Paper Submission to Computer & Graphics Magazine

1. Paper Title: *Emergence of Multiagent Spatial Coordination Strategies through Artificial Coevolution*

2. Authors: André L. V. Coelho, Daniel Weingaertner, Ricardo R. Gudwin and Ivan L. M. Ricarte

3. Physical Address:
   Department of Computer Engineering and Industrial Automation (DCA)
   School of Electrical and Computer Engineering (FEEC)
   State University of Campinas (Unicamp)
   P.O. Box 6101
   Zip-code 13083-970
   Campinas-SP Brazil

4. E-mail: {coelho\(^1\), danielw, gudwin, ricarte}@dca.fee.unicamp.br

5. Phone and Fax Numbers: +55-19-37883771 | +55-19-32891395

\(^1\) Contact Author
Emergence of Multiagent Spatial Coordination Strategies through Artificial Coevolution

André L. V. Coelho, Daniel Weingaertner, Ricardo R. Gudwin and Ivan L. M. Ricarte

Department of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computer Engineering (FEEC)
State University of Campinas (Unicamp)

{coelho,danielw, gudwin, ricarte}@dca.fee.unicamp.br

Abstract

This paper describes research investigating the evolution of coordination strategies in robot soccer teams. Each player (viewed as an agent) is provided with a common set of skills and is assigned to perform over a delimited area inside a soccer field. The idea is to optimize the whole team behavior by means of a spatial coadaptation process in which new players are selected in such a way to comply with the already existing ones. The main results show that, through coevolution, we progressively create teams whose members act on complementary areas of the playing field, being capable of prevailing over a standard opponent team with a fixed formation.

Keywords

Multiagent teams, spatial coordination, artificial coevolution, emergence, simulated robot soccer.
1 Introduction

One of the most compelling and challenging tasks inside the Distributed Artificial Intelligence (DAI) field is that of suitably devising coordination protocols customized to the problems in mind. Coordination can be summarized as a property of a system of agents performing some activities in a shared environment, concerning with how to effectively orchestrate the group (inter-) actions, in time and space, for achieving coherence [8][24]. It usually incurs complexity, as there are no predefined general recipes indicating how to establish, \textit{a priori}, the rules of group behavior in view of all possible situations/scenarios. Moreover, there is a range of aspects, such as the homogeneity/heterogeneity of the agents’ skills or the environmental characteristics (static versus dynamic), that should also be regarded when one chooses the coordination mechanisms to be employed.

Soccer seems to be a rich testbed domain for the study of multiagent coordination issues. In such context, a set of players must work together in order to put the ball into the opposing goal (augmenting its score) while at the same time defending its own. This is a typical domain where cooperation [5] should take part in—the individuals have the same global objective. One important issue for a soccer team to win a game is the strategy it uses, during a game period, to place each of its components in a given region of the field (such as backfield, leftwing, attack etc). That is, how to delimit the zone at which a certain player can perform better in order to improve the capabilities of the whole group. This sort of coordination effort is referred to here as \textit{spatial strategy}. Our aim is to evolve this process of players formation through a co-evolutionary approach [7][13][14], so that a spatial strategy can emerge without human interference. We also want to qualitatively analyze how much the arrangement of a team can influence its overall performance.
In the sequel, we introduce the robot soccer problem and the artificial coevolution approach applied to multiagent spatial coordination, present our framework and solution, show the results from our experiments, and finally identify future plan of work.

2 Background and Related Work

Many approaches to tackle coordination problems are currently available in the literature [5][8][9]. Most of them center around the specification and implementation of high-level protocols (many times based on human social interactions) containing the actions to be taken at particular cases, either by a single agent or by the whole group. Well-known examples following such idea are Laird and others’ knowledge-based coordination model and Jenning’s formalism of commitments and conventions. In the latter, for instance, rules to undertake a specific course of action are conceived before the actual deployment of the team in the environment (joint commitments). In order to monitor whether these rules have been fulfilled or not or whether they are still valid in changing circumstances, there are also additional emergency instructions towards the dynamic adjustment of the group activities through time (social conventions).

Such kind of endeavor aiming at the conception of an explicit scheme of coordination seems to be only suited for a constrained class of problems, showing both performance and scalability bottlenecks when applied at more complex domains. Alternative mechanisms have been conceived in order to surpass those deficiencies, such as distributed planning and real-time (re-) planning [19][24].

