Scenario-Based Teamworking, How to Learn, Create, and Teach Complex Plans?

Ali Ajdari Rad[1], Navid Qaragozlou[1], and Maryam Zaheri[2]

[1] Computer Engineering Faculty, Amirkabir University of Technology, Tehran, Iran
{alirad,navidq}@aut.ac.ir
[2] Computer Science Department, University of North Carolina at Charlotte, USA
mzaheri@uncc.edu

Abstract. This paper presents the application of a novel method in the multi-agent teamwork field called Scenario-based Teamworking (SBT). In SBT method a team of cooperative intelligent agents could be able to execute complex plans in nondeterministic, adversary, and dynamic environments which communication cost is high. The base idea of this method is to define Scenario for different situations. With a graph of scenarios, a team of agents can execute, learn, adapt, and create team plans automatically. This method has implemented in a soccer team of intelligent agents (players and coach) and evaluated in the standard RoboCup simulator environment [1] and results show a significant improvement.

1 Introduction

One of the most important and complicated problems in designing multi-agent systems is the agents teamwork. A team of intelligent agents without cooperation will not going to act as well as a team of agents with less individual intelligence but teamwork understanding. The more complex environment and the higher cost for communication among agents make it harder to design a method for teamwork and managing agents.

The creation of the robotic soccer, the robot world cup initiative (RoboCup), is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined [2]. Some of the fields covered include multi-agent collaboration, strategy decision making, intelligent robot control and machine learning. In the RoboCup simulation league, there is a soccer simulator that simulates a soccer game. Each team should introduce 11 players and an optional coach to this simulator. The simulator sends the sensory information to players and players should declare their actions to the simulator via a network connection in a standard protocol. Like a real match, each player can not see the entire field (just watch what is in front of him) and has limited stamina. Players should communicate via the soccer simulator (with sending say commands). In this situation, a coach with global and noiseless view of the field can dynamically improve the teamwork of this team.

Similar to real soccer games, the simulated teams can have a coach. The duty of the coach is to employ appropriate tactics based on abilities of teammates and also the

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 137–144, 2004.

strategy of the opponents. Furthermore, the coach is responsible to finding the weakness of its own team and improves their teamwork by applying appropriate strategies [3].

In this paper we present a powerful method to define plans for a team of soccer player agents called Scenario-based Teamworking. The main idea in this method was introduced by the Canned Plans concept of Essex Wizard team [4]. Using scenario – based approach, player agents are able to learn, adopt and execute complex team plans against their opponent, and the coach agent is able to modify and teach plans to its players. Another advantage of SBT is opponent modeling and the capability of automatically creating new plans.

2 Teamwork Based on Scenarios

The SBT method is based on the concept of defining scenarios. In this method a scenarios is defined for each team plan. Each scenario includes:

- Triggers: conditions that explain the current situation of the environment and internal state of the agent. These conditions are divided as follows:
  - Data Triggers: facts, produced by direct information about the environment. For example, agent is in area #1, or ball is on opponent side.
  - Time Triggers: concepts which are dependent on time, such as playing modes.
  - Communication Triggers: situations that are affected by agents' communications, such as "Agent #1 said …" or "Agent #2 sent a pass request."
  - Action Triggers: situations, which are related to an agent's action, such as owning the ball, or shooting the ball.
  - Situation Triggers: Conditions that are brought about by what happens in the game. Usually these conditions are related to the high level concepts in soccer such as attack mode, or crowding around the ball.

  According to the above classifications, we can define different and complex situations. For example:
  T1: Agent1 is ball owner & Agent2 is near Area3 & Agent3 sends a pass request
- Goal: describes the final goal of the plan. In simulated soccer field, we categorize the final goal as *Scoring*, *Clearing*, and *Possession,* when the team is ball owner; and *Blockade, Close goal*, and *Close pass,* when the opponent is the ball owner. Scoring scenarios occur near the opponent's goal and describe a plan that its final action is shooting to opponent's goal and scoring. Clearing scenarios occur near home goal and describe a plan that aims at kicking the ball away from home goal. Possession scenarios occur in the middle of the field and aim art keeping the ball and creating a chance to achieve Scoring situation. Blockade scenarios are selected when some agents want to obtain the ball from opponents (make pressure on ball). In Close goal scenarios, agents will close home goal so that the opponent ball owner will not be able to score. The final goal of Close pass is to force the opponent ball owner to keep the ball or make a bad pass by closing its useful pass lines.
- Abort Conditions: describes the conditions that abort the plan. They are defined just like Triggers (but in negative meaning). With separating these two set of con-

ditions, we make the concept simple. In addition, we can benefit from some features, such as approximate matching and risk management involving Triggers or Abort conditions.

- Evaluation: each scenario has its own evaluation parameters. We can classify evaluation parameters in to two groups: Cost and Score parameters. Cost is a real number that is determined by the designer in the beginning. The designer can describe the amount of cost as a function of time, power of the agent, number of agents participating in the scenario, the effects of the incomplete scenario (if this scenario fails) and other concepts. The score is also a real number and shows the rewards that are obtained if the scenario finishes successfully. These parameters can be learned and changed during a game.
- Side Effects: doing a scenario will change the environment and create new situations. For example "playing in the field width" scenario makes the play wide, and a scenario based on fast and long passing will increase the speed of the game.

Considering above descriptions, we define the main plan in the scenario model. Each scenario includes a set of sub-plans that are performed step-by-step. Each step includes actions, whose its main goal is scoring is shown in figure 1.

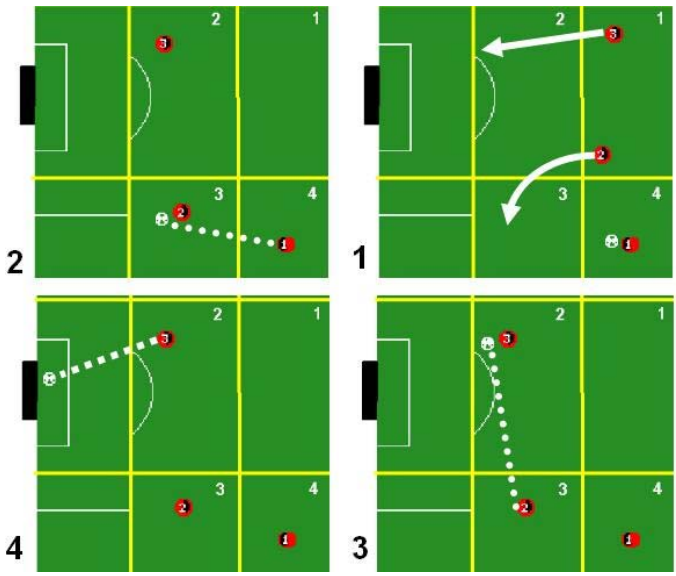

Fig. 1. Sample Scoring scenario.

```
Goal: Scoring   Cost: 3.8   Score: 17.3
Step 1:
Triggers: Agent1 is ball owner
          Agent1 is in Area3
          Agent2 is near Area4
          Agent3 is near Area2
```

```
Actions:  Agent1 passes ball to agent2 in Area4
          Agent2 receives pass from Agent1 in Area4
          Agent3 moves to Area2
Step 2:
Triggers: Agent2 is ball owner
          Agent2 is in Area4
          Agent 3 is in Area2
Actions:  Agent2 passes ball to agent 3 in Area2
          Agent3 receives pass from Agent2 in Area2
Step 3:
Triggers: Agent3 is ball owner in Area2
Actions:  Agent3 shoots to goal
Side Effects: Speed up the game
```

In the real game three players in the above condition can choose this scenario even with the lowest and least accuracy information. To make sure that agents are in the same scenario, each agent can have a priority to suggest a scenario. For example the agent who owns the ball has more priority to suggest the scenario (selecting a scenario could be done in voting or contract-net techniques). Choosing a scenario identifies the agent's roles and then there will be a matching procedure to match players with agents in the scenario, such as Player#7→agent2 (This assignment shows the player number 7 should accept agent number 2 role in selected scenario). This matching also could be done central or distributed (We implement central matching). In this case each player knows its role, and the player can do it without extra communication (in fact with a few communications).

A major problem in the other methods (like Canned Planes) was incompatibility of plans with the new situation or new opponent's plan. In SBT model, while a scenario is performed in a game and it is successful, its score increases with a coefficient. If the scenario fails, then its score decreases with a coefficient. In this way we can have a kind of adaptation during the game. Later we define a more flexible method for adaptation.

In the above, we have described how to define and implement a scenario. There is one more problem remaining which it is effect of past scenarios on current one. In a system like RoboCup, the current situation is affected by previous situations, such as environment and agents' behaviors; therefore one of the most important decision factors is the previous situations. For example in playing soccer, the percentage of success for the scenario shown in fig 1 after a scenario that its main goal is *Possession* is more than a scenario that its main goal is *Scoring*. So the success or fail of a scenario is related to its previous scenarios.

We implement this fact with making a graph by scenarios. Each node of the graph represents a scenario. Presence of an edge between each two nodes shows the probability of happening two scenarios sequentially. The weight of each edge represents the probability of success of the destination scenario after playing the departure sce-

nario. The above idea makes the team to learn and perform a sequence of scenarios. For example a graph which starts with "Clearing" scenario and ends with a Scoring scenario is a complete plan.

For implementing this part, we use a graph of scenarios. In the first phase, scenarios are not related to each other. The first scenario is chosen in an experimental game based on matching of triggers. Agents execute the selected scenario and when it finishes; another scenario is selected based on triggers matching. In this case an edge is created between the first scenario and the second one in the graph. We continue the process as long as nodes of the graph are connected together with an acceptable threshold. In the second phase, we randomly choose a path of the graph that begins with the clear scenario and end with scoring scenario. Now agents execute scenario sequence without any evaluation. When a scenario wins we give positive score to the edge that leads us to the node, and if the scenario fails we give negative score to that edge and the sequence is failed. If the total path is successful we can give positive score to the total path.

In this manner, meaningful weights would be assigned to the edges of the graph in different games. Finally, in a real game, agents select a scenario based on the previous parameters and weight of the output edges. Also we can continue the learning phase during the real game. In this order, we can use the effect of more performing scenarios. At last scenarios are chosen with respect to these parameters:

1. Matching triggers with current conditions: Scenarios are evaluated based on matching with the current situation of the game. We can obtain a matching score for each trigger set (just like fuzzy rule based systems). For example nearness of a player to an area could be used as a fuzzy variable in this case.
2. Matching abort conditions with current conditions: Matching abort conditions are evaluated like triggers. This parameter has negative effect in the decision process.
3. Matching the scenario goal with the local team goal: The Scenario goal is compared with the team goal. For example, all teams desire scoring near opponent goal area. So scenarios whose goal is clearing should not be chosen and vice versa for scoring scenarios near home goal.
4. Matching side effect of the scenario with general strategy of the team. If a team wants to speed up the game, the scenarios having speed up side effect should be chosen. So the side effect of the scenario should be considered in the decision process.
5. We take into consideration costs and scores. In general, scenarios that have good score and low cost should be chosen. For example, a scenario that takes long time (high cost), or has lost many times (poor score) should be omitted.

In implementation, we mapped all above parameters to a real number between 0 and 1. Finally the optimum scenario is determined by using a weighted average of all the above-mentioned parameters. The designer can adjust the effect of each parameter with its coefficient.

Another benefit of SBT approach is the ability of the coach to automatically create new scenarios. This can be done in two ways:

– Watching the opponent's playing method, the coach can model the opponents' scenarios by determining current trigger, opponent's selected actions, side effects, and other features of a scenario (some features may not be determined exactly). The opponent modeling procedure can be done in a similar way [6, 7].

– Creating new scenario with evolutionary methods by the coach. New scenarios are built with random triggers and actions and then an evaluation phase determines the usability of it. Because of complexity of the environment, some limited rules should be considered to reach a better performance.

Then these new scenarios are added to team's scenario bank and are used during games.

In a real game, based on the opponent's conditions (prior knowledge about its pervious plays), one of the scenario graphs will be selected by human, before the beginning of the game (a team may have different scenario graphs for different strategies). Players select and execute scenarios and observe the result. Results will modify the graph so the behavior of team will be changed based on its online experiments. The modification of graph could be done distributed (each player modifies its graph and some decision sharing is done) or done by the coach and broadcast to players periodically. So two levels of adjustable autonomy are seen here: Human-Coach (if online modification is allowed) and Coach-Player.

3 Evaluation

In this part, we explain the results which are gained by performing and evaluating SBT method. The scenario bank of Pasargad team has around 50 scenarios. 50% of the scenarios are with Possession goal, 20% with Scoring, and the rest have other main goals. Usually, in each time of game two scenarios are performing, and the average time of any scenario is 100 cycles. For evaluating the effect of SBT method in a team, the evaluation part has implemented by two version of Pasargad team with different decision methods (Individual techniques are same).

1. Pasargad team plays based on decision tree.
2. Pasargad team plays based on scenario model.

We should mention that there are 10 games held for evaluation, and then the average result is rounded by 0.5. Each win scored 3 and each draw scored 1.

3.1 Decision Tree Version

Pasargad team has a high capability for individual actions, because of its specific architecture and optimum algorithms, which are using for technical actions in the player's code. So this team can show good performance with using simple ways to organize team working (Table 1).

Table 1. Results that obtained by Pasargad team that uses decision tree.

	1st half	2nd half	Final	Defeat	Draw	Score
FCPortugal2000	2-1	2-0	4-1	6	3	6
ATTUnited2000	1-0	2-1	3-1	3	4	13
Dirty Dozen	0-4	0-4	0-8	0	0	30
Pasargad	3-2	2-2	5-4	2	7	10

3.2 SBT

We conducted games between the same teams and Pasargad that play based on the scenario model. The result changed remarkably (Table 2).

Table 2. Results that obtained by Pasargad team that uses SBT method.

	1st half	2nd half	Final	Defeat	Draw	Score
FCPortugal2000	1-1	1-2	2-3	1	4	21
ATTUnited2000	0-1	0-3	0-4	0	2	26
Dirty Dozen	0-5	0-9	0-15	0	0	30
Pasargad (DTree)	1-2	1-4	2-6	0	1	28

In the decision tree version, Pasargad scored 1.4 average score and obtained an average result 3-3.5 for a game. Using SBT method, Pasargad reached 2.6 average score and 1-7 average results per game. So SBT could increase scoring about 40% and increased results about 6 goals per game.

Another interesting conclusion regarding the behavior is convergence of SBT approach. The team performance improves during the game and better results are always obtained in the 2nd half. Successful sequences of scenarios are repeated during the game. This fact means that the weaknesses of opponent strategy are found and team uses this knowledge to select appropriate scenarios to defeat the opponent. Also success of selected scenarios increases about 50% in the second half.

The benefit of using the SBT approach is shown in the game between the two versions of Pasargad shown in the last row of table 2. Both teams use the same base team, and the individual player actions are the same. A 60% growth of scoring is observed here just due to the SBT approach.

In a different experiment, the power of the SBT approach is seen more clearly. We made the duration of games five times longer than standard game duration (30'000 cycles), so the team had more time to learn. Table 3 shows the scoring results for two games with 6000 cycle window.

Table 3. Performance growing during a long game.

	1st	2nd	3rd	4th	5th
FCPortugal2000	2-3	2-4	2-7	1-4	0-5
ATTUnited2000	0-3	0-4	0-8	0-9	0-8
Dirty Dozen	0-16	0-18	0-24	0-26	0-26
Pasargad (DTree)	2-5	1-8	1-12	0-13	1-14

It seen that there is a fast growth of success until the 3rd segment of the game. Then this growing is slowed down, and reaches to the highest result that Pasargad with SBT could reach against its opponent (if the game was continued the result approximately would be unchanged). According to this, using offline training and the SBT approach (for example using past games against a specific opponent), the team can achieve a result twice better than the ordinary result.

4 Conclusion and Future Works

The Scenario-based approach was used in this paper for analyzing and arranging. The results show improvement of teamwork. Also implementing this team and analyzing situations are easier.

The SBT approach could be used in other areas of multi agent systems. In simulated soccer games, implementation of SBT with standard coach language [6], representing an architecture that adapts this concept [7], using efficient methods for automatically creating new scenarios, and using more parameters to implement efficient adjustable autonomy for coach and players are suggestion for future works.

Acknowledgements

We hereby appreciate Dr. Reza Safabakhsh our advisor and Dr. Ahmed Abdollahzadeh the professors of computer engineering department of Amirkabir University for their technical advices.

References

1. Noda, H. Matsubara, K. Hiraki, and Frank, l. “Soccer server: A tool for research on multi agent systems”, Applied Artificial Intelligence, 12:233-250, 1998.
2. Riley, P., and Manuela Veloso, M., “Coaching a Simulated Soccer Team by Opponent Model Recognition”, In Proceedings of the Fifth International Conference on Autonomous Agents (Agents-2001), 2001.
3. Pourazin, S., Ajdari Rad, A., Atashbar, H., “Pardis, Our team in RoboCup simulation league”, Linkoping University, Electronic press, pp 95-97, 1999.
4. Kalyviotis N. and Hu H., “A Co-operative Framework for Strategic Planning”, Proceedings Towards Intelligent Mobile Robots (TIMR) '01, Manchester, 2001.
5. Riley, P., and Manuela Veloso, M., “Towards Behavior Classification: A Case Study in Robotic Soccer”, In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), AAAI Press, Menlo Park, CA, 2000.
6. Ajdari Rad, A., “Design and implementation of a soccer coach; Dynamic leading of a team of intelligent agents”, BE project, Computer engineering faculty, Amirkabir University of Technology, Feb. 2001.
7. Qaragozlou, N., “Design and implementation of soccer player architecture; an architecture for multi agent systems, dynamic and real-time environments”, BE project, Computer engineering faculty, Amirkabir University of Technology, Feb. 2001.