Evolving Effective Multi-Robot Coordination Strategies for Dynamic Environments Using Cultural Algorithms

Ayad Salhieh
School of Engineering
Australian College of Kuwait
Safat, Kuwait 13015
a.salhieh@ack.edu.kw

Mostafa Z. Ali, Ibrahim Mahmoud
Jordan University of Science & Tech.
Irbid, Jordan 22110
mzali@just.edu.jo
Ibrahim.just@gmail.com

Robert G. Reynolds
Computer Science
Wayne State University
Detroit, MI 48202 U.S.A
Robert.reynolds@wayne.edu

Abstract: - Simulated robotic soccer is frequently used as a test method for contemporary artificial intelligence research. It provides a real-time environment with complex dynamics and sensor information that is both noisy and limited. Team coordination between the robots is essential for success. Cultural Algorithm (CA) is a branch of evolutionary algorithms, and it is used in this research to teach software robots to play soccer by finding the best action to execute depending on its position on the field, and its relation to the nearest opponent. The action of each agent is encoded by an integer string that represents the action rules. Our agents played against a team of defenders from well-known teams in order to enhance their offensive capabilities. Agents developed good offensive abilities through team coordination processes supported by Cultural Algorithms. The simulation results are obtained using the well-known Robo-Cup soccer simulator. The results of this research suggest the effectiveness of the proposed method as well as indicating future research directions.

Keywords: - Cultural Algorithm, Evolutionary Algorithms, Knowledge Learning, Population Swarms, Robocup

1 Introduction
Teamwork, Advanced learning, and opponent modeling concepts are three of the most important issues necessary for building a competitive soccer robot team [1]. However, robot players can receive only limited vision and noisy data of the world-state due to the highly dynamic environment they operate within. At the same time, in order to make the right decision in a given scenario in such dynamic environment robot players will have to select actions from a very large search space of possible actions. In this research we assess the use of Cultural Algorithms [2] to generate a team to play in a simulated Robo-Cup Soccer League. The key to searching the environment space can be found in constraints that serve to coordinate agent behaviors. These constraints are derived from the rules of the game along with principles of collective interaction stored in the Belief Space of the Cultural Algorithm. These principles can be modified and refined as the system learns to develop a team of players.

2 Related Work
Previous work has investigated the use of several Evolutionary algorithms to simulate the evolution of cooperation within humans [2]. There, the extent to which concepts of information sharing and cooperation among a group of agents was studied.

Genetic Algorithms (GAs) have been employed on several occasions for use in the Robo-Cup domain. For example, in [3] a Genetic Algorithm was used to learn a team strategy that could defeat the strategy of a fixed opponent. An example rule is given below. For this rule, the playing area was split into 48 different sub-areas. The location of a player with the ball is represented as \( A \) being a value in the interval \([1 - 48]\). The distance between this player and an opposing player is represented as \( B \).

Depending upon the distance between the players, it can be deemed as near or not near (relative to a pre-defined threshold value). The result of \( A \) and \( B \) is \( C \) which is the chosen action made as a result of the information provided. \( C \) can be one of 10 different high-level actions which include a variety of pass, dribble or shoot options.

\[
\text{If Agent is } A_i \text{ and Opponent is in } B_j \text{ then } \text{Action is } C_k
\]

The team strategy is stored as a chromosome of 960 genes in length. As a player may be in 48 different positions on the playing field with an opposition player or players being either near or far. There are \( 48 \times 2 = 96 \) different action rules for each player and hence 960 for all 10 field players in total. The fitness of the strategy chromosome is calculated...
by playing in an actual match. The experiments concluded that even with a minimal population, results can be achieved using evolutionary techniques.

Other researchers [4] have applied GAs for learning team strategies; their technique was similar to that of [3] [5] [6]. In [7] [8] [9] a class named ‘Fitness’ which allowed the trainer to allocate points to a player type based on their performance (for example 1 point for scoring a goal) is used. An average score for each player type is used as the measure of fitness for a chromosome [10] [11].