A new line of research (followed in this work) involves the pervasive use of evolutionary techniques as a means to improve both individual as well as group
abilities in a concurrent manner. This methodology stipulates for group organization in a seamless and implicit manner; that is, there is no need for explicit pre-codified protocols. The coordination activities can now be viewed as an optimization problem whose solution(s) is (are) searched via a computational procedure that mimics the steps of the natural evolutionary process. Applying such strategy in the robot soccer scenario constitutes an innovative contribution of this work.

The CMUnited [20][21], developed at Carnegie Mellon University, has been one of the most successful physical robot soccer teams in the contests of the RoboCup world championship [17]. It encompasses a layered learning technique to first train the players basic skills (dribbling, shooting) for then building more complex capabilities (passing, positioning) upon the basic ones. The formation of the team can change in the course of the game, but the set of possible formations is determined empirically and one of them is chosen in accordance with the current situation of the match [19].

By other means, Balch [4] has used his robot soccer simulator [22] to investigate behavioral specialization in learning robot teams. In his work, all agents have a common set of skills from which they build a task achieving strategy using a Q-learning (reinforcement learning) algorithm. After playing for some time against a fixed strategy control team, the learning agents specialize into complementary roles because their reward depends on the score of the game, not on individual actions.

Some papers already report on work concerning the application of artificial learning and evolution to some soccer-related problems. For instance, the approach proposed by Agah et al. deals with the production of evolutionary cooperative strategies by means of a devised cognitive architecture based on Tropism [1], whereas Matsubara and colleagues employed a neural network approach for players on-line learning on how to take correct decisions (pass a ball to a peer or shoot towards the opposite goal)
according to some pre-established field positioning situations [12]. Andou has already assessed the employment of reinforcement learning schemes to update players positions on the field based on where the ball has previously been located [2]. By other means, Luke et al. set out to create a completely learned team of agents using genetic programming [10]. Their approach already employed an artificial coevolutionary methodology but, instead, it was conceived towards behavior-based team coordination, not coping with spatial organization problems.

The approach underlying this work differs from others in several aspects. First, we do not have any predefined formation for the players, but want that the formation emerges by means of an evolutionary scheme. We do not use reinforcement learning, but also apply the result of the game as a reward function for the employed evolutionary technique (in order to calculate the level of group adaptability), so that the performance of a single player depends on the performance of the whole team. Finally, as a more adequate strategy for the soccer players progressive spatial co-adaptation, a novel memory-based, cooperative coevolutionary architecture [14][16] towards the dynamic popup (emergence) of instances of evolutionary algorithms has been designed.

3 Simulated Robot Soccer

Research on robot soccer has received an increasing attention through the last years. Soccer is an attractive domain for multiagent study as the success of a team depends very much on some form of coordination [18][12]. It is also very appealing because the game is played in a dynamic, real-time, competitive and cooperative environment, from which the agents (players) percept only a small part (limited visual perspective), what typically incurs the need of world modeling, distributed learning and planing.
The control of the agents is decentralized and the changes in the environment neither are fully predictable nor happen in discrete time steps. But in our research, the game is simplified in a few aspects:

- Teams are composed of five players.
- The sidelines are walls—the ball bounces back instead of going out-of-bounds.
- After a scoring event, the ball is immediately placed back in the center of the field.
- Each player has accurate information about the position of the other peers and adversaries as well as of the ball.

According to Huhns and Stephens [24], there are two commonly used methods for apportioning tasks among cooperative agents. One is the functional distribution, in which cooperation comes as the union of the individual capacities of the players (one player is a good shooter, other is a passer, and so on). The second method, called spatial distribution, is a form of cooperation where the agents divide the search or performing space (in soccer, this could be the field) into well-defined areas, in such a way to quicken the team performance through the sharing of goal responsibilities. In this work, the latter method was chosen for experimenting with the coevolutionary coordination of robot soccer agents.

The Java-based soccer simulator used in this work [22] (Fig. 1) implements each player on a separate OS thread and runs the simulation in discrete steps. At each step, the robots process their sensor data before ascertaining their appropriate effector commands.

