Soft Computing (2019) 23:7287–7302
https://doi.org/10.1007/s00500-018-3376-6

METHODOLOGIES AND APPLICATION

A framework based on evolutionary algorithm for strategy optimization in robot soccer

Asma Larik[1] · Sajjad Haider[1]

Published online: 12 July 2018

Abstract
In any competitive and uncertain environment, designing an optimal strategy is a challenging task. The manual hand-coding of strategy is a tedious job, and its evaluation on all possible situations becomes even more complicated. This paper proposes a novel distributed framework, named FEASO, based on evolutionary algorithms, for strategy optimization in the domain of robot soccer. In the context of robot soccer, strategy denotes the critical areas where home team agents should be positioned. The focus of this study is to optimize the strategic placements of agents that are defending the goal. The presented approach comprises three modules: evolutionary algorithm execution, parallel fitness evaluation and fitness computation. It executes matches in parallel on different machines for fitness evaluation. The fitness function takes into account three parameters: the goal difference, regions occupied by defending players and ball possession by the home team players. The framework has been successfully implemented in our 3D soccer simulation team that participates in RoboCup event. Experiments are conducted using binaries of various teams taking part in the competition. A comparison of strategies between teams is conducted and analyzed. The results clearly demonstrate that the team that executes optimized strategy is able to defend more goals as compared to the team with hand-coded strategic points.

Keywords Opponent · Strategy optimization · Autonomous robots · Evolutionary algorithms · RoboCup Soccer Simulation 3D

1 Introduction

Strategy development for a team of agents/robots, with decentralized decision making, is a challenging task in an uncertain and adversarial domain. An agent playing soccer not only has to score a goal against an opponent team but also has to defend its own goal. Team strategy, thus, becomes a collective outcome of individual agents' strategies. To evaluate a team strategy, a number of variables have to be considered. For instance, if a team is on defense, the list of variables may include alignment of defenders with the goal post, distance maintained between the ball and the defenders, the alignment of players in a supporting role with the ball and the distance between the opponents. In contrast, when a team is on the attack, then the number of players involved in the attack, their attacking pattern of moving ahead or staying behind at a certain distance becomes important for devising a strategy. In situations like these, it becomes difficult for an expert to design a strategy that works well in every scenario. In addition, manual evaluation of the strategy on all possible scenarios becomes even more time-consuming. To overcome these challenges, this paper presents a parallel evolutionary computation-based framework for producing an optimized strategy. Parallelization is introduced to reduce the time taken by a single machine in evaluating all possible strategies. Offspring in the evolutionary algorithm is produced by a server, while the workload is distributed among the clients for strategy evaluation.

It is important to note that in the context of this paper a strategy denotes the critical areas that should be occupied by defending players. The alignment of players with the goal and the ball has been evolved to obtain optimized strategic points. These optimized points are then executed in real time

Communicated by V. Loia.

✉ Asma Larik
asma.sanam@khi.iba.edu.pk

✉ Sajjad Haider
sahaider@iba.edu.pk

[1] Artificial Intelligence Lab, Faculty of Computer Science, Institute of Business Administration, Garden Road, Karachi 74400, Pakistan

for team performance evaluation. The fitness of a solution takes into consideration the difference between goals of the home and the opponent teams, the average ball possession by the home team and certain regions occupied by home team agents.

For the experimental purposes, RoboCup Soccer Simulation Server 3D has been chosen as a test bed. The environment provides a promising platform which is truly dynamic and partially observable in nature. A RoboCup Soccer Simulation 3D League team (Darab and Ebrahimi 2007), namely Karachi Koalas (Haider et al. 2014), that was developed as a part of a larger project has been taken as the home team in this research. This team has been frequently participating in the RoboCup competitions with the best ranking of being the fifth ranked team in the year 2013. The base code of this team has been taken as a benchmark as matches of the base code team, and the evolved team are executed against a similar opponent. It is also worth mentioning that much work has been done in the literature in which other techniques rather than the evolutionary approaches have been applied such as reinforcement learning (Salustowicz et al. 1998; Riedmiller et al. 2009), handcrafted structuring of code (Kazakov and Kudenko 2001), layered learning (Stone and Veloso 2000; Cherubini et al. 2007; Urieli et al. 2011). In this paper, we have compared other approaches with the evolved approach by executing matches to demonstrate that the evolved strategy works well. The rest of the paper is organized as follows. Section 2 provides an overview of the existing literature along with a subsection presenting novelty of FEASO. Section 3 briefly describes the technical background of the RoboCup Soccer domain as well as evolutionary algorithms, and Sect. 4 gives a comprehensive overview of FEASO with its implementation details. Experimental setup and results are discussed in Sect. 5. Section 6 compares FEASO with most relevant evolutionary approaches proposed previously, while Sect. 7 concludes the paper and provides future research directions.

2 Related work

There are various techniques reported in the literature to incorporate strategies in robot soccer. The list of techniques includes reinforcement learning, decision trees; handcrafted rule-based techniques, coordination graphs, layered learning approaches, evolutionary algorithms. The notion of strategy varies among each of the specified techniques. This literature review focuses on learning team strategy and thus techniques that employ strategy optimization are discussed in detail in each of the subsections.

2.1 Reinforcement learning techniques

In reinforcement learning, strategy denotes learning a good policy for a team of agents and concept of collective reward and punishment was introduced. The earliest approach was proposed by Salustowicz et al. (1998) who compared several reinforcement learning algorithms in multi-agent learning scenarios. Riedmiller et al. (2009) suggested learning strategy at a tactical level, such as moves to intercept the ball, wait at a position, passing the ball to a teammate, shoot to goal. Many variants of reinforcement learning such as observational reinforcement learning, team partitioned opaque transition reinforcement learning, Q-learning have been proposed in the domain of RoboCup soccer simulation 2D, but in 3D there are many challenges in current reinforcement learning research. Firstly, with the addition of each new agent, it becomes memory expensive. Secondly, similar behaviors are exhibited in robot soccer and the problem of learning everything all over again exists. Lastly, due to limited perception of an agent, it is often impossible to fully determine the current state and this result in loss of performance of the algorithm. As this study considers an optimized strategy for a team of agents, reinforcement learning has not been used.

2.2 Rule-based techniques

In contrast to reinforcement learning, rule-based and inductive logic programming-based approaches (ILP) have also been proposed by different researchers. The inductive learning agent (Kazakov and Kudenko 2001) uses first-order formalism and ILP to acquire rules to predict failures. The agent gathers instance of actions and classifies them. Next prediction rules are formed that assist soccer agents in deciding each action taken in the game to be good or bad. The limitations of this approach are that its focus lies on verification and validation of knowledge-based system with no new knowledge acquisition. Consequently, agents cannot adapt their own behavior using rules or knowledge acquired by ILP. Murray et al. (2000) developed a multi-agent team based on script language, where procedural aspects were specified by state charts and declarative aspects by using decision trees. The rules have to be designed by an expert and it becomes a tedious job to think of all sets of possibilities. To overcome this limitation, the study (Kok et al. 2005) presented a strategy that was devised by using a few rules provided by experts and the rest of the coordination rules were computed online using coordination graphs. Bezek (2005) and Wang et al. (2009) proposed the use of coordination graphs in the assignment of roles and actions to the players. Svatoň et al. (2014) proposed a methodology to improve the description of strategy by creating sub-strategies and thus ensuring a smooth implementation of actions defined by each strategy. The main focus of coordination graphs is to aid humans in strategy building. A human modeler analyzes the various patterns in the graph and designs hand-coded agents to act under certain circumstances. The focus of this study is not on rules or graph creation but the generation of an optimized strategy that works well in an automated fashion. Thus, no handcrafting of rules is performed but the evolution process applied has the advantage of giving multiple strategies, which are equally good in a specific situation.

2.3 Layered learning technique

Layered learning has also been applied for flexible team formation in RoboCup Soccer Simulation Leagues. Stone et al. (2000) pioneered the architecture in which the problem consisted of breaking into a bottom-up the hierarchy of subproblems. The subproblems are solved and they serve as input to the next layer. The problem used Genetic Programming for selection of an action by a team of agents. Urieli et al. (2011) presented an approach for optimizing interdependent skills in the 3D domain. This layered architecture has the benefit that primitive locomotive skills can be learned precisely and multiple skills are learned in conjunction with each other. The issue lies in learning strategy because it is a high-level behavior exhibited by a team of agents that requires numerous learning efforts if performed via layered learning approach.