In Fernandez et. al. [12][13][14] the authors investigated how policy reuse can be applied among several domains, as versions of a simulated RoboCup Keepaway problem, by using a mapping between their state and action space. Their work was a proof that Policy Reuse can be used as a mechanism for transferring learning in a way that outperforms the basic policy learner. In [15], [16], [2] Reynolds and Chung provided a framework to develop multi-agent cooperation among a group of robot soccer players. In that representation, plays can be learned without opposition, with passive opposition, or with active opposition. In their work, Cultural Algorithms (CA) required several hundred generations for each play to get effective results with evolution programming (EP) used to represent the population space.

3 Agent Framework

3.1 Agent Architecture

Once an agent connects to the RoboCup soccer server, it starts receiving a large amount of unformatted data every cycle and the agent has to parse this data in order to extract useful information about the game’s state. The extracted information must be converted into a suitable data representation that fits the agent's world model. The parser handles the interactions with the Soccer Server and converts server messages to C++ objects.

The received data is gradually abstracted to form a high-level model of the world, and the knowledge present at the highest level is then used to decide on the action to be performed by each agent. This action is then gradually translated into commands, which are executed at the lowest level. Agents must be capable of reasoning about the best possible action without losing too much time sending or receiving data. In order to deal with the timing constraints caused by the real-time nature of the domain, it is therefore desirable that the agents can perform these high-level reasoning processes independently from their interactions with the environment.

3.2 Implementation

We have used two different base teams, UvA and HELIOS as the starting point for our new teams. The benefit of using base teams as a starting point is that it saves us from having to deal with low-level issues such as parsing server messages and updating the world model. This way we can concentrate on our specific high level reasoning problems. We named our teams Cultured_UvA and Cultured_HELIOS, and for simplicity we will call our teams the Cultured teams.

As we mentioned previously, a coach can be connected to the server and it has complete information about the match in progress. The coach is capable of communicating with different team agents and providing them with the necessary information about the new strategies and actions that they have to consider in order to improve the overall level of team play. So we used the coach as a facility to communicate between the agents of the Cultured team and the Cultural Algorithm. Our Cultured teams are implemented using C++, while the Cultural Algorithm is implemented using JAVA.

In order to use Cultural Algorithms to solve the problem in the new domain, we needed to implement an interface (Soccer.java) and a new class (SoccerMatch.java). The class SoccerMatch is used to hold the values of the current soccer match, so it communicates with the coach of the team periodically, gets information about the current match, stores these values, and updates them. At the same time, it sends information from the Cultural Algorithm to the coach. While the interface to SoccerMatch is used to represent the problem domain, solutions length, parameters ranges, keeps checking problem constraints and evaluates the fitness value for each individual solution by extracting necessary values from the SoccerMatch class. Fig. 1 shows our modifications to the SoccerMatch class in order to implement the interface, where these parts are colored in gray.

In the Genetic Algorithm, a solution is represented as a concatenated values called a chromosome. Here, Cultural Algorithm encodes solutions as individual agents. Each individual agent (solution) consists of values for different parameters within pre-specified ranges.

The length of the individual agent description and the ranges of the parameters are problem dependent. Each individual must have a fitness value that is computed by applying the individual in the problem
domain in order to generate the fitness value. The work of the Cultural engine is to search for individuals with better fitness values (higher) and to try to generate new individuals that maximize the fitness by using the five different knowledge sources [2]. The fitness value can be calculated in many different ways and it depends on the problem that the Cultural Algorithm is trying to solve. In our research we used the match score as our fitness value.

![Fig. 1. Cultural Algorithm connected to the soccer match](image)

For the soccer domain we represented the solution as the different actions that an agent can take while being in a sub-area of the field or game context, so the more actions we have that our agent can select from the wider ranges we have for our parameters. While the length of the individual solution depends on the number of sub-areas to which we divided the field, the more sub-areas we have the longer solution length will be. **Fig. 2** shows a sample individual for the soccer domain.