**FIGURE 1 SHOULD BE PLACED HERE.**
4 Coordination via Coevolution

Evolutionary Computation (EC) has come out as the branch of computational intelligence research employing metaphors from natural evolutionary phenomena as a means to achieve efficient problem solving (search) techniques [3]. Its applicability has constantly increased in recent years, and many are the engineering fields that have some of their processes improved through the appliance of evolutionary-based approaches. The majority of the implementations of such approaches descend from three independent lines of research, namely genetic algorithms (GA), evolutionary programming (EP), and evolution strategies (ES).

In the conventional GA model, a population of strings (chromosomes) codifying the possible solutions for the problem in hand passes through a cyclic (generation-based) process in which new candidates are constantly created and evaluated in accordance with some measure of environment adequacy known as fitness. Ancestors are charged by computational operators very much resembling natural evolutionary phenomena, such as reproduction, selection and mutation, being progressively replaced by more adapted newcomers. The population fitness tends to converge in the course of the process and (sub-) optimal solutions are obtained at final stages.

Some problems with this model have already been reported. First, it is very prone to the “local minima/maxima problem”, as it depends very much on the configuration (search space distribution) of the initial population. Likewise, some fast convergence problems may occur if the population size is not properly set. This model has, as well, scalability problems, like how to incorporate all the knowledge about the problem and to discriminate and prioritize (possibly several) distinct factors in a unique evaluation function. In the same manner, the representation of some heterogeneity issues behind the problem may be constrained, as the phenotypic interpretation of parameters is the
same for all the individuals (single species). Moreover, the model is also not adequate for the evolution of sets of interacting rules with variable sizes whose individual fitness are determined by their interactions via a simulated micro-economy. (Classifier systems and other related works, such as SAMUEL [15], have been devised to surpass such drawback.) Finally, it is not very suited for the representation/generation of complex structures such as those composed by many sub-entities (as it is the case of multi-agent coordination systems).

In order to tackle such deficiencies, distributed genetic algorithms [7] have been introduced. The idea is to bring about a set of genetic algorithm instances working together in a parallel/distributed environment in order to find out the best solution for a common problem. Each GA runs independently from the others. Other, more recently investigated, concepts are those of niches and speciation [11]. The first brings the idea of dynamically mounting small groups of correlated individuals that act upon a close region of a large search space. Individual niches compete for the allocation of trials. The second refers to new forms of “on-the-fly” species generation. Extending the boundaries, there is now such a trend to apply artificial coevolution [13] as a more suitable technique towards complexity overcoming. Artificial coevolution has its roots in its biological counterpart. Simply put, coevolution means “any reciprocal evolutionary change in interacting species [23].” Although vague, such definition is powerful enough to comprehend any natural process in which two or more species, typically coexisting in a same environment, have their evolutionary trajectories somehow affected by the stable ecological interactions and interrelationships their members jointly promote and take part in. In the artificial realm, two or more populations of different species are optimized together, one influencing the other by some means.
Artificial coevolution seems very suited for simulating cooperation and/or competition behavior among multiagent entities. Following such premise, Puppala et al. have devised a share-memory based approach [16] to evolve cooperating individuals of two different species—painters and whitewashers—for solving a room painting problem (see Fig. 2). In this case, each of the agents has unique abilities necessary to complete a job; that is, they are interdependent and the group behavior depends on the joint behavior of both components. The idea is to find pairs whose members are best adapted to each other, so the overall performance can be improved.

**FIGURE 2 SHOULD BE PLACED HERE.**

An individual from the first population (codified by a rule of behavior) is assessed by mating it with other individuals of the second population (the reverse is also true). Its highest performance evaluation on all pairs that it participates is assigned as its fitness. Instead of randomly picking the individuals, the authors conceived a buffer for grabbing and remembering the most successful pairs achieved so far: In this case, the mating is done by selecting the $N$ best partners from the other population which prevailed at the last generation. The memory is updated if a fitness value of a new assembled pair is higher than any of those currently stored, promoting the replacement of the stored pair with the minimum value (tail of the list) by the new one.

