# Genetic Programming for Strategy Learning in Soccer Playing Agents: A KDD-Based Architecture

**Steven M. Gustafson** **William H. Hsu**

Kansas State University
234 Nichols Hall, Manhattan, KS 66506
{steveg | bhsu}@cis.ksu.edu

## Abstract

A KDD-based architecture should serve as a good framework to learn an improved strategy of ball control for intelligent soccer playing agents. Current work on using genetic algorithms to improve large scale data mining has been successful and provides an architecture for implementing future systems. This architecture is well suited for genetic programming and it is proposed that it can be extended to such applications. By learning "real-world" strategies, the performance of robot agents can be improved and become more similar to that of their biological counterparts. *Layered learning* is used to learn high-level behaviors that encompass previously learned low-level behaviors. Three phases of implementation are presented, with the goal of reducing the difference between soccer agent learning and human soccer learning.

## 1 BACKGROUND

Current work for the Knowledge Discovery in Databases (KDD) research lab at Kansas State University is in the area of using genetic algorithms to perform feature subset selection and feature construction for large-scale KDD problems, namely those with a large number of irrelevant features [HCGG00]. This work involves an inductive algorithm and the evaluation of performance of a particular subset of features. The system was built in an architecture that was meant to be easily extended to other data mining problems through the reuse of core modules. The system was developed in a visual programming tool called *D2K* (Data To Knowledge), which is under continuing development [HWRC00]. The design of the specific system to implement a genetic algorithm in *D2K* lends itself well to developing other genetic methods which require the evaluation and modification of populations from generation to generation. For this reason, this particular KDD architecture is being extended to learning using a genetic programming method where the extension models traditional genetic programming engines.

The RoboCup competition has served for the past several years as the testing bed for intelligent soccer agents and has been shown to a be a good test domain for multiagent learning [Lu98]. Through the past competitions some entries have attempted to develop their soccer agents by means of learning rather than hand-coding instructions. Some have tried using genetic programming [Lu98], backpropagation in neural networks [MNH96], and others have used methods such as layered learning [SV99a] for reinforcement learning [SV99b]. However, many of these learning methods are geared specifically for the task of performing well in the simulated environment and to beat other simulated teams of agents. While some have been more successful than others, few have developed human-level strategies.

Current success of RoboCup teams that use learning to develop behavior often rely on heuristics that can be counterintuitive to the real-world game of soccer. An example from [SV99b] is that the amount of ball advancement towards the opposing goal can be tracked as a measure of good performance. However, in real-world soccer, ball possession is far more important than simply advancing the ball downfield. This means playing the ball backward towards the defending goal to maintain possession is often a desirable behavior. These types of behaviors are ones that need to be developed to produce highly competitive soccer agent teams.

Intelligent learning agents, especially genetically programmed agents, possess the potential for learning highly complex strategies [Ko92], which can be too complex for hand-coded methods. To make substantial advancements in the modeling of real human soccer players, more highly developed strategies need to be learned. The strategy I propose to learn is one that is crucial to playing the game of soccer, that of *ball passing and possession*. If players can effectively control the ball by successfully passing the ball to other teammates, they have eliminated the chance of the other team from even

scoring (without the ball, the other team cannot possibly make a goal).

The following sections describe the major portions of this research. Section 2 illustrates the extension of the KDD architecture to a genetic programming engine. Section 3 discusses genetic learning, hierarchical learning, and the specific problem and proposed solution. Current work is discussed in section 4.

# 2 KDD APPROACH TO INTELLIGENT AGENTS

The *D2K* tool was designed as a framework to easily build data mining applications in. Researchers working with *D2K* implemented a genetic wrapper algorithm to optimize feature subset selection in large data sets. The success of the genetic algorithm suggests that it can be used as a genetic programming framework as well. The specific learning task of the genetic algorithm is described next to illustrate its similarities to a typical genetic programming engine.

The KDD architecture [HCGG00] uses supervised learning where the inductive algorithm is decision tree learning. A genetic wrapper is used to wrap around successive trials of the inductive algorithm with different subsets of features. The work extends existing methods with the exception of the genetic wrapper that attempts to optimize each successive generation for smaller sets of features, performance on a test set, and size of the tree induced. The architecture has in place methods for managing the different sets of features, the data sets, the communication between master and slave processes, and the evaluation of performance. This architecture is also flexible because of its design in *D2K* [HWRC00], being modular and object oriented. It is somewhat optimized to minimize the time and resource consuming task of evaluating generation to generation of feature subsets. Next, a solution to learning better strategies for soccer playing agents is proposed, following a discussion of genetic learning.

