Co-evolving Soccer Softbots

The University of Maryland RoboCup simulator entry (Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler) consisted entirely of computer-*evolved* players developed with Genetic Programming [Koza 1992]. Genetic Programming is a branch of evolutionary computation which uses natural selection to optimize over the space of computer algorithms. Unlike other entrants who fashioned fine-tuned softbot teams from battery of relatively well-understood robotics techniques, Maryland's goal was to see if it was even *possible* to use evolutionary computation to develop high-level soccer behaviors that were competitive with the human-crafted strategies of other teams. While evolutionary computation has been successful in many fields, evolving a computer algorithm has proven challenging, especially in a domain like robot soccer.

Luke *et al* used Genetic Programming to evolve a population of "teams" of LISP s-expression algorithms, evaluating each team by attaching its algorithms to robot players and trying them out in the simulator. Early runs tested individual players; later runs pitted whole teams against each other using *co-evolution*. After evaluation, a team's fitness assessment was based on its success relative to its opponent. This fitness score determined which teams would be selected to interbreed and form the next generation of algorithms.

The RoboCup soccer simulator makes evolutionary computation extremely difficult. The simulator gives noisy data, limited sensor information, and complex dynamics. Most problematic is that the simulator runs in real-time; even at full-throttle, games can take many seconds or minutes. Unfortunately, evolving a team of eleven soccer players can require hundreds of thousands of evaluations, so in the worst case a single soccer-evolution run might take a year or more to complete.

In order to keep the number of evaluations to a minimum, Maryland severely limited the population size, which demanded special customizations to prevent the population from converging to a suboptimal strategy. Maryland also cut down the number of evolved algorithms by grouping players into "squads", with one algorithm per squad, or by using one single algorithm for the entire team [Luke and Spector 1996]. They performed runs for both strategies; at the time of the competition, the single-team strategies had better fitness. To further speed up runs, evaluations were run in parallel on an Alpha supercomputer cluster.

As they had only one shot to evolve teams, Luke *et al* cannot make rigorous scientific claims as to population development. Nonetheless an admittedly anecdotal observation is still interesting. After a hesitant start, most early teams soon began to learn the worrisome suboptimal "kiddie soccer" strategy: "everyone go after the ball and kick it to the goal" (Figure 1). Thankfully, players learned to hang back and protect the goal, and ultimately to disperse through the field to provide better coverage (Figure 2).

By the end of the final runs, the computer had produced teams which passed to teammates, blocked the ball, protected different parts of the field, and tried to stay open. Maryland's entry was successful in competition, beating its first two hand-coded competitors before succumbing.

References

J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge MA, 1992.

S. Luke and L. Spector. Evolving Teamwork and Coordination with Genetic Programming. In J.R. Koza et al., editors, *Proceedings of the First Annual Conference on Genetic Programming (GP-96)*. The MIT Press, Cambridge MA, pages 150–156, 1996.

Figures

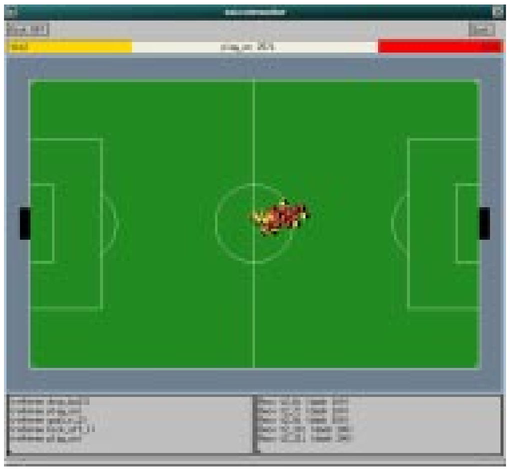

Figure 1. "Everyone go after the ball":a troublingly suboptimal early team strategy.

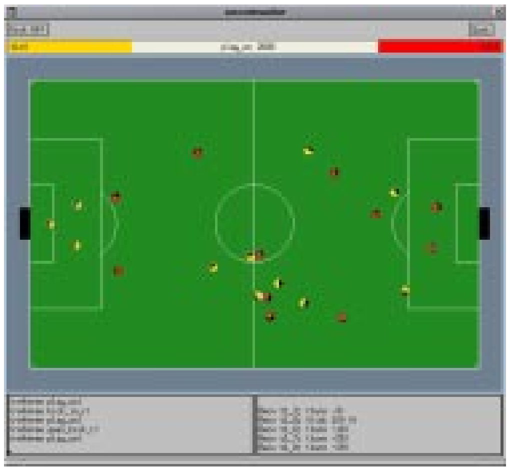

Figure 2. Soccer teams eventually learn to disperse themselves throughout the field.