2.4 Evolutionary algorithm-based techniques

Evolutionary algorithms have also been applied to evolve strategies in robot soccer. Nakashima et al. (2004, 2006) proposed the use of Genetic Algorithms to learn a team strategy in the domain of 2D simulation league. They divided the field into 48 subregions and the 10 players of both the teams excluding the goalkeeper could exist in any of the regions. The chromosome comprised of the action to be executed in a certain situation. The fitness of a strategy is computed by running simulated matches and taking average goal difference. One limitation of the approach was that a single opponent was fixed and strategy would not be generalized. Cultural Algorithms (CA) was applied (Salhieh et al. 2012) to generate a team strategy for playing robot soccer. A chromosome contained regions and the corresponding player numbers. This was a centralized approach where a coach agent was developed that sent messages to all the agents and only the intended agent executed the action. Goal difference was used as the evaluation function. Recent advances in team strategy were contributed by Ali et al. (2014) in which they investigated the use of a simplified and adaptive version of CA to develop defensive and offensive plays and cooperative strategies in robot soccer. This work was very relevant to the presented approach, and a detailed comparative discussion is provided later in Sect. 6. Lekavy (2011) applied evolutionary approach for evolution of pass execution in standard situations such as kickoff in 2D simulation league. Okada et al. (2011) utilized particle swarm optimization to evolve team formation. Cartesian coordinates of the ten players and 15 possible positions of the ball were modeled as elements of an individual solution, while the initial population was generated randomly. They reported how well formations for various team performances (e.g., offensive, defensive, balanced) could automatically be obtained. Ant Intelligence (Ramani et al. 2008) was utilized for player strategies generation. Multi-group ant colony optimization algorithm (Chen et al. 2016) was proposed that used ant pheromone evaporation mechanism to learn offensive strategies in 2D soccer simulation league. A comparative analysis of this approach is presented in Sect. 6. Luh et al. (2006) used the immune system to develop cooperative strategies in robot soccer. The system selected a behavior for the player such as shot, pass, kick, chase, track, and guard. The approach was validated on the SimuroSot Middle League. It is important to note that all of the above-mentioned techniques using evolutionary computation face the challenge of the time needed to evolve a solution as a large number of fitness evaluations are needed before a well acceptable solution can be found. In light of this requirement, distributed evolutionary approaches are proposed (Gong et al. 2015) and this research also contributes to a distributed approach.

2.5 Novelty

The FEASO framework is a distributed approach for evolution of defensive strategy that has not been addressed in other techniques proposed in the literature. The major contribution of this framework is that it facilitates automatic generation of optimized strategy against a specific opponent team and then generalizes it across various similar opponents. This paper focuses upon the strategic placements of all the agents in the field, while most of the above-mentioned evolutionary techniques consider either subfield or two-three players in the field to demonstrate the required behavior. Moreover, the domain of RoboCup Soccer Simulation 2D in which the above-mentioned approaches were tested is less complex and knowledge about the action to be executed is available to the players that make it easier to exhibit a cooperative strategy. But the domain of RoboCup Soccer Simulation 3D is highly dynamic and unpredictable in which only positions of the agents are known and actions have to be inferred by observational data. The author finds no technique for strategy evolution in the domain of RoboCup 3D. The proposed framework is both robust and time inexpensive in real-time strategy execution. This framework can also be applied to similar competitive domains as well.

Fig. 1 Two teams playing RoboCup Soccer Simulation 3D

3 Technical background

This section presents an overview of the domain of RoboCup and also discusses some basics of evolutionary algorithms.

3.1 Domain description

RoboCup Soccer (Kitano et al. 1998) is a scientific venture that provides an exciting platform that has been used for the advancement of research in artificial intelligence and robotics. The competition has a goal that, by the middle of the twenty-first century, a team of robots would defeat a team of humans in soccer. This competition is held every year, and several leagues have been designed to cater for different problems and challenges in robot coordination, locomotion, etc. The main leagues in this tournament are a middle-sized league, small-sized league, simulation 2D league, simulation 3D league, standard platform league and the humanoid league. In contrast to other leagues, the simulated leagues do not involve physical robots and instead simulated robots play soccer on the virtual field. In the simulation 2D league, the ball and the players are represented by circles on the plane of the field, while in the simulation 3D League, the players are represented as articulated, rigid bodies having 22 hinges. In the 2D league, commands such as move, dash, turn and kick are available; however, in 3D these commands do not exist and locomotion is a big challenge there. The 3D game comprises of two teams of eleven robots in a field of 21 * 30 m as shown in Fig. 1.

This research utilizes RoboCup Soccer Simulation 3D as a test environment for the development of opponent-specific strategies. The match is divided into two halves of 5 min each. Continuous time is approximated with discrete cycles with each half being 300 cycles and 1 min equals to 60 cycles. Each robot has its own local view of the field and is controlled by a separate autonomous program. All robots can move and act independently as long as they comply with the league rules. They can freely communicate with each other via message broadcasting, but their visual and hearing perception is distance limited. Due to humanoid locomotion, there are chances of robots falling in the field or colliding with other robots so the server, in such cases, can penalize them by throwing them out of the field. The domain is a complex multi-agent system and is easily understandable by humans due to its soccer-related content.

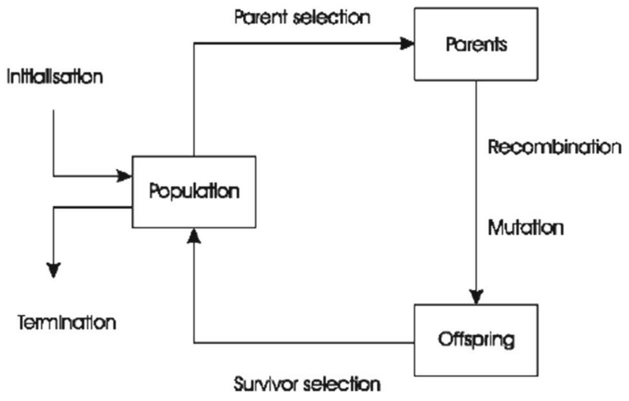

Fig. 2 Typical evolutionary algorithm. (Reproduced with permission from Bäck 1996)

3.2 Evolutionary algorithms

In many real-world applications, there exist problems that are hard to solve such as traveling salesman problem, time series prediction, bankruptcy prediction, credit scoring. Traditional algorithms to solve these are either very specialized or more general. To approach such hard problems, evolutionary algorithms (EA) were introduced a few decades ago. Evolutionary Algorithms (Bäck 1996) are stochastic search and optimization heuristics derived from classical evolution theory. It follows the Darwinist evolution, which is described as survival of the fittest. EA methods only need the target (fitness) function for a given problem, which is to be optimized.

A typical evolutionary algorithm passes through the following phases as shown in Fig. 2. At first, the population is initialized randomly. The fitness of each solution is computed, parents are selected via selection procedure and offspring are created by variation operators. Next fitness of the new offspring is computed. Members of the population die using survival selection and the process continues until a termination criterion is met.

3.2.1 Parent selection mechanism

Parent selection is used to distinguish among individuals based on their quality to allow better individuals to become parents of the next generation. There are various selection schemas (Bäck 1996), namely fitness proportional selection (FPS), rank-based selection (RBS), tournament selection (TS). In FPS, a probability distribution proportional to the fitness is computed and individuals are selected by sampling the distribution. This schema has high selection pressure.

Rank-based selection attempts to remove problems of FPS by computing selection probabilities on relative fitness rather than absolute fitness. RBS mechanism ranks population according to fitness and selection probabilities on the basis of the assigned rank. This schema imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time. The advantage of this schema is that the selection is independent of actual fitness values and it preserves a constant selection pressure.