# 3 GENETIC LEARNING

Genetic programming comes from the Darwian principle of survival of the fittest. It is chosen for this research project for its closeness to the biological method that it mimics. To learn a very humanistic strategy, the learning method should mimic human methods as much as possible. Genetic programming is similar to genetic algorithms in that it maintains a core set of items that will be used to develop the best set of items across generations. The correlation from the architecture used in the genetic wrapper problem to a genetic programming system is straightforward. The master process, or wrapper, can maintain a representation of the order and collection of functions and send those off to the slave processes. The slave processes can then execute in a simulated environment and collect results to be sent back to the master. The master can then determine the best, or most fit, individuals from that population and perform genetic operations on them, such as reproduction, crossover and mutation, to create a new generation of *agent descriptions*. For ease of experimentation, individuals are single-agent descriptions (including behavior parameters) rather than team descriptions. Again, the master will send the population to the slave processes for execution and evaluation [IHK96].

## 3.1 GA VS. GP

The distinction between the genetic algorithm in current work and genetic programming is subtle at first, then obviously different. The GA is concerned with finding optimal features from a large set of features to learn a target concept. The operations of genetic methods like crossover are typically applied to bit masks and then applied to a vector of the features to determine which features in that individual are active. In GP, we are concerned with building a program made out of core functions, instead of features. These functions can take the form as (kick-goal i k) as in [Lu98] or (say *Message*) as in [IHK96] where *Message* is a message to be communicated in the Soccer Server. A decision tree is oftern formed with GP and the genetic operations are performed on the tree.

## 3.2 PROPOSED HIERARCHY OF BEHAVIORS

Stone and Veloso [SV99a] introduced the *layered learning* strategy for learning complex behaviors by breaking the behavior into independent layers of tasks. This paradigm seems especially good for modeling biological processes as it is easy to think of the complex actions humans perform on a regular basis as being composed of several, smaller tasks. For example, when learning to pass a soccer ball to a teammate, a player first must master the skill of balance and coordination of moving their feet without falling. Next, the player needs to develop the skill determining how hard to pass the ball and in what direction. Also, the skill of using one‘s feet to kick a ball in the desired direction and with a particular velocity needs to be learned. Watching professional soccer athletes of today will demonstrate that humans can often master the high level task of passing, but at times fail when other tasks such as balance break down.

## 3.3 PROBLEM DEFINITION AND TRAINING

The proposed hierarchy, extending upon layered learning, uses three levels of learning, three tasks which build upon each other to learn one high level task. Each level will be

learned separately using genetic programming in the KDD architecture. The lowest, or primitive, level is that of simply passing a ball to a fixed point. The second level is learning to pass to a player moving in a given direction with a given velocity, further work may include accounting for accelaration. The highest level task is to learn to coordinate among mulitple players the task of moving to an acceptable position to accept or give a pass (see Table 1). A common tactic used in real world soccer is to always have a triangle of players, where the point of the triangle is the player with the ball and the other two points are other teammates with defender-free paths between them and the player with the ball. This strategy allows the player with the ball to always have a place to pass the ball. Each player without the ball can move to free himself from being in the path of a defender.

*Table 1. Hierarchy of Behaviors*

| | |
|---|---|
| **Global Behavior** | *Coordinate player movement and passing* |
| **Intermediate Behavior** | *Pass to player i* |
| **Primitive Behavior** | *Pass in direction x* |

The first implement will be simple passing between two players, where the players move downfield towards the opposing goal. Next is to incorporating a defender and forcing the players to move and pass around the defender. Last is implementing the common practice technique of keep-away where one defender is in the middle of a triangle made from three players. The three players must move and pass to keep the ball away from the defender in the middle, see Figure 1. These types of activities mimic training for real human soccer players. In this way, learning more closely models human players and creates a better all around soccer player with more refined basic techniques.

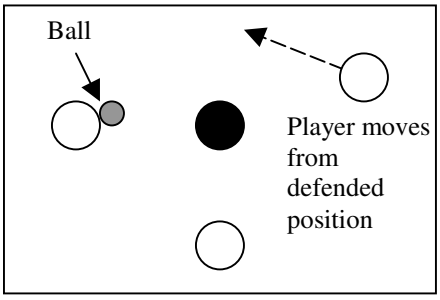


*Figure 1. Soccer triangles drill with one defensive player (black), and three offensive players.*

There are several possibilities for evaluating performance of individuals in a generation. Individuals can perform in a round-robin tournament or with a randomly selected percentage of the population. For more advanced task learning, like the triangle problem, running simulations against the Soccer Server agents could be beneficial. The outcomes of matches are then used for reinforcement learning [RN95] to evaluate each individual.

The evaluation of the learned strategy can be made by comparing it to other learning and hand coded agents. For example, the Soccer Server allows the controlling of the number of players, or agents, from each team. A good test for learning the high-level strategy of ball possesesion is to place three genetically programmed agents with three others, such as Q-learning agents, or hand coded agents, and evaluate their performace. Another possibility is to use tournament selection, where the genetically programmed agents perform against each other to determine the winner, the more successful strategy.

Current work with genetic wrappers use a network of workstations (Beowulf cluster) to deploy slave processes on and execute decision tree learning in parrallel. This is a good resource for learning with genetic programming. Additional resources may be availble at NCSA, the research group's parent organization.

The implications of succeeding at these goals are that more and more higher level tasks can be learned such as team play, offensive strategies against specific styles of play, defensive strategies, and so on.

The main goal of this research is to learn an improved, more humanistic, soccer playing strategy. When this strategy is learned, it an be incorporated into an even higher level team oriented strategy, with the objective of outperfirming other learning and hand coded agents. Secondly, is it hoped to demonstrated that genetic programming has the ability to evolve very complex programs.

## 4 CURRENT WORK

The success of the KDD framework in a current GA-based data mining application [HWRC00] shows good potential as an architecture for genetic programming. The goal of my project is to create agents with more real-world training that model human soccer players more closely. A layered learning approach will be used with the core learning method at each level being genetic programming. After the global behavior is learned, it can be applied towards three goals: two player passing, two player passing with a defender, and lastly, the triangle keep-away game.

Once implemented, experiments will be performed to compare this method with other reinforcement learning RoboCup teams that use methods such as *Q*-Learning, or hybrids of *Q*-Learning [SV99b], and hand coded strategy teams.

Current work consists of implementing, testing and evaluating the proposed project and later incorporating these learned strategies into a team to compete in a RoboCup competition. Note that the target behavior is not currently a complete team strategy and is not concerned with soccer field position. Later work will be used to determine when it is beneficial to attempt to advance downfield or when to pass backwards to maintain possession of the ball.

**References**


[HCGG00] Hsu, W. H., Cheng, Y., Guo, H., Gustafson, S. M. 2000. Genetic Algorithms for Reformulation of Large-Scale KDD Problems with Many Irrelevant Attributes. In *Proceedings of GECCO-2000*, Las Vegas, Nevada, to appear.

[HWRC00] W. H. Hsu, M. Welge, T. Redman, and D. Clutter. Constructive Induction Wrappers in High-Performance Commercial Data Mining and Decision Support Systems. NCSA technical report, URL: http://chili.ncsa.uiuc.edu, to appear, 2000.

[IHK96] Itsuki, N., Hitoshi, M., Kazuo, H. Learning Cooperative Behavior in Multi-agent Environment: --- a case study of choice of play-plans in soccer ---. *PRICAI'96: Topics in Artificial Intelligence (Proc. of 4th Pacific Rim International Conference on Artificial Intelligence, Cairns, Australia)*, pp. 570--579, Aug. 1996.

[Ko92] Koza, J.R. 1992. *Genetic Programming*. MIT Press.

[Lu98] Luke, S. 1998. Genetic Programming Produced Competitive Soccer Softbot Teams for RoboCup97. *Proceedings of the Third Annual Genetic Programming Conference (GP98)*. J. Koza *et al*, eds. 204-222. San Francisco: Morgan Kaufmann.

[MNH96] Matsubara, H., Noda, I., Hiraki, K. (1996). Learning of Cooperative Actions in Multi-Agent Systems: a case study of pass in Soccer. *AAAI-96 Spring Symposium on Adaptation, Coevolution and Learning in Multi-agent Systems*, SS-96-01, pp. 63--67, Mar. 1996.

[RN95] Russell, S., Norvig, P. 1995. *Artificial Intelligence, A Modern Approach*. Prentice-Hall, Inc.

[SV99a] Stone, P., & Veloso, M. 1999. Layered Learning. *Proceedings of the IJCAI-99 Workshop on Learning About, From, and With Other Agents*.

[SV99b] Stone, P., & Veloso, M. 1999. Team Partitioned, Opaque Transition Reinforcement Learning. *Proceedings of the Third Annual Conference on Autonomous Agents* (pp. 206-212). ACM Press.