5 **Devised Approach for Soccer Team Coordination**

In this section, the features underlying our proposal for soccer team spatial coordination are gradually presented.
5.1. Players and Regions

Since our main interest was on the formation of the team and on its influence on the result of the game, all players, from both opposing teams, were modeled with the same basic skills and control algorithm (see Appendix A). Each player was allowed to perform only inside a particular actuation area, which was characterized by three mark points: defense (D), middle (M), and attack (A) (Fig. 3a). In order to avoid a player choosing a too small area to play, the field was separated into 18 squares, as shown in Fig. 3b. For classification purposes, we considered nine delimited regions covering the whole field (Fig. 3b). Each player of an experimental team was bestowed with a label indicating the region to which it belongs to, in accordance with the minimum Euclidean distance between the center of its actuation area and the center of all nine regions. Table 1 shows the coordinates of these regions.

FIGURE 3 SHOULD BE PLACED HERE.

TABLE 1 SHOULD BE PLACED HERE.

5.2. Teams

The investigation was conducted by engaging experimental teams against a fixed opponent control team that uses a 1:3:1 formation (Fig. 5a). The purpose was to evolve (or create) new teams that were able to defeat the control team in soccer contests, owing only to a different spatial distribution of the players. The motivation is to certify whether a different formation strategy has a direct influence on the relative performance of an evolved team versus the control team.
5.3. Architecture

Based on the concepts of niches, speciation and cooperative coevolution, we have designed a new architecture for multiagent spatial coordination. The most innovative idea is that of progressively assembling the evolving teams (niches) by allocating for each of the five possible players (positions) a promising acting region to be represented by a dynamically created species. In the beginning, all players are randomly selected from the same GA instance (called GA-0)—we could employ any other evolutionary algorithm as well. Then, in the course of generations, some players, competing against all, will prevail, spawning new offspring very akin to them. Those most adapted players and their offspring certainly will perform over similar field regions, characterizing a promising searching area for another GA instance (speciation). (This is why we partitioned the field into nine logical regions.) As times passes by, new GAs are popped up and the GA-0 is restarted with another initial population if the number of players of new species (may be more than one) fires up a certain threshold. Each new GA reserves a place in all future teams formation. That is, one of its members will be selected to take place in each experimental team. The coadaptation process is granted as the new GAs are formed by in accordance with the already existing ones. Fig. 4 shows the details of such strategy.

FIGURE 4 SHOULD BE PLACED HERE.

Some other considerations are worthy to be mentioned:
• This approach also guides the spring of new individuals in the GA-0 population that have complementary features from those of the already selected species.

• The new created GA instances will not promote evolution of its individuals until each of the five field positions has an associated species. This avoids the possibility of badly influencing the formation of the new GAs with corrupted (vitiated) initial populations of the GA-0.

• In order to give to its individuals a better chance to survive and to be selected for a new team, each new created GA instance can not have more than the half of the number of individuals in the GA-0. Only the most adapted are picked.

• The GA-0 population decreases as the number of species increases. This is because there will be less slots in a team the GA-0 individuals will struggle for.

• The architecture is also memory-based. However, in order to avoid combinatorial explosion problems (as we should have populations in the range of hundreds of elements), we did not adopt Puppala’s individual fitness evaluation based on cross-mating. Instead, we opted to apply random teams assemblage (limiting the number of possible teams) for those positions that do not already have an associated species.

For implementation purposes, six classes codified in Java mainly compose this architecture; they are Player, Team, Team_pool, Memory, Simulator, and GA, whose interrelationship model is presented in Fig. 6. Figure 7 brings an execution flowchart.

5.4. Genetic Algorithm

The creation and evolution of the players are controlled by a genetic algorithm that uses elitist selection, one-point crossover and mutation [3] to generate new players from a previous population. The initial population is randomly generated, in a uniform
distribution. The fitness function used to reward the players is not based on their single performance, but on the score of the team where the player actuates. Equation (1) brings such evaluation function, where $S_{\text{team}}$ and $S_{\text{control}}$ are the scores of an evolved team and the control group, respectively.

\[
f(\text{player}) = \begin{cases} 
3 + \frac{(S_{\text{team}} - S_{\text{control}})}{10}, & \text{for } S_{\text{team}} > S_{\text{control}} \\
-1 + \frac{(S_{\text{team}} - S_{\text{control}})}{10}, & \text{for } S_{\text{team}} < S_{\text{control}} \\
1 + \frac{S_{\text{team}}}{10}, & \text{for } S_{\text{team}} = S_{\text{control}} 
\end{cases}
\]  
(1)

It may happen that a player is chosen to play in more than one team. In this case it will keep the highest fitness of all teams it participated. In the case that a player does not play any match, it will receive the reward as being the average reward of all players on its region.

