correlation between reward function and performance metric is helpful, since reinforcement learning mechanisms seek to maximize their reward. If the reward and the performance measure are similar, the agent stands a better chance of maximizing its performance. Now, consider a local function where each agent is rewarded individually:

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

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.

4 Results

A static (non-learning) version of our Java-based soccer system was completed and utilized in the RoboCup-97 competition. Unfortunately, there was not enough time to integrate the learning system. The results reported below were generated in Georgia Tech's Java-based simulator.

Experimental data were gathered by simulating thousands of soccer games and monitoring robot performance. The learning robots are evaluated on three criteria: objective performance (score), policy convergence, and diversity of behavior.

For each trial, the learning robots are initialized with a default policy (all Q-values set to zero). A series of 100 10-point games are played with information on policy convergence and score recorded after each game. The robots retain their learning set between games. An experiment is composed of 10 trials, or a total of 1000 10-point games. Each trial uses the same initial parameters but different random number seeds (the simulations are not stochastic, but Q-learning is).

4.1 Objective Performance

When rewarded using the global reinforcement signal R_{global}, the learning teams out-score the control team by an average of 6 points to 4. The average is for all games, even during the initial phase of training. The winning margin is notable since the losing control team was hand-coded. When trained using the local reward R_{local}, the learning teams lose by an average of 4 points to 6.

4.2 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

	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

Fig. 3. 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 - *bb*). 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.

when behind it, but due to a recent reinforcement it switches to the *get_behind_ball* assemblage instead. These switches are tracked as policy changes.

The data for robots rewarded using the local signal shows good convergence. The average number of changes per game drops to 0.05 after 100 games. An individual simulation to 1000 games 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. Future simulation studies will include longer simulation runs to investigate whether convergence occurs eventually.

4.3 Behavioral 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 for the learning agents since for each of the two perceptual states, they may select one of three assemblages. Figure 3 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 1. The team on the left has converged to identical policies. In fact, *all* robots on the 10 locally-reinforced teams converged to the same "forward" policy used by the control team (Figure 2). 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 settle on 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 (zero policy changes per game), 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 help quantify the varying degree of diversity in these teams, *Social Entropy* [3] is used as a measure of behavioral heterogeneity. Social Entropy, inspired by Shannon's Information Entropy [7], evaluates the diversity of a robot society based on the number of behavioral castes it includes and the relative size of each. $\mathrm{Het}(\mathcal{R})$, 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. $\mathrm{Het}(\mathcal{R}) = 0$ for the homogeneous teams trained using local reinforcement and $\mathrm{Het}(\mathcal{R}) = 1.5$ for the heterogeneous teams. For more detail on Social Entropy, the reader is referred to [3].

5 Discussion and Conclusion

The results reported above show that in this task local reinforcement provides quicker learning, while global reinforcement leads to better performance and greater diversity. The globally-reinforced teams perform significantly better than the human-designed control team.

The locally-reinforced teams converge to "greedy" behaviors that maximize their individual reward, but lead to poor team performance. This is probably because 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.

The globally-reinforced teams perform better but do not converge to stable policies. It may be that longer experimental runs will show convergence with R_{global} reinforcement. It may also be that for complex multi-robot domains, convergence does not always occur. Either way, convergence is not a requirement for good performance: the globally rewarded teams perform significantly better than the locally reinforced teams in spite of a lack of convergence.

To conclude, the relative benefits of local versus global reinforcement in learning robot soccer teams has been evaluated in terms of team performance, learning rate, and social entropy in the resulting team. The teams were evaluated as they engaged a fixed opponent team over thousands of trials. In summary, the primary results are:

- Individual learning robots will, in many cases, automatically diversify to fill different roles on a team.
- Teams of learning robots can better the performance of 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 fully homogeneous behavior, but fast policy convergence.

The author thanks Ron Arkin, Chris Atkeson, Gary Boone, John Pani and Juan Carlos Santamaria for their comments on this work.

References

1. R.C. Arkin and T.R. Balch. Aura: principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997. to appear.
2. T. Balch. Clay: Integrating motor schemas and reinforcement learning. College of Computing Technical Report GIT-CC-97-11, Georgia Institute of Technology, Atlanta, Georgia, March 1997.
3. T. Balch. Social entropy: a new metric for learning multi-robot teams. In *Proc. 10th International FLAIRS Conference (FLAIRS-97)*, May 1997.
4. Tucker Balch. Learning roles: Behavioral diversity in robot teams. In *AAAI-97 Workshop on Multiagent Learning*, Providence, R.I., 1997. AAAI.
5. L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
6. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proc. Autonomous Agents 97*. ACM, 1997. Marina Del Rey, California.
7. C. E. Shannon. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.
8. Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q learning. *Machine Learning*, 8:279–292, 1992.