A Comprehensive Evaluation of the Methods for Evolving a Cooperative Team

Yasuyuki Suzuki Takaya Arita
Graduate School of Information Science
Nagoya University
yasu@create.human.nagoya-u.ac.jp arita@nagoya-u.ac.jp

Abstract

This paper focuses on the techniques of evolutionary computation for generating players performing tasks cooperatively. However, in using evolutionary computation for generating players performing tasks cooperatively, one faces fundamental and difficult decisions including the one regarding the so-called "credit assignment problem". We believe that there are some correlations among design decisions and therefore a comprehensive evaluation for them is essential. We first list three fundamental decisions and possible options in each decision in designing methods for evolving a cooperative team. We find that there are 18 typical combinations available to execute. Then we describe the ultimately simplified soccer game played on one-dimensional field as a testbed for comprehensive evaluation for these 18 candidate methods. The results are analyzed in this paper.

1 Introduction

Some problems can be efficiently solved only by teams consisting of cooperative autonomous players. Many researchers have developed methods that don't require human designers to define specific behaviors of players for each problem. The work reported here focuses on the techniques of evolutionary computation, which has been regarded as one of the most promising approaches to solve such complex problems. However, in using evolutionary computation for generating players performing tasks cooperatively, one faces fundamental and difficult decisions including the one regarding the so-called "credit assignment problem" [1]. For example, if we can only evaluate the global performance of each team, how do we divide up the team's performance among the participating players? We believe that there are some correlations among design decisions and therefore a comprehensive evaluation for them is essential, although not a few researchers have proposed evolutionary methods for evolving teams performing specific tasks.

The rest of the paper is organized as follows. In Section 2, we list three fundamental decisions and possible options in each decision in designing methods for evolving a cooperative team. We find that there are 18 typical combinations available to execute. Then, in Section 3, we describe the ultimately simplified soccer game played on one-dimensional field as a testbed for comprehensive evaluation for these 18 candidate methods. Section 4 reports on the results of the comprehensive evaluation of these methods, and Section 5 summarizes the paper.

2 Methods for Evolving a Team

Three fundamental decisions are necessary when one designs an evolutionary computation method for generating players performing tasks cooperatively, and there may be not a few combinations of the options in these decisions.

The first decision is: How many evolving populations are there? The answer is derived by considering whether or not the population structure depends on the number of the teams in the game or the number of the player roles in the game (Figure 2). Suppose that the game is played by 2 teams consisting of 3 players. We can assume an evolutionary computation with 2 populations corresponding 2 teams, with 3 populations corresponding 3 players, or with 6 populations corresponding to 2 teams and 3 players. So, the typical options for the number of the populations are 1, R, T and $T \cdot R$ (T: Number of teams in the game, R: Number of the player roles in the team).

The second decision is: What does each individual (genome) represent? Typical options are a player and a team. In case of that each genome represents a player, there can be two further options: all players in the team share one genome ("homogeneous players") and all players are represented by different genomes ("heterogeneous players"). Also in case that each genome represents a team, there can be two further options: whether or not the roles of the players represented in each genome are fixed. In case that the roles of the player is fixed, for example, if a part of a genome represents a defender in the game, this part always represents a defender.

The third decision is: How is the fitness function evaluated? One option is that fitness is evaluated for a team as a whole. In this case, if each genome represents a player, each player in a team is supposed to have the same fitness. The other option is that the fitness is evaluated for each player directly or indirectly. Direct evaluation of players in a cooperative team is sometimes a very difficult task, as in general altruistic behavior is important or essential in the establishment and maintenance of cooperation in population. Some methods for indirect evaluation has been proposed [2]. We adopt a method as this option in which the fitness of a player is defined as the decrease in the fitness of the team when the player is replaced by a predefined "primitive player" which has a minimum set of behavior rules.

Therefore, there could be 18 combinations available to execute for evolving players performing tasks cooperatively as shown in Table 1.