6 Simulation Results

Experiments were conducted by running the algorithm described in Appendix B and employing Balch’s Java-based soccer simulator for 50 players and 25 teams. The most important parameter settings can be found in Table 3.

For each GA-0 generation, all created teams played an 8-minutes long mach against the control team. Each simulation of a soccer match was performed on a separated Java thread and those simulations were the most time-consuming tasks, taking about 1 hour\(^2\) to simulate the 25 matches of an evolutionary cycle.

The evaluation of the results was focused on three criteria: spring of new formations, cooperative coevolution of the agents and convergence of the genetic algorithm.

\(^2\) The experiments were executed on an Enterprise 450 Machine with two 300-Mhz Ultra Sparc-2 processors, 512MB RAM, running SunOS 5.7 and using JDK 1.2 for code implementation.
6.1. Formations

Since the only difference between the emerged teams and the control team is the formation, we could verify that it played a prominent role in robotic soccer. The 10-best experimental teams were able to win the control team with an average of four goals of difference. It is worthy to remind that our approach created formations automatically, without human soccer expertise, and could be used to train different team formations according to its adversary. The formations of the control team and of three winning experimental teams can be seen in Fig. 5.

6.2. Coevolution

Looking at Figs. 5b, 5c and 5d, we can notice that the players actuation areas that emerged were complementary. Since each of the players came from a different GA instance, and since those GAs were created in different time steps, we can conclude that the coevolutionary approach was pivotal for the arrangement of teams whose players actuate cooperatively for the field covering. To assist the reader in this assessment, Table 2 brings the order in which the new GA instances were progressively created and the respective regions to which their initial population belongs.

| TABLE 2 SHOULD BE PLACED HERE. |

6.3. Convergence of GAs

Using the 50-players/25-teams configuration, we observed, through some experiments, that the GA-0 was very susceptible to quick convergence. Almost always, the new GA instances tended to emerge within the minimum necessary
number of generations (Table 3), and their initial population was formed by only one or two different classes of players. This was typically a non-diversity problem. To avoid this kind of behavior, we increased the number of players and teams to 500/250, and observed that the population was more diverse and, thus, did not converge so fast. However, such decision incurred, as a side effect, unaffordable simulation cost increases, in such a manner to hamper the performance assessment process.

TABLE 3 SHOULD BE PLACED HERE.

7 Conclusion and Future Work

Applying coevolutionary techniques in complex problem domains has been proved to be a promising alternative strategy for achieving both performance and quality improvements. Recent research works have addressed the employment of such approach in a variety of problems, including single/multicriteria function optimization [6][7][14] and multiagent scenarios [15][16][10]. In this work, a new coevolutionary-based architecture for robot soccer teams spatial coordination was depicted and evaluated, confirming: (i) the feasibility of obtaining an automatic method for the generation of implicit coordination rules; (ii) that the spatial distribution of homogeneous players across the field can direct influence the behavior and performance of the whole team; and (iii) that the approach encourages the formation of stable niches of cooperative subcomponents (players) whose acting regions tend to be complementary on the field covering.

Some problems were detected during the simulations execution, demanding for new design or parameter setting corrections. For instance, the fast convergence in the fitness of the new created GAs’ populations surely had a great bad effect on some attained results. Increasing the size of all GAs, however, would complicate the players
evaluation process and augment the computational time required at each running cycle. As a consequence, there is an intrinsic hard trade-off for configuring parameters of this sort, as well as for those relating to the GA operators (higher mutation rates shall also provide a means to overcome such fast convergence).

Another problem that deserves attention relates to the possibility of loosing the constituents of the best existing teams (those already in the memory). We did not provide a means to maintain these players in their respective GAs’ populations, that is, a means to also memorize them. As a consequence, some very successful teams that emerged during the evolutionary process were lost.

The formations that emerged during our tests are suitable only to play against the control team used in the experiments, performing badly against teams with different formations. This is a big limitation if the team is intended to participate in a competition like the RoboCup. Therefore, an interesting extension to this work would be to train many different formations against distinct configurations of control teams, store the best formations in a run-time memory and then, during the contests, dynamically adapt the team spatial distribution in conformance with the opponent’s strategy.

FIGURE 5 SHOULD BE PLACED HERE.

FIGURE 6 SHOULD BE PLACED HERE.

FIGURE 7 SHOULD BE PLACED HERE.
Acknowledgements

CNPq and Capes have sponsored the first and second authors through Ph.D. and MSc. scholarships.

Appendix A

Algorithm that controls the actions of each player:

```plaintext
Compute: (A)attack, (M)middle, (D)defense

if (Player outside it's area )
    Move to area;
else if (Ball inside area )
    if( Closest player to ball )
    Move to ball;
    Else
    Move close to ball;
else // ball outside the area
    if( Ball on defense side of field )
    Move to (D);
    else if( Ball on attack field )
    Move to (A);
    Else
    Move to (M);
if( Can kick and Is worth kicking )
    Kick the ball;
```
Appendix B

Algorithm for the creation of teams through coevolution:

1. Create GA-0’s initial population;
2. Classify players in regions;
3. Create N teams;
4. Simulation:
   for each team 
   Play against static team;
   Compute fitness;
   if( $S_{us} > S_{them}$ )
     fitness $\leftarrow 3 + (S_{team} - S_{control})/10$;
   else if( $S_{team} < S_{control}$ )
     fitness $\leftarrow -1 + (S_{team} - S_{control})/10$;
   else
     fitness $\leftarrow 1 + S_{team}/10$;
   (where $S_{team}$ and $S_{control}$ is the score of each team at the end of the game)
   Set fitness of the players;
   Save M best teams;
   for each region
   Compute fitness of the regions (Fr);
   Players without fitness $\leftarrow Fr$;

5. for( G generations )
   for( GA-0 xor GA-i )
     Elitist selection;
     Crossover;
     Mutation;
     Classify players in regions;
     Create N teams;
     Take one player from each GA-i;
     Complete team with players from GA-0;
     Simulate the games;
     if( $|GA| < |players per team| and G > min$ )
       if( $|region| > Threshold$ )
         Create new GA-i;
         Initialize it with population of region;
         GA-0 $\leftarrow$ New initial population;
       else if( timeout )
         New GA-i $\leftarrow$ Biggest region;
         GA-0 $\leftarrow$ New initial population;

6. Print best teams
References


Figure 3

(a)

(b)
Figure 4

[Diagram showing a flow of teams and genetic algorithms (GA) over time, with arrows indicating the direction of interaction.]
Figure 5

<table>
<thead>
<tr>
<th>Player Nr.</th>
<th>Coordinates</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0,0) (2,3)</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>(2,0) (5,1)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>(2,4) (4,3)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>(2,2) (5,3)</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>(4,0) (6,3)</td>
<td>6</td>
</tr>
</tbody>
</table>

(a): Contact team

<table>
<thead>
<tr>
<th>Player Nr.</th>
<th>Coordinates</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.0) (4,2)</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>(0.1) (5,3)</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>(1.1) (4,5)</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>(3.0) (5,2)</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>(4.0) (6,5)</td>
<td>5</td>
</tr>
</tbody>
</table>

(b): $S_{	ext{new}} = S_{	ext{old}}$ = 6

<table>
<thead>
<tr>
<th>Player Nr.</th>
<th>Coordinates</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0,0) (1,4)</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>(0,0) (6,1)</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>(1,1) (6,3)</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>(2,0) (5,2)</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>(5,0) (6,3)</td>
<td>3</td>
</tr>
</tbody>
</table>

(c): $S_{	ext{new}} - S_{	ext{center}}$ = 5

<table>
<thead>
<tr>
<th>Player Nr.</th>
<th>Coordinates</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0,2) (1,3)</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>(1,0) (6,1)</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>(1,0) (5,2)</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>(0,1) (5,3)</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>(5,0) (6,2)</td>
<td>5</td>
</tr>
</tbody>
</table>

(d): $S_{	ext{new}} - S_{	ext{center}}$ = 4
Figure 6
Init GA-0

Run GA-0 Cycle

New region?

Yes

Create New GA

Store

No

Mount teams

Any team in memo?

Yes/No

Run simulation

Yes

Show results

No

Finish
**Figure 1:** Game start position on the Java-based robot soccer simulator.

**Figure 2:** Shared-memory based approach for multiagent coordination.

**Figure 3:** (a) Actuation area of a player with its defense (D), middle (M) and attack (A) position. (b) Discretization of the field (6x3 rectangles). Each player is associated to the closest region, from nine defined on the field, according to the distance measured between the center of the actuation area to the center of the region.

**Figure 4:** Creation of the teams: On the beginning, all players are taken from the GA-0. As soon as a region of the field has enough players to form a new population, this population is copied into a new GA-1. After that, each team will have one player coming from that GA. The GA-0 is restarted and the procedure is repeated until we have one GA for each position of the team. Then the GA-0 is destroyed and the others are activated to co-evolve.

**Figure 5:** Team formations. (a) Formation of the control team. (b), (c) and (d) are evolved formations that were able to defeat the control team with a goal difference of 6, 5 and 4 respectively. We verify that the players occupy complementary and overlapping parts of the field, in a way that almost all the field is covered by the team.

**Figure 6:** Object model showing the relationships among the main Java classes. $N$, $T$, and $M$ are parameters indicating the maximum number of GA instances (5), the number of experimental teams per generation (25) and the number of buffered teams (10).

**Figure 7:** Flow of execution in a typical run of the proposed coevolutionary approach.
<table>
<thead>
<tr>
<th>Region</th>
<th>Coordinates</th>
<th>Region</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>(1.5, 0.75)</td>
<td>r6</td>
<td>(5.0, 1.5)</td>
</tr>
<tr>
<td>r2</td>
<td>(3.0, 0.75)</td>
<td>r7</td>
<td>(1.5, 2.25)</td>
</tr>
<tr>
<td>r3</td>
<td>(4.5, 0.75)</td>
<td>r8</td>
<td>(3.0, 2.25)</td>
</tr>
<tr>
<td>r4</td>
<td>(1.0, 1.5)</td>
<td>r9</td>
<td>(4.5, 2.25)</td>
</tr>
<tr>
<td>r5</td>
<td>(3.0, 1.5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 1:** Coordinates of the region points used to classify the players according to their acting area
<table>
<thead>
<tr>
<th></th>
<th>GA-1</th>
<th>GA-2</th>
<th>GA-3</th>
<th>GA-4</th>
<th>GA-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r2</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>•</td>
</tr>
<tr>
<td>r6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r7</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>r8</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2:** Order in which the new GA instances were created and the regions to which their initial population belonged
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>POP_LENGTH</td>
<td>Number of players in the initial GA-0</td>
<td>50</td>
</tr>
<tr>
<td>MEM_SIZE</td>
<td>Number of best teams that are kept on memory</td>
<td>10</td>
</tr>
<tr>
<td>TEAMS</td>
<td>Number of teams to be formed</td>
<td>25</td>
</tr>
<tr>
<td>PLAYERS</td>
<td>Number of players per team</td>
<td>5</td>
</tr>
<tr>
<td>MAX_NEW_GA_POP_LENGTH</td>
<td>Maximum number of players that a new GA may have</td>
<td>TEAMS / 2</td>
</tr>
<tr>
<td>TIME_TO_LIVE</td>
<td>Maximum number of generations</td>
<td>120</td>
</tr>
<tr>
<td>THRESHOLD</td>
<td>Percentage of players a region must have to form a new GA</td>
<td>40%</td>
</tr>
<tr>
<td>MIN_TIME_TO_POPUP</td>
<td>Minimum number of generations for a new GA to be formed</td>
<td>5</td>
</tr>
<tr>
<td>MAX_TIME_TO_POPUP</td>
<td>Maximum number of generations for a new GA to be formed</td>
<td>20</td>
</tr>
<tr>
<td>GENE_CROSSOVER_CHANCE</td>
<td>Chance of crossover occurrence</td>
<td>25%</td>
</tr>
<tr>
<td>GENE_MUTATION_CHANCE</td>
<td>Chance of mutation occurrence</td>
<td>1%</td>
</tr>
<tr>
<td>ELITIST</td>
<td>Number of best players copied to next generation (elitist selection)</td>
<td>20%</td>
</tr>
</tbody>
</table>

**Table 3:** System configuration parameters