3.2.2 Variation operators

The role of variation operators is to create new individuals from the old ones. There are two types of variation operators: mutation and crossover. Mutation is a unary variation operator and is always stochastic. Crossover is a binary variation operator that merges information from two parents to generate two offspring. The two fundamental design concepts in evolutionary algorithms are exploration and exploitation. Exploration means discovering promising areas in the entire search space, while exploitation deals with optimizing within a promising area. Crossover is explorative as it makes a big jump to an area between the parents. Mutation is exploitative as it creates random diversions while standing near the parents.

3.2.3 Survival selection mechanism

The role of survivor selection is to distinguish among individuals based on their quality. A survival selection scheme can be categorized as generational (current population replaced by its offspring) or steady-state model (few members move are replaced by its offspring). Truncation is a survival selection schema in which out of a **2n** population of parents and offspring, the top **n** survive to the next generation.

3.2.4 Performance indicators

Following are some of the performance indicators for evolutionary algorithms:

- Best-so-far (BSF): The best solution found by the algorithm in each generation.
- Average-so-far (ASF): the average solution found by the algorithm in each generation.

4 A framework based on evolutionary algorithm for strategy optimization (FEASO)

This section provides a comprehensive overview of the entire system architecture of the proposed framework. The framework focuses on devising an optimized strategy in robot soccer by discovering the key points where the agents should place themselves under various scenarios.

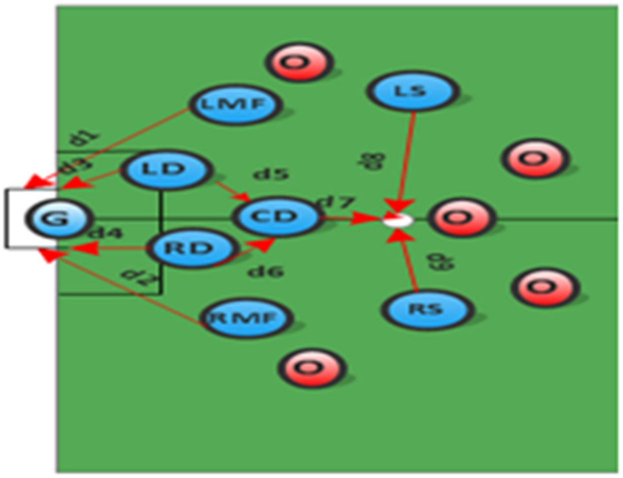

Fig. 3 Simulated soccer field with arrows denoting optimization variables (color figure online)

4.1 Optimization parameters

To develop an understanding of the parameters being optimized, an example from robot soccer is presented. Figure 3 depicts an instance of a simulated soccer field in which agents of both the teams (red and blue) are placed. The red circles with 'O' represent the opposing team, while the blue circles represent the home team. A small white circle denotes the ball. The scenario shown is an instance of attack being executed by the opponent team. There are different roles associated with the home team players such as goalkeeper (G), main attacker (MA), left forward (LF), right forward (RF), left supporter (LS), right supporter (RS), left midfielder (LM), right midfielder (RM) and finally three defenders left defender (LD), right defender (RD) and center defender (CD). Each of these roles occupies certain positions in the field. These positions are in turn determined by factors such as the distance of a player from the ball, distance from the goal posts and distance with other teammates. The red arrows highlight the variables ($d1$, $d2$, …., $d9$) that needs to be optimized. Table 1 signifies the relationship of these variables with the roles.

The variables $d1$, $d2$, $d3$ and $d4$ represent the distances of LM, RMF, the x coordinate of LD and x coordinate of RD with respect to the goal post. The variables $d5$ and $d6$ represent the distances maintained between the CD player and y coordinates of the left and right defense players. CD also maintains a distance $d7$ with the ball. The variables $d8$ and $d9$ denote the distances between the ball and the left and

Table 1 Solution Representation

LM	RM	LD (x)	RD (x)	LD (y)	RD (y)	CD	LS	RS
$d1$	$d2$	$d3$	$d4$	$d5$	$d6$	$d7$	$d8$	$d9$

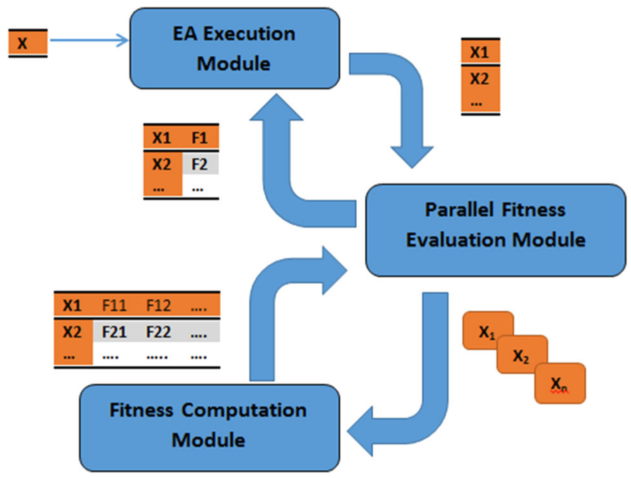

Fig. 4 Proposed modules of FEASO

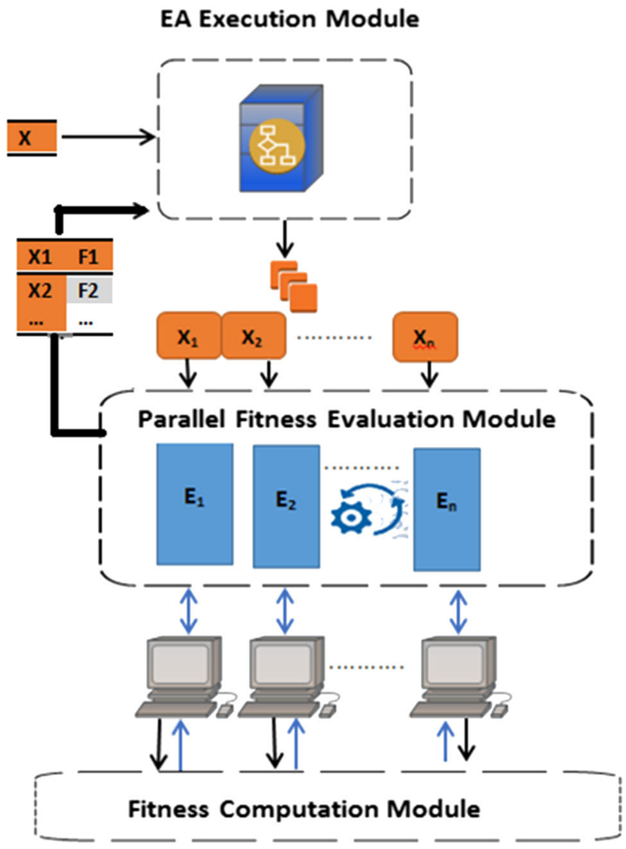
EA Execution Module
X
X1 F1
X2 F2
Parallel Fitness Evaluation Module
E1
E2
En
Fitness Computation Module

Fig. 5 System architecture of FEASO

right supporters. These nine variables form a single tuple X_i. Attackers, goalkeepers and forward players are not considered in this representation.

The initial value for X_i that forms the seed is populated from the code of our home team. The rationale behind the selection of this seed is that it is a viable solution already available. If X_i is initialized randomly, then it is possible that we get distances that are not feasible for a player according to its role.

4.2 System architecture

FEASO comprises of three basic modules, namely evolutionary algorithm Execution Module (EAEM), Parallel Fitness Evaluation Module (PFEM) and Fitness Computation Module (FCM) as shown in Fig. 4.

The input to EAEM is a single tuple $\mathbf{X_i}$ and its output is a set of newly generated variables formed by application of an evolutionary algorithm. These variables are then passed to the PFEM module which replicates each one turn by turn on multiple machines. Each of the machines then executes FCM and sends a fitness value back to PFEM. The PFEM applies a more complex function on all fitness values to produce the final fitness that is returned to the EAEM and the cycle continues. Figure 5 gives the high-level system architecture depicting all the modules.