![Fig. 2. Structure of a sample individual.](image)

An individual solution that is generated from the Cultural Algorithm engine is then translated into a concatenated string of values and transferred to the coach, through an instance of the **SoccerMatch** class. We synchronized the communication between the **SoccerMatch** class instance and the coach so that when each new solution is given to the agents in the current game, they can use it as their strategy for a pre-specified period of time after which the coach sends the fitness value for that solution back to the Cultural Algorithm. We implemented the communication between the C++ and Java sides in two different ways First we used UDP connection and then we write the necessary data to files and read from files. Once the coach receives the solution from the **SoccerMatch** side, it broadcasts it to all of the agents in the team except the goalie, since we used a fixed goalkeeper with intermediate skills.

Every agent in the cultured team receives the coach’s messages, but only the one that can kick the ball (called active player) at a given time uses the received data from the coach to decide what action to execute depending on his location on the field. We define a strategic position as the valid position for the player depending on the formation and position of the ball in the field. **Fig. 3** below shows the overall architecture for our Cultured team connected to the Cultural Algorithm.

### 3.3 Fitness

The main idea of evolutionary methods is to exploit the information about the play of those individual agents whose performance is above average. The choice of the fitness function is very important and it affects the evolution process. In this research we evaluate the performance of individual solution strategy in terms of the number of goals scored by the team in those matches in which the strategy controls the team.
4 Experiments

Several experiments have been performed using the framework described above. Two selected experiments are presented in this section. All experiments were performed under Linux Ubuntu 10.04 on a Dell laptop with a Core 2 Duo 2.2 GHz processor and 4 GB of memory.

In this experiment we used the Cultured_HELIOS team. HELIOS is a Japanese team that was the champion of RoboCup 2010, and they released a base source code for their team so other researchers can make use of it and develop their new team’s strategies. Cultured_HELIOS is able to model the soccer match environment accurately so that agents have high-level actions that they can perform. These high-level actions provide in its implementation a few actions. As an example, if the agent was dribbling with the ball and finds that he is able to score a goal directly; he will shoot the ball to the opponent's goal even if we did not select the action that scored a goal. Also, when the player has the ball and finds that an opponent can take the ball from him, he will clear the ball automatically.

Since Cultured_HELIOS can implicitly perform some of the needed actions according to the situation our team has two high-level actions for which it will evolve strategies. The first task is when to dribble with the ball, and the second is when to pass the ball to a teammate. When passing the ball to a teammate, it is not necessary to pass it to the closest; the agent will select the best teammate who can receive the ball.

Action selection depends on the sub-area over which the agent is located. Fig. 4 shows our field division. In the figure, area 3a and 3b are treated as one area (if an agent located on 3a it is the same as if it is located on 3b), the same goes for 4a and 4b. Areas 6, 5a and 5b are combined into one single sub-area (opponents' penalty area) in order to reduce the number of sub-areas. Individual length is 5 depending on the field division and the selection will be between two values: 1- Dribble with ball and 2- Pass ball to a teammate.

The first run for this experiment was made for two generations; the population size for each generation was ten. Our Cultured_HELIOS played against UvA base team. UvA Trilearn is the champion team in RoboCup 2003 world competition. They released source codes of the soccer team with the source code for high-level decision-making omitted. The strategy of UvA Trilearn agent is to kick the ball toward the opponent’s goal regardless of the position that the agent is in. Each individual solution was tested for three minutes before being given a fitness value.

We then performed a second run for five generations each with a population of size ten to determine whether improvements would continue to be made with increasing learning time. Much improvement has been made in five generations and more is expected as the number of generations used to learn is increased. Fig. 5 shows the best individuals of the first generation compared to the best individuals of the fifth generation.

Fig. 4. Soccer field divided into different areas

Fig. 5. Enhancement in individuals for the 1st and 5th generations for Cultured_HELIOS

Fig. 6 compares the best individuals of the second generation of the first run with the best individuals of the fifth generation of the second run. We have chosen individuals with highest fitness value from each generation and made them play against each other.