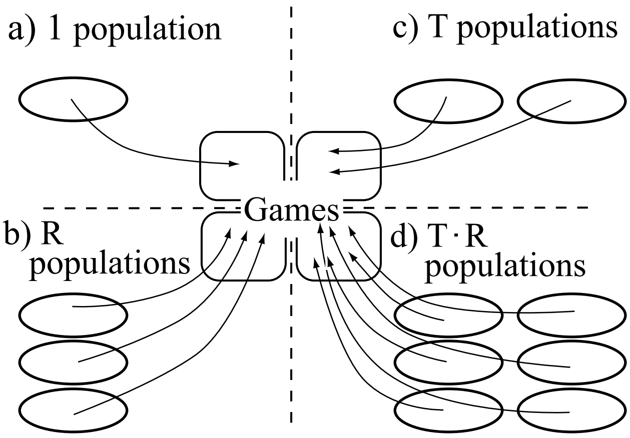

Figure 1: 4 options for the population structure. a) The population represents all player roles in all teams. b) Each population represents one player role in all teams. c) Each population represents all player roles in each team. d) Each population represents one player role in each team.

3 Ultimately-Simplified Soccer Game

The ultimately-simplified soccer game is defined as a testbed for comprehensive evaluation for these 18 candidate methods. It is a 2 vs. 2 player game played on one-dimensional cellular field as shown in Figure 2 (field[1-20]). Players are homogeneous except their starting positions (Left team: player1 (field[8]), player2 (field[5]), Right team: player1 (field[13]), player2 (field[16])), and each player makes a run, dribbles a ball, makes a shot on goal or put a ball up to the player of his/her team. One of the action is decided to take based on the relative locaton of all players and the ball (72 patterns). Action is taken in turn alternatively between 2 teams. Each step in the game is composed of 4 actions by all players.

Multiple players can't be in a cell. The ball is always in a cell where a player resides. Moving action of a player with a ball means dribbling. Players move to either of the neigboring cells, but when a player moves to the cell with a player, it skips the neigboring player (it cannot skip more than one player). In this case, if both are in opposite teams and one of them has a ball, the ball moves to the other player with a certain probability (P_{steal}). If there is an opponent player between the passer and the receiver, the ball-passing becomes failure with a certain probability (P_{cut}), and in this case the ball moves to the cell where the oppenent player resides. The success rate for shooting is anti-proportional to the length between the player's position and the goal irrespective of the presence of the oppsite players. In case of scoring a goal, the game restarts with initial player-location. In case of the failure, the game restarts after the ball moves to the opposite player nearer to the goal post.

We expect two types of altruistic behavior which could lead to the emergence of cooperation in the game. One is putting a ball up to the other player in his/her team instead of dribbling the ball or getting a shot at the goal. The other type is making a run in the opposite direction but not toward the goal. The former type of altruistic behavior is analyzed in 4.3.

4 Evaluation

4.1 Expression of the Players

Each player selects next action deterministically based on the positional relationship of players and the ball. In the recognition of each player, opponent players are not distinguished. So, to be precise, genetic information of each player decides the next action of the player based on one of 48 patterns, in which each pattern is associated to one of the four actions: running/dribbling to the right, running/dribbling to the left, feeding (passing) the ball to the player of his/her

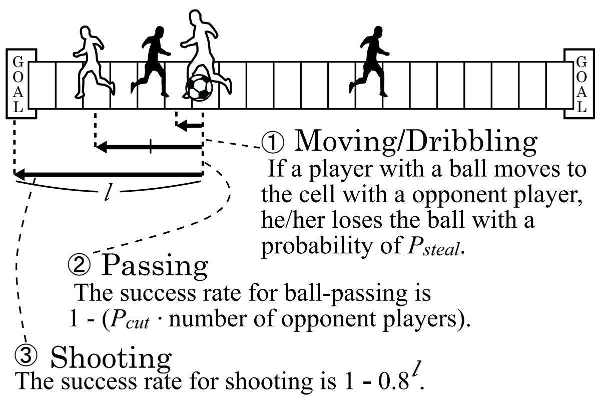

Figure 2: The ultimately simplified soccer game.

Table 1: Classification of the methods for evolving a team.

Population structure depends on T?	Population structure depends on R?	Number of populations	Each genome represents		Unit of fitness evaluation is		Code name
No	No	1	a player	heterogeneous players	a player	by direct evaluation	1-PHe-PD
						by indirect evaluation	1-PHe-PI
					a team (same fitness in a team)		1-PHe-T
				homogeneous players	a team (same fitness in a team)		1-PHo-T
			a team	fixed player-roles	a team		1-TFi-T
				unfixed player-roles	a team		1-TUn-T
	Yes	R	a player	heterogeneous players	a player	by direct evaluation	R-PHe-PD
						by indirect evaluation	R-PHe-PI
					a team (same fitness in a team)		R-PHe-T
Yes	No	T	a player	heterogeneous players	a player	by direct evaluation	T-PHe-PD
						by indirect evaluation	T-PHe-PI
					a team (same fitness in a team)		T-PHe-T
				homogeneous players	a team (same fitness in a team)		T-PHo-T
			a team	fixed player-roles	a team		T-TFi-T
				unfixed player-roles	a team		T-TUn-T
	Yes	$T \cdot R$	a player	heterogeneous players	a player	by direct evaluation	TR-PHe-PD
						by indirect evaluation	TR-PHe-PI
					a team (same fitness in a team)		TR-PHe-T

(T: Number of teams in a game, R: Number of player roles in a team)

team, making a shot on goal. Therefore each player is represented by 96 bits genetic information.

4.2 Evaluation Setting

The evaluation is conducted through two steps: an evolution step and an evaluation step. In the evolution step, populations are evolved for 2000 generations using 18 methods independently. Each population has 40 individuals in all methods. The round-robin tournament of the ultimately-simplified game of 200 steps is held to evaluate the fitness in each generation. The parameters P_{steal} and P_{cut} are set to 0.8 and 0.4 respectively in both steps. In case of <team-evaluated> option, fitness is calculated as the goals the team acquired minus the goals the opponent team acquired. In case of <direct-player-evaluated> option, fitness is calculated as the goals the player acquired minus the opponent team's goals divided by 2. Then tournament selection (selecting repeatedly the individuals with higher fitness as a parent by comparing randomly picked 2 individuals), crossover with a 60% probability and one-point mutation with a 3% probability are adopted as genetic operators. In case of <indirect-player-evaluated> option, we use a primitive player designed a priori as follows. In case of the player keeps a ball, if he (or she) is behind the other player he passes the ball to the other player, otherwise he makes a shoot. In case of the player doesn't keep a ball, if he is behind the other player, he moves back, otherwise he moves toward the goal. In the evaluation

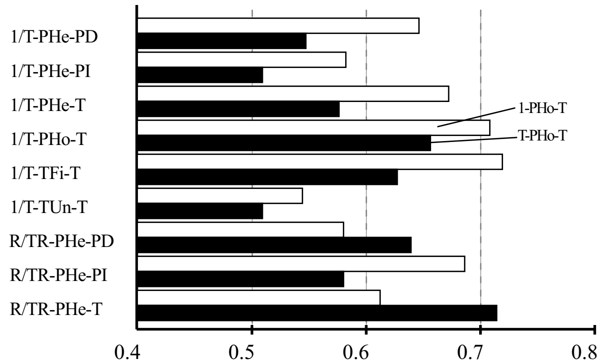

Figure 3: The average winning ratio of the best 10 teams evolved by each of 18 methods.

step, the best team is selected in each of the last 50 generations in the evolution step, and selected 50×18 teams conduct the other round-robin tournament of the game of 1000 steps.

4.3 Evaluation Results

Figure 3 shows the winning ratio of the teams evolved by 18 methods, each of which is the average of winning ratio of the best 10 teams from 50 teams in the all-play-all tournament described above. Table 2 (the left-hand in the results) also shows it. Each pair of bars shows the results of the strategies with same options in genome representation and fitness evaluation except the population structure option (Upper white bars: <$1/R$-population> options, Lower black bars: <$T/T \cdot R$-populations> options).

It is shown that the top 3 methods in this evaluation are <1-population, team-represented with fixed player-roles, team-evaluated>, <$T \cdot R$-population, heterogeneous-player-represented, team-evaluated>, and <1-population, homogeneous-player-represented, team-evaluated>. Their winning ratios are 74.6%, 74.1% and 73.5% respectively. An additional evaluation using the team consisting of 2 primitive players showed that its winning ratio was 16.0%. This ratio could be a measure for the performance of the methods.

Regarding to population structure, <$1/R$-populations> options performed better than <$T/T \cdot R$-populations> options in general. This might be because of the ill-balanced evolution, over-specialization or "round and round going". Adoption of an asymmetric game as a testbed would make this tendency weaker. Regarding to genome representation, <homogeneous-player-represented> option performed well in general. Also, <team-represented with fixed player-roles> option performed well, though <team-represented with unfixed player-roles> option performed badly. Regarding to fitness evaluation, <team-evaluated> option performed well in general as the fact that 5 methods among top 6 methods adopt this option has shown. The performance of <indirect-player-valuated> option depended largely on the other options.

We have observed interesting separation of roles among 2 players in the teams with high winning ratio. For example, in some teams the forward player tended to play near the goal and the backward player tended to move in order to intercept the ball, and in some teams both players seemed to use man-to-man defense.

Next we examined the relationship between altruistic behavior which could lead to cooperative behavior and the winning ratio. Here we focus on the following behavior pattern. A player with a ball makes a pass for the other player, who receives the ball without being intercepted and then successfully shoots a goal immediately or after dribbling. We termed this series of actions as "assisted goal". Table 2 shows the assist ratio, which is the ratio of assisted goals among all goals, and winning ratio of the teams evolved by 18 methods. We see from this table that good performing teams have a tendency to also have a high assist ratio. In contrast, it is not necessarily the case that teams with a high assist ratio have a tendency to have a high winning ratio. This means that above-defined assisting behavior is a necessary requirement for the teams to perform well.

It is a remarkable fact that <indirect-player-evaluated> option made the assist ratio much higher compared with the winning ratio. As this option, we adopted a method in which the fitness of a player is the decrease in the fitness of team when the player is replaced by the primitive player. This method should generate the strong interaction between 2 players because it tends to make large decrease when the player is replaced. Therefore the teams generated by the indirect evaluation method have a higher assist ratio despite having the relatively low winning ratio.

Table 2: Average winning ratio and assist ratio.

Code name	Results			
	Winning ratio	Rank	Assist ratio	Rank
1-PHe-PD	0.673	7	0.146	15
1-PHe-PI	0.609	11	0.289	10
1-PHe-T	0.699	5	0.310	9
1-PHo-T	0.735	3	0.390	5
1-TFi-T	0.746	1	0.342	7
1-TUn-T	0.571	16	0.336	8
R-PHe-PD	0.607	12	0.109	16
R-PHe-PI	0.713	4	0.503	1
R-PHe-T	0.639	10	0.402	3
T-PHe-PD	0.574	15	0.080	17
T-PHe-PI	0.536	17	0.391	4
T-PHe-T	0.603	14	0.242	12
T-PHo-T	0.683	6	0.226	13
T-TFi-T	0.654	9	0.260	11
T-TUn-T	0.536	18	0.214	14
TR-PHe-PD	0.666	8	0.077	18
TR-PHe-PI	0.607	13	0.388	6
TR-PHe-T	0.741	2	0.416	2

5 Conclusion

This paper has focused on the methods for evolving a cooperative team by conducting a comprehensive evaluation for 18 methods. We have found that some methods performed well and at the same time that there are complex correlations among design decisions. Also, further analysis has shown that cooperative behavior can be evolved and can be a necessary requirement for the teams to perform well in even such a simple game. Future work includes more detailed analysis of cooperative behavior and extension of the ultimately-simplified soccer game.

References

[1] Thomas Haynes, Sandip Sen, Dale Schoenefeld and Roger Wainwright, "Evolving a Team", *Working Notes for the AAAI Symposium on Genetic Programming*, pp. 23-30, 1995.

[2] Thomas Miconi, "A Collective Genetic Algorithm", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pp. 876-883, 2001.