4.2.1 Module I: evolutionary algorithm execution module (EAEM)

This module is a vital part of the entire architecture as it runs on the server machine and is responsible for the generation of offspring. The process starts with the generation of an initial population represented by P. It contains a set of $\mathbf{X}_i$ that in turn are produced randomly from Gaussian random mutation of X_{seed} value. The fitness computation is performed by PFEM and FCM as shown in the gray box in Fig. 6. The newly generated population is then sent to the parent selection phase that selects a set of parents using Rank-Based Selection Scheme. Crossover and mutation of the parents are performed and the resultant population along with new offspring is sent to the survival selection phase. The proposed algorithm introduces a random offspring generation phase for increasing diversity in the population. This phase makes the algorithm exploratory in nature and different from the traditional EA.

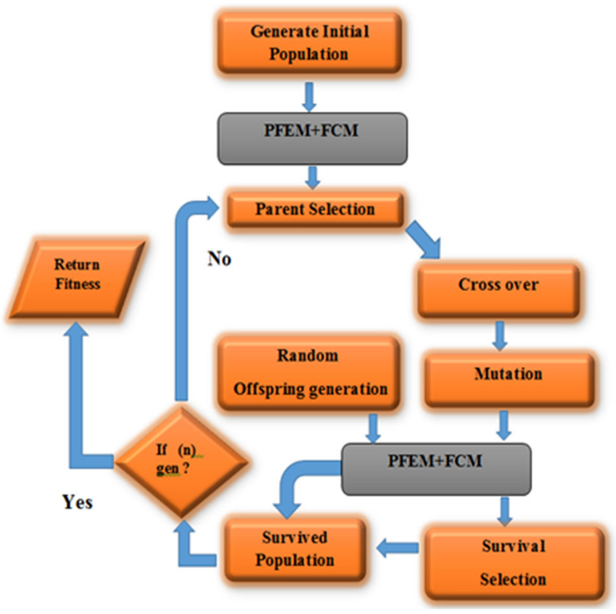

Fig. 6 The process flow of EAEM

Table 2 Algorithm for initial population

```
// Input variables
$X_{seed} \leftarrow \{d_1,d_2,\ldots,d_m\}$ denotes seed
$n_{size} \leftarrow$ denotes population size
$\sigma \leftarrow$ denotes standard deviation
Procedure GenerateInitialPopulation ( $X_{seed}$, $n_{size}$ , $\sigma$)
1: $P \leftarrow \emptyset$
// Randomly variable generation and call to PFEM
2: for ($i \leftarrow 1$ to $n_{size}$)
3:     $d_i^* \leftarrow$ GaussianMutation($d_i,\sigma$)
4:     $X^* \leftarrow \{(d_1^*,\ldots, d_m^*)\}$
5:     $S^* \leftarrow \{X^*$, fitness$\leftarrow$ PFEM ($X^*$)$\}$
6:     $P \leftarrow P \cup \{S^*\}$
7:   end for
8:   return P
```

The algorithm for initial generation of *P* is exhibited in Table 2 and the algorithm for EAEM is depicted in Table 3.

4.2.2 Module II: parallel fitness evaluation module (PFEM)

This module is responsible for the generation of multiple instances which are executed in parallel. Each instance $\mathbf{E_i}$

Table 3 Algorithm for EAEM

```
// Input variables
$n_{gen} \leftarrow$ denotes number of generations
$r \leftarrow$ denotes the number of parents to select
$k \leftarrow$ denotes rate of mutation
$n_{rand} \leftarrow$ denotes random offsprings
$n_{size} \leftarrow$ denotes population size
$\sigma \leftarrow$ denotes user defined standard deviation
$X \leftarrow \{d_1,d_2,\ldots,d_m\}$ represent set of variables
$P \leftarrow$ denotes initial population
Procedure
EvolutionaryAlgorithmExecutionModule ($n_{gen}$ , r, k)
1: Offsprings$\leftarrow \emptyset$, RandOffsprings$\leftarrow \emptyset$, Offsprings*$\leftarrow \emptyset$
// Call to Initial Population Generation Algorithm
2: $P \leftarrow$ GenerateInitialPopulation ( X, $n_{size}$ , $\sigma$)
3: for ($i \leftarrow 1$ to $n_{gen}$)   // Stopping criteria
// Call to Rank based selection method
4:    Parents$\leftarrow$ RBS*(P, r)
5:    for ($j \leftarrow 1$ to r/2 )
// Generating Offsprings (Cross over and Mutation)
6:    Offspring $_j$* $\leftarrow$ CrossoverAndMutation (Parents$_j$, Parents$_{j+1}$)
7:    Offspring $_{j+1}$* $\leftarrow$ CrossoverAndMutation (Parents$_{j+1}$, Parents$_j$)
8:    Offsprings$\leftarrow$ { Offspring $_j$* U Offspring $_{j+1}$* U Offsprings }
9:    end for
// Parallel Fitness Evaluation Module is executed
10:  foreach (X* ϵ Offsprings)
11:     X*.fitness$\leftarrow$PFEM (X*)
12:     $P \leftarrow P \cup \{X^*\}$
13:     $P \leftarrow$ Truncate(P,$n_{size} - n_{rand}$) // Survival
// Random Offsprings generation
14:   for ($c \leftarrow 1$ to $n_{rand}$)
15:     $d_c^* \leftarrow$ GaussianMutation($d_c,\sigma$)
16:     $X^* \leftarrow \{(d_1^*,\ldots, d_m^*)\}$
17:     $S^* \leftarrow \{X^*$, fitness$\leftarrow$ PFEM (X*) $\}$
18:  RandOffsprings$\leftarrow$ RandOffsprings U {S*}
19:    end for
// Combining survived population with random
20:  $P \leftarrow P \cup$ { RandOffsprings }
21:   end for
```

Table 4 Algorithm for PFEM

```
// Input variables
  Xi ←{d_1,d_2,….d_n} denote optimization variables
  M←{M_1,M_2,…,M_p} denote set of workstations
  p← denotes the number of client machines

    Procedure ParallelFitnessEvaluationModule (X_i)
 // Create instance Ei
1:  Ei ←CreateEvaluation(X_i, fitness_i←0)
2:  double sumfitness← 0 , avgfitness←0,
    variance←0
// Parallel Replication
3:  for (i ← 1 to p) , in parallel
4:  Replicate (X_i, M_i)
// Fitness Computation Module is executed
5:  fitness_i←FitnessComputationModule (X_i)
6:  sumfitness←sumfitness+ fitness_i
7: end for
// Average fitness computation
8: avgfitness←sumfitness/p
//  Variance computation
9: for (i← 1 to p)
10:   variance← sqr(fitness_i - avgfitness )+variance
11: end for

// Returning aggregated fitness value
12:   E_i.fitness ← avgfitness – variance/p

13:   return E_i .fitness
```

receives a single tuple $\mathbf{X_i}$ and is responsible for its replication across **p** workstations. After distribution, this module waits synchronously for receiving fitness values from FCM. Since FCM is executed on multiple machines in parallel, therefore, a different fitness value $\mathbf{f_i}$ is received corresponding to same $\mathbf{X_i}$. Thus, the PFEM is responsible for aggregation of all $\mathbf{f_i}$'s to form a single one which accounts for noise and variation in the values received. The consolidated fitness value f_i is then returned to EAEM. The algorithm in Table 4 explains the working of this module.

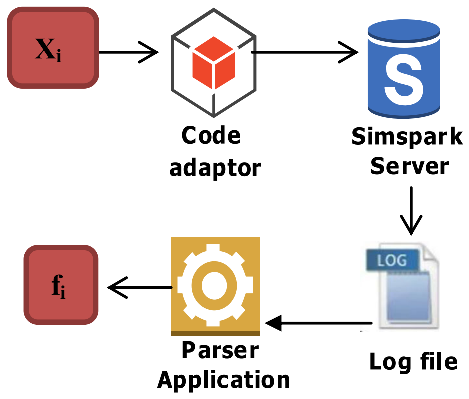

Fig. 7 Phases of fitness computation module

4.2.3 Module III: fitness computation module (FCM)

FCM is responsible for the computation of fitness corresponding to each X_i. This module is deployed on every client machine and performs the following sequence of steps, also depicted in Fig. 7:

1) A code adapter configures the variables of X_i into the developed code of RoboCup Soccer Simulation League 3D team, namely Karachi Koalas (Haider et al. 2014).
2) Simspark (Xu and Vatankhah 2013) server is used to execute simulation match between two teams, T_1 and T_2, and generate a log file.
3) A parser application extracts the required game statistics from the recorded log file for fitness computation that is returned back to the PFEM.

The algorithm for Fitness Computation Module is shown in Table 5.

4.3 Fitness function

This section describes the factors that contribute to the fitness function of a solution.

4.3.1 Goal scored against defending team

The defending team should be able to reduce the number of goals scored by the opponent. Equation 1 describes the goal difference where Score_h denotes the number of goals scored by the home team, while Score denotes the goals scored by the opponent

$$\mathbf{GoalScored} = \mathbf{Score_h} - \mathbf{Score_o} \tag{1}$$

Table 5 Algorithm for FCM

```
// Input variables
T←{T_1, T_2}   where T_1 is the home team and
               T_2 is the opponent team
X← {d_1,d_2,…,d_n} denote optimization variables
S← {X, fitness←0} denote solution in which fitness is missing
L← Log file of match
M← {M_1,M_2,….,M_m} denotes set of workstations
Procedure FitnessComputationModule(S)
1: foreach (m ϵ M)
// Code Adapter
2:   T*_1 ←T_1 (S)
// Play matches
3:   L←play(T*_1 , T_2 )
// Execute Parser Application with log file as input
4:   fitness_m← ParserApplication(L)
5: end for
6: return fitness_m
```

4.3.2 Ball possession by home team players

The defending team should maintain possession of the ball. For this research, it is assumed that ball possessor is the player having the minimum distance from the ball. Equation 2 describes ball possession where Cp_h denotes the number of cycles ball is with the home team player and T_c denotes the total number of simulation cycles.

$$\mathbf{BallPos} = \mathbf{b_p} = \mathbf{Cp_h} / \mathbf{T_c} \tag{2}$$

4.3.3 Average time ball out of danger zone

The defending team should be able to keep the ball out of the danger zone most of the time. For this purpose, the field has been divided into four regions, namely LeftRegion1, LeftRegion2, RightRegion1 and RightRegion2 as shown in Fig. 8. The home team in this scenario is initialized on the left-hand side of the field and the opponent on the right-hand side. The danger zone for the home team is the LeftRegion1. Equation 3 describes the average time ball is out of danger where $C_{\text{LeftRegion2}}$ denotes the number of cycles ball is in LeftRegion2 and $C_{\text{LeftRegion1}}$ denotes the number of cycles ball is in LeftRegion1.

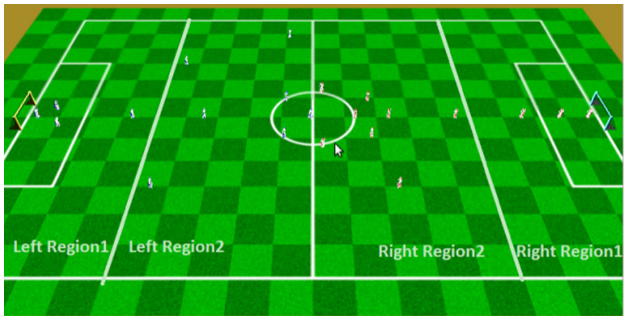

Fig. 8 Regions in a simulated soccer field

$$\mathbf{Avg\ C_{outofdanger}} = \mathbf{C_{LeftRegion2}} / \left(\mathbf{C_{LeftRegion1}} + \mathbf{C_{LeftRegion2}}\right) \tag{3}$$

To evolve a good strategy for defense, we focus on instances where the ball is in the LeftRegion1 and LeftRegion2. However, at times, an uncertain situation may arise in which ball remains most of the time in the right regions. Thus, a solution is penalized if the ball is more than delta cycles in the opponent half. The fitness function that needs to be maximized is shown in Eq. 4.

$$\mathbf{f(x)} = \mathbf{GoalScored} + \mathbf{b_p} + \mathbf{Avg\ C_{outofdanger}} - \partial_{\mathbf{cycles}} \tag{4}$$

The range of fitness function comes out to be [− 2, + 2]. To visually verify the fitness values, logs of few matches were manually replayed.

4.4 Parser application

A parser application that extracts the three mentioned factors has been developed earlier (Larik and Haider 2012) and is shown in Table 6.

5 Experimental setup and results

To demonstrate the effectiveness of FEASO, a series of experiments were designed and executed to investigate:

- parameters of evolutionary algorithm,
- the evolved strategy
- performance of the evolved team against some benchmark scenarios.

The above-mentioned aspects are discussed in subsequent subsections.

Table 6 Algorithm for parser application

```
L← denotes log file of match
$T_t$ ← denotes threshold value
p1← denotes score of the home team
p2← denotes score of the opponent team
Procedure ParserApplication (L)
1: foreach (line ϵ L)
2:   Extract Playmode
3:   If (Playmode == p1) then
4:         $Score_h \leftarrow Score_h + 1$
5:   else If (Playmode ==p2) then
6:         $Score_o \leftarrow Score_o + 1$
7:   Extract ball position
8:   if (ComputeBallRegion($b_x$,$b_y$) == LeftRegion1)
9:          $C_{LeftRegion1} \leftarrow C_{LeftRegion1} + 1$
10:   else   if (ComputeBallRegion($b_x$,$b_y$) ==
             LeftRegion2)
11:            $C_{LeftRegion2} \leftarrow C_{LeftRegion2} + 1$

12: else if (ComputeBallRegion($b_x$,$b_y$) ==
          RightRegion1)
13:     $C_{RightRegion1} \leftarrow C_{RightRegion1} + 1$
14: else if (ComputeBallRegion($b_x$,$b_y$) ==
           RightRegion2)
15:      $C_{RightRegion2} \leftarrow C_{RightRegion2} + 1$
16: Extract home team players position in the field
17: Extract position of opponents in the field
18: Compute distances of players with the ball
19: Extract player with minimum distance as $b_p$
20: if ($b_p$ == home team player)
21:          $Cp_h \leftarrow Cp_h + 1$
22:   $T_c \leftarrow T_c + 1$
23: end foreach
24: GoalScored = $Score_h - Score_o$
25: $b_p = Cp_h / T_c$
26: Avg $C_{outofdanger} = C_{LeftRegion2} / (C_{LeftRegion1} +$
                  $C_{LeftRegion2})$
27:   if   $((C_{RightRegion1} + C_{RightRegion2}) > T_t)$
28:      $\partial_{cycles} \leftarrow 1$   else   $\partial_{cycles} \leftarrow 0$
27: f(x) =   GoalScored+ $b_p$+ Avg $C_{outofdanger}$ - $\partial_{cycles}$
28   return f(x)
```

Table 7 Specifications for FEASO

Operating system	Linux Ubuntu 12.04 LTS
Memory	16 GB RAM
Application	MonoDevelop, Ruby 1.9
Simulator	Simspark version 6.0.6
Workstations	3 (i5)

5.1 Design and analysis of results for parameters of evolutionary algorithm

Initially, experiments were conducted with a varying population size of 5–15. It was observed that with a small population there was little diversity in the solutions, while with larger size, the learning process became slow. Next to fitness evaluations ranging from 1 to 5 were experimented to determine the acceptable number that works best. It was found that due to the noisy environment, single fitness evaluation produced erratic results, while using more than three evaluations was very time-consuming. Thus, the population size was set to 10, while a solution was evaluated three times (that is, three matches were played with the same solution parameters) to get its final fitness. The hardware and the software specifications are shown in Table 7.

The teams that participate in the RoboCup event publish their working code, known as binaries, and after completion of the tournament, these binaries are made public. For the experiments, the binary of team RoboCanes, a leading RoboCup Soccer 3D team from the University of Miami that has been participating since the beginning of the league, was chosen. The reason for its selection was its strong attacking capability. The code of the home team, Karachi Koalas, was deployed on all the workstations which were capable of executing 20 matches of 10-min duration consecutively. Extensive shell scripting was performed for the distributed evolutionary algorithm to minimize the processing time. Karachi Koalas was initialized on the left-hand side while RoboCanes on the right-hand side of the simulated field. During the experiments, the evolutionary algorithm was run for 25 generations. Initial population was generated by applying Gaussian random mutation on the seed solution. For parent selection, RBS scheme was implemented. One-point crossover technique was employed because of its simplicity and also because no obvious advantage was observed in the initial experiments with different orders of *n*-point crossover. The rate of mutation was varied from 0.5 to 0.1, and 0.3 was selected as the final rate. For survival selection, the truncation scheme with the steady-state model was used. For injecting random solutions, a ratio of 20% of the population size was chosen intuitively. The final features of EA are shown in Table 8.

Table 8 Features of evolutionary algorithms

Feature	Technique used
Solution representation	Floating point
Parent selection	Rank-based selection
Mutation	Gaussian random
Crossover	One-point
Survival selection	Truncation
Model	Steady state

Table 9 Input parameters for EAEM

Parameter	Value
n_{gen}	25
R	6
K	0.3
n_{rand}	2
n_{size}	10
Σ	2

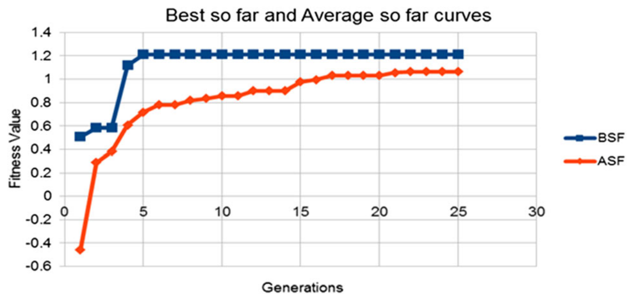

Fig. 9 Average-so-far and best-so-far curves

The input parameters for execution of EAEM, as discussed in Sect. 4, are presented in Table 9.

To compute the fitness of a solution, a match of 300 cycles corresponding to 5 min was simulated on different machines. It took almost 40 min to complete evolution of a single generation. Initially, the evaluation function only considered goal difference but that was not viable as many matches ended without a goal. Next, the average time the ball was out of danger region was added to the fitness function, but in many cases when the ball was moving toward opponent region more fitness value was assigned to those solutions. Thus, a penalty was imposed to cut down those solutions. Figure 9 depicts the average-so-far and best-so-far curves where the blue line represents the best-so-far graph, while the red signifies the average achieved in each of the 25 generations. As evident from the graph, the best fitness value of 1.2 was achieved during the first five generations and the same value dominates toward the end. This signifies that much improvement has been made in initial generations.

The average-so-far graph, on the other hand, started initially with a negative fitness value, but due to truncation-based survival mechanism, the average fitness of the population had improved by the end of the experiments. Next, the best individuals of the fifth generation were compared with the best individuals of the twenty-fifth generations to see whether there is significant diversity in the population. The results shown in Fig. 10 demonstrate the proof of concept where the blue line represents the fifth generation, while the red line denotes the twenty-fifth generation. Although the tenth individual of both the populations achieved the best fitness value 1.2, there is diversity in the rest of the individuals.

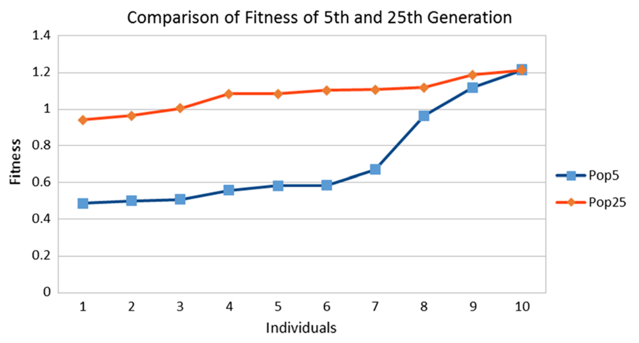

Fig. 10 Best individuals of the fifth generation versus best individuals of the twenty-fifth generation

5.2 Analysis of the evolved strategy

For analyzing the evolved strategy throughout generations, the factors contributing toward overall fitness of solution were given due consideration. At first, only the goals scored against the home team was considered as fitness criteria with the assumption that if the volume of goals scored is reduced then the learned strategic positions were better. However, if the matches ended without a goal, then it was difficult to infer about the overall team strategy. To overcome this issue, the ball possession by the home team was considered a metric for good defense. Moreover, the region where the ball resides also influences the positioning of the agents; thus, the instances in which the defenders were able to keep the ball were also incorporated during fitness computation. Since the log files of matches were available, the best fitness solution files were replayed to manually verify whether there was a significant difference between the initial and final generations. The matches started with the opponent team kicking the ball in the defensive region of the home team. In the initial phases of strategy evolution as shown in Fig. 11, if this kick from opponent team entered the danger zone and five of the opponent agents rushed toward the defensive half, then the home team defenders were able to stop the goal.

Later as the generations evolved, it was observed that the defenders considered their distances with the ball and the goal post and aligned themselves in such a way that they

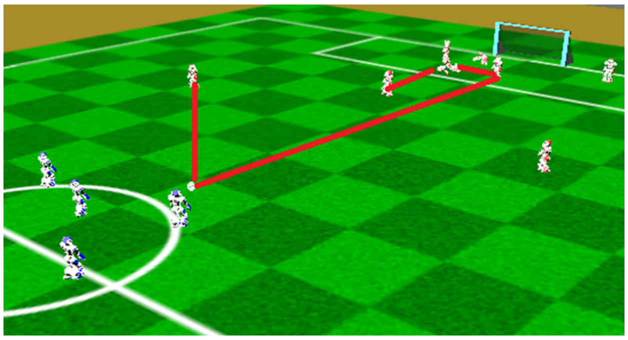

Fig. 11 Initial phases of strategy evolution

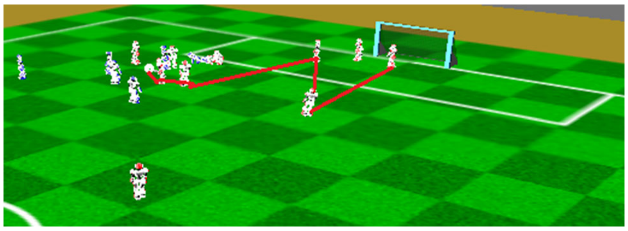

Fig. 12 Final phases of strategy evolution

were able to block the path between the ball and goal as depicted in Fig. 12. The role of defender at the center was of great importance because if either the ball was coming from the left or right it should be able to clear the ball toward the opponent half.

In the evolved strategy, the defending agents were able to keep the ball out of their danger zone. It was noted that the placements of both defenders and midfielders when the goal was on attack were also appropriate in the evolved generations. Figure 13 shows a bar chart comparing the original and the evolved strategic points. The blue bar represents original points, while the orange bar represents evolved points. A yellow line shows the difference between the two values. As evident from the figure, there are two peaks and there is a substantial difference between the values for optimized distances, namely $d2$ and $d7$.

The distance $d2$ signifies the right midfielder and $d7$ shows the position of center defense. Thus, these two play a vital role in the strategic placements of the team.

5.3 Benchmark scenarios

This section conducts a series of matches for analyzing the impact of evolution on team performance. A set of benchmark scenarios were created to test the applicability of the FEASO. For this purpose, three variants of Karachi Koalas team have been devised as follows:

- KKRandom team that decides strategic points randomly
- KKBase team that uses the heuristic seed values that was our viable working strategy in the RoboCup events and
- KKEvolved team that uses the optimized variables obtained via FEASO approach.

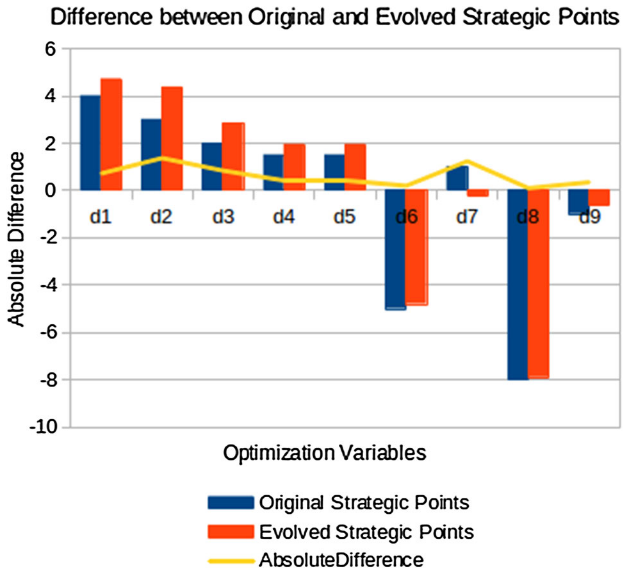

Fig. 13 Bar chart with a line showing the difference between original and evolved strategic points

Table 10 Summarized results of 40 matches executed between variants of KK

Performance measure	KKRandom	KKBase	KKEvolved
Goals scored against	10	7	2
Time to score goal against (cycles)	~ 120	~ 120	~ 180
% of ball possession against	66	50	40

The selected opponent team is the same RoboCanes team that was used for offline strategy evolution.

5.3.1 Competition among the KKRandom, KKBase and KKEvolved

In this experiment, approximately 120 matches were executed between the three sets of teams. Each match was of 220 cycles amounting to 5-min duration under the same time constraints and same opponent used in the learning phase. Table 10 shows the summarized results obtained from 40 matches against each of the teams.

The results demonstrate that the evolved team is able to defend more goals as compared to the KKRandom and the KKBase team. Furthermore, it has also been noted that normally the opponent team scored a goal against both the

Table 11 Statistical test on the significance of results obtained

Comparison	*t* test	Significant
KKEvolved versus KKBase	$p = 0.0383 < 0.05$	Yes
KKEvolved versus KKRandom	$p = 0.0059 < 0.05$	Yes

KKRandom and KKBase team in around 120 cycles but with the KKEvolved team, it was either unable to score a goal and even if it did then it took place after 180 cycles of the match. The percentage of ball possession against each of the teams is also observed and provides a proof of concept that the evolved team defenders are able to retain ball possession as compared to the other two teams.

Table 11 examines the statistical significance of the differences between the KKRandom team, KKBase team and KKEvolved team on the basis of the attribute, goals scored against. The *p* value demonstrates that the KKEvolved team is significantly better than the other two teams.

5.3.2 Competition with other teams participating in RoboCup tournament

This section compares the strategy obtained via evolutionary algorithms with other approaches discussed in the literature review. Experiments were conducted with three sets of teams, namely A, B, and C. Team A uses reinforcement learning strategy, team B uses handcrafted strategy, and team C uses layered learning approach. The entire focus of this paper is to learn defensive strategy; thus, the objective is not to win against other strategies but to show that the evolved team KKEvolved performs better than the benchmarked KKBase team. The novelty of this comparison is to demonstrate that the evolved strategy could be generalized to other opponent teams.

Ten matches each were executed between KKEvolved versus team A, KKEvolved versus team B and KKEvolved versus team C for 5 min. Similarly, ten matches were executed between KKBase versus team A, KKBase versus team B and KKBase versus team C with the same time constraint. Tables 12, 13 and 14 exhibit the summarized results.

It should be mentioned that all the three teams A, B, and C are strong offensive teams with team C being the champion team participating in the RoboCup event. The results show a drastic performance improvement in KKEvolved team proving this method to indeed be very powerful. The approach learned against a single opponent was generalized among other similar opponents.

Table 12 Summarized results of 20 matches executed between team A, KKBase and KKEvolved

Opponent team	Performance measure	KKBase	KKEvolved
Team A (reinforcement learning)	Goals scored against	2	0
	Time to score a goal against (cycles)	~ 180	–
	% of ball possession against	50	30

Table 13 Summarized results of 20 matches executed between team B, KKBase, and KKEvolved

Opponent team	Performance measure	KKBase	KKEvolved
Team B (handcrafted rules)	Goals scored against	10	0
	Time to score a goal against (cycles)	~100	–
	% of ball possession against	60	40

Table 14 Summarized results of 20 matches executed between team C, KKBase, and KKEvolved

Opponent team	Performance measure	KKBase	KKEvolved
Team C (layered learning)	Goals scored against	28	19
	Time to score a goal against (cycles)	~ 80	~ 130
	% of ball possession against	80	30

6 Comparison with most relevant evolutionary approaches

This section discusses the two evolutionary approaches that are similar to the presented FEASO framework.

It was highlighted earlier in Sect. 2 that the work of Chen et al. (2016) and Ali et al. (2014) is more recent and relevant in terms of the evolution of strategy in robot soccer. A comparison among these approaches and FEASO would be drawn theoretically to signify the novelty of proposed framework.

Table 15 Comparison of FEASO with most relevant approaches

	MACO	Adaptive CA	FEASO
No. of agents	Limited	Limited	The entire team of 22 agents
Generalized	No	No	Yes
Domain	2D simulation league	Simple 2D simulator	3D simulation league
Distributed	No	No	Yes

6.1 FEASO versus MACO

Multi-group ant colony optimization algorithm was proposed (Chen et al. 2016) that used ant intelligence to learn offensive strategies in 2D soccer simulation league. In their work, they learned three models, namely shooting model, passing model and dribbling model based on the success of foraging behavior of ants as an effective cooperative strategy. They divided the field into regions, and their focus was to decide the best model to pick based on the pheromone evaporation preference value. They simulated the attack environment by considering two attackers, two defenders and one goalkeeper in the field. The positions of all the five players were set randomly and the training was conducted till 15,000 times in a single match. For evaluation of the proposed strategy, matches were conducted among the baseline team and the evolved team concluding that the evolved MACO team enjoyed a 100% winning advantage. The approach is similar to FEASO as we have also conducted the same set of experiments with varying opponent teams for performance evaluation. The approach, however, is different from FEASO as FEASO utilizes distributed evolutionary algorithm for learning defense, and the focus is on minimizing the number of goals scored. Secondly, ACO comes under swarm intelligence in which the leader dictates the behavior of the whole swarm and in the described approach the leader is the attacker that takes a decision. However, in FEASO approach, there is no leader and team strategic positioning is governed by the optimization algorithm. The MACO approach considers only a smaller version of the team for learning strategy; however, this paper uses the complete team that makes the optimization process much more complex. A known problem with ACO is the huge convergence time required by the algorithm; however, FEASO approach gets an optimal solution in a couple of generations. MACO approach has been tested in a simple, 2D environment with five players in the field. As described earlier, the domain of 3D is much complex due to its humanoid locomotion and FEASO provides fruitful results while working under dynamic, uncertain environment of simulation 3D.

6.2 FEASO versus adaptive CA

The study (Ali et al. 2014) contributed the use of a simplified and adaptive version of CA to develop defensive and offensive plays in simulated robot soccer. The agents were able to develop the most suitable team formation by utilizing a set of finite state machines in the field. The number of goals scored was used for fitness. For each individual in the population, they defined a set of actions and a set of regions. At the start of CA, all the individuals were initialized randomly. The goal was to train the team to choose the best states depending upon the scenario faced. The belief space comprised of general behaviors associated with the overall plan of the team. Experiments were conducted to test the effectiveness of the approach for 150 generations. After every 15 generations, some statistics were computed that indicated the best regions and the best states. For performance evaluation, matches were executed with multiple opponent teams that either played offensively or defensively and the team that used CA played better in terms of defense as well as offense. The CA-based approach is very relevant as this paper also conducts matches against multiple opponent teams and uses goals scored for fitness evaluation. However, it differs from the CA-based approach as besides considering goals scored it also considers other parameters for fitness evaluation. Secondly, they have stated in their research that the simulator provides noisy data that affects the performance of CA algorithm so to overcome this limitation they have developed a simplified version of the simulator in which actions of all the agents are known. In the FEASO approach, the evolutionary algorithm is robust to noise and the approach is learning strategy in an environment that contains the interactions of 22 agents at a given time instance. In addition, CA-based approaches require some knowledge in the form of belief space, whereas this paper starts with a simple heuristic-based seed value with no prior knowledge required at the time of evolution. Lastly, FEASO is scalable and less time expensive due to its distributed nature that is missing in the CA-based approach. Table 15 highlights the comparison between the proposed FEASO and the other two evolutionary approaches discussed in this section.

7 Conclusion

The paper proposes an evolutionary algorithm-based framework for strategy optimization. RoboCup Soccer Simulation 3D has been selected as the test bed for performing these experiments. The team evolved using proposed EA produced better defensive capability as compared to the original team. The simulation results show enhancements in solutions over generations which have been very encouraging and provide initial proof of concept. The performance of the evolved team was compared against other approaches, and it was found the evolved team performed better defense in terms of the goals defended. The future task that needs consideration involves testing and validation of this optimization process for several other teams. Furthermore, a similar approach would also be tested to learn offensive strategies as well. Finally, a multi-objective evaluation function would be formed to test both offensive and defensive scenarios.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Ali MZ, Morghem A, Albadarneh J, Al-Gharaibeh R, Suganthan PN, Reynolds RG (2014) Cultural algorithms applied to the evolution of robotic soccer team tactics: a novel perspective. In: 2014 IEEE congress on evolutionary computation (CEC), pp 2180–87. https://doi.org/10.1109/CEC.2014.6900616

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford

Bezek A (2005) Discovering strategic multi-agent behavior in a robotic soccer domain. In: Proceedings of the fourth international joint conference on autonomous agents and multiagent systems. AAMAS'05. ACM, New York, NY, USA, pp 1177–1178. https://doi.org/10.1145/1082473.1082681

Chen S, Lv G, Wang X (2016) Offensive strategy in the 2D soccer simulation league using multi-group ant colony optimization. Int J Adv Robot Syst 13(1):25. https://doi.org/10.5772/62167

Cherubini A, Giannone F, Iocchi L (2007) Layered learning for a soccer legged robot helped with a 3D simulator. In: Visser U, Ribeiro F, Ohashi T, Dellaert F (eds) RoboCup 2007: robot soccer world cup XI. Lecture notes in computer science. Springer, Berlin, Heidelberg, pp 385–92. https://doi.org/10.1007/978-3-540-68847-1_39

Darab MAD, Ebrahimi M (2007) RoboCup 3D soccer simulation server: a progressing testbed for AI researchers. In: Ellis R, Allen T, Tuson A (eds) Applications and innovations in intelligent systems XIV. Springer, London, pp 228–232. https://doi.org/10.1007/978-1-84628-666-7_18

Gong Y-J, Chen W-N, Zhan Z-H, Zhang J, Li Y, Zhang Q, Li J-J (2015) Distributed evolutionary algorithms and their models: a survey of the state-of-the-art. Appl Soft Comput 34(September):286–300. https://doi.org/10.1016/j.asoc.2015.04.061

Haider S, Williams, M-A, Raza S, Raza A (2014) Karachi koalas 3D simulation team description paper. http://fei.edu.br/rcs/2014/TeamDescriptionPapers/SoccerSimulation/index.html

Kazakov D, Kudenko D (2001) Machine learning and inductive logic programming for multi-agent systems. In: Selected tutorial papers from the 9th ECCAI advanced course ACAI 2001 and agent link's 3rd European agent systems summer school on multi-agent systems and applications. EASSS'01. Springer, London, UK, pp 246–272. http://dl.acm.org/citation.cfm?id=646141.680963

Kitano H, Tambe M, Stone P, Veloso M, Coradeschi S, Osawa E, Matsubara H, Noda I, Asada M (1998) The RoboCup synthetic agent challenge 97. In: Kitano H (ed) RoboCup-97: robot soccer world cup I. Lecture notes in computer science 1395. Springer, Berlin, Heidelberg, pp 62–73. https://doi.org/10.1007/3-540-64473-3_49

Kok JR, Spaan MTJ, Vlassis N (2005) Non-communicative multi-robot coordination in dynamic environments. Robot Auton Syst 50(2):99–114. https://doi.org/10.1016/j.robot.2004.08.003

Larik AS, Haider S (2012) Rule-based behavior prediction of opponent agents using robocup 3D soccer simulation league logfiles. In: Iliadis L, Maglogiannis I, Papadopoulos H (eds) Artificial intelligence applications and innovations. IFIP advances in information and communication technology, vol 381. Springer, Berlin, Heidelberg, pp 285–95. http://link.springer.com/chapter/10.1007/978-3-642-33409-2_30

Lekavy M (2011) Optimising multi-agent cooperation using evolutionary algorithm. In: Proceedings of IIT, Bratislava, pp 49–56

Luh GC, Wu CY, Liu WW (2006) Artificial immune system based cooperative strategies for robot soccer competition. In: 2006 international forum on strategic technology, pp 76–79. https://doi.org/10.1109/IFOST.2006.312251

Murray J, Obst O, Stolzenburg F (2000) Towards a logical approach for soccer agents engineering. In: RoboCup 2000: robot soccer world cup IV. Springer, London, UK, pp 199–208. http://link.springer.com/chapter/10.1007%2F3-540-45324-5_18

Nakashima T, Takatani M, Udo M, Ishibuchi H (2004) An evolutionary approach for strategy learning in robocup soccer. In: 2004 IEEE international conference on systems, man and cybernetics, vol 2, pp 2023–28. https://doi.org/10.1109/ICSMC.2004.1399998

Nakashima T, Takatani M, Ishibuchi H, Nii M (2006) The effect of using match history on the evolution of robocup soccer team strategies. In: 2006 IEEE symposium on computational intelligence and games, pp 60–66. https://doi.org/10.1109/CIG.2006.311682

Okada H, Wada T, Yamashita A (2011) Evolving robocup soccer player formations by particle swarm optimization. In: 2011 proceedings of SICE annual conference (SICE), pp 1950–53

Ramani RG, Viswanath P, Arjun B (2008) Ant intelligence in robotic soccer. Int J Adv Robot Syst 5(1):5. https://doi.org/10.5772/5657

Riedmiller M, Gabel T, Hafner R, Lange S (2009) Reinforcement learning for robot soccer. Auton Robots 27(1):55–73. https://doi.org/10.1007/s10514-009-9120-4

Salhieh A, Mostafa A, Mahmoud I, Reynolds R (2012) Evolving effective multi-robot coordination strategies for dynamic environments using cultural algorithms. In: Proceedings of the 12th WSEAS international conference on system theory and scientific computation. Istanbul, Turkey. http://www.bibsonomy.org/bibtex/2273bdd62c423eab5eb9d2a8667ddb127/imahmoud

Sałustowicz RP, Wiering MA, Schmidhuber J (1998) Learning team strategies: soccer case studies. Mach Learn 33(2–3):263–282. https://doi.org/10.1023/A:1007570708568

Stone P, Veloso M (2000) Layered learning and flexible teamwork in robocup simulation agents. In: RoboCup-99: robot soccer world cup III. Springer, London, UK, pp 495–508. http://dl.acm.org/citation.cfm?id=646584.698521

Svatoň V, Martinovič J, Slaninová K, Snášel V (2014) Improving rule selection from robot soccer strategy with substrategies. In: Saeed K, Snášel V (eds) Computer information systems and industrial management. Lecture notes in computer science, vol 8838.

Springer, Berlin, Heidelberg, pp 77–88. https://doi.org/10.1007/978-3-662-45237-0_9

Urieli D, MacAlpine P, Kalyanakrishnan S, Bentor Y, Stone P (2011) On optimizing interdependent skills: a case study in simulated 3D humanoid robot soccer. In: Richland SC (ed) The 10th international conference on autonomous agents and multiagent systems, vol 2. AAMAS'11. International foundation for autonomous agents and multiagent systems, pp 769–776. http://dl.acm.org/citation.cfm?id=2031678.2031727

Wang J, Wang T, Wang X, Meng X (2009) Multi-robot decision making based on coordination graphs. In: International conference on mechatronics and automation, 2009. ICMA 2009, pp 2393–98. https://doi.org/10.1109/ICMA.2009.5246091

Xu Y, Vatankhah H (2013) SimSpark: an open source robot simulator developed by the robocup community. In: Robot soccer world cup. Springer, Berlin, Heidelberg, pp 632–39. https://doi.org/10.1007/978-3-662-44468-9_59