Fig. 6. Enhancement in individuals for the 2nd generation from 1st run compared to the 5th generation from 2nd run for Cultured_HELIOS
Table 1 shows the results, every result obtained by taking the average of 10 soccer matches. As expected the more opportunities to learn that a team had, the better its’ performance was. Also, we made the best individuals of the fifth generation from the second run to play against the best individuals of the second generation from the first run; Table 2 shows the average results of 10 soccer matches.

**Table 1. Performance of different generations for Cultured_HELIOS**

<table>
<thead>
<tr>
<th>Generation (old vs new)</th>
<th>Win</th>
<th>Lost</th>
<th>Draw</th>
<th>Avg. goals</th>
<th>Avg. goals lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 vs 1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>2.4</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2. Performance of the last generations of the 1st and the 2nd runs

<table>
<thead>
<tr>
<th>Generation (old vs new)</th>
<th>Win</th>
<th>Lost</th>
<th>Draw</th>
<th>Avg. goals</th>
<th>Avg. goals lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(of 2nd run) vs 2(of 1st run)</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>3.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Finally, our Cultured_HELIOS played against half a team consisting of defenders only. This experiment ran for five generations with a population size of ten. We omitted all of the attackers (players whose strategic position is in the front of the field and are capable to reach our goal) and kept the defenders (players whose strategic position is in the back of the field away from our goal). In this way our team agents will have higher probability of playing on the opponent's side of the field and will be able to develop good offensive strategies. However, this is not allowed in official soccer matches. Fig. 7 shows the best individuals of the first generation compared to the best individuals of the fifth generation.

**Fig. 7.** Enhancement in individuals for the 1st and 5th generations (offensive play).

Fig. 8 compares the best individuals of the fifth generation of the second run with the best individuals of the fifth generation of the offensive run.

**Fig. 8.** Enhancement in individuals of the 5th generation of 2nd run compared to the 5th generation of the offensive play.

The Best individuals of the third generation are selected to play against the best individuals of the fifth generation; Table 3 shows the average results of 10 soccer matches. Also, the best individuals of the fifth generation from run two are selected to play against the best individuals of the fifth generation from the offensive run; Table 4 shows the average results of 10 soccer matches.

**Table 3. Performance of 3rd vs 5th generations from offensive play**

<table>
<thead>
<tr>
<th>Generation (old vs. new)</th>
<th>Win</th>
<th>Lost</th>
<th>Draw</th>
<th>Avg. goals</th>
<th>Avg. goals lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(offensive) vs. 3(offensive)</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1.6</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**Table 4. Performance of 5th generation (offensive) vs 5th generation (run 2)**

<table>
<thead>
<tr>
<th>Generation (old vs new)</th>
<th>Win</th>
<th>Lost</th>
<th>Avg. goals</th>
<th>Avg. goals lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(offensive) vs 5(2nd run)</td>
<td>8</td>
<td>2</td>
<td>2.1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

5 Conclusion

In this paper we have introduced a new evolutionary algorithm to find the best actions that agents can perform in a specific sub-area on the soccer field. The best solutions generated from the Cultural Algorithm were effective when applied as hand-coded strategies for our team.

In the described experiments our Cultured_HELIOS team enhanced its scoring over the first, third and fifth generations as follows (8, 7.8 and 8.7 average goals per time slot) respectively. And for the offensive plays, enhancement was (8.3, 9.5 and 11.9 average goals per time slot) for the first, third and fifth generations respectively, so the enhancement between the normal plays compared to
the offensive plays for the fifth generations was (8.7 normal, 11.9 offensive).

The simulation results shows enhancement in individuals over generations in terms of goals scored. The players rapidly developed a strategy about what action to perform in a specific location. Over time the selected strategy is very efficient against opponents. Although much improvement has been made in five generations, more generations will be needed in order to completely remove the individuals with low fitness value.

References: