Tạp chí Khoa học ĐHQGHN, Khoa học Tự nhiên và Công nghệ 23 (2007) 200-211

Dynamic coordination in RoboCup soccer simulation

Nguyen Duc Thien*, Nguyen Hoang Duong, Pham Duc Hai, Pham Ngoc Hung, Do Mai Huong, Nguyen Ngoc Hoa, Du Phuong Hanh

College of Technology, Vietnam National University, 144 Xuan Thuy, Hanoi, Vietnam

Received 15 August 2007

Abstract. The RoboCup Soccer Simulation is considered as a good application of the Multi-Agent Systems. By using the multi-agent approach, each team in this simulation is considered as a multi-agent system, which is coordinated each other and by a coach agent. Different strategies have been proposed in order to improve the efficiency of this agent. In this paper, we investigate firstly the coordination in several modern teams by identifying all of their disadvantages. We present then our approach related the dynamic coordination in order to improve the performance of our team. The experimentation and evaluation to validate this approach will be concluded in this paper.

Keywords: Multi-Agent Systems; Dynamic coordination; Coordination Graph.

1. Introduction

RoboCup Soccer Simulator is considered an effective instrument in both research and training on Multi-Agent Systems – MA in particular and in sector of Artificial Intelligence - AI. Proceeded from Robot Soccer World Cup, which is held annually with participation of namely world-known robotic research groups, a champion of RoboCup Soccer Simulation is parallely held in order to build and develop effective algorithm, considerate strategies as well as reasonable learning methods, etc directing to a supreme targets of *« building a robot football team which is capacble to defeat the world best football teams (with real players)»* [1].

For its importance to research and development of RoboCup Soccer Simulation, applications of Multi-agent System and Artificial Intelligence plays a more and more essential role. Coordination between team members, both players and coach agent, is own of key factors that brings success for robot soccer simulation team. According to information thanks to environment experience (such as positions of each player, position of ball, context of play ground, coach agent, etc.), each player (each agent) must collect and classify, then analyze, accordingly to coordinate with other fellow-agents in order to generate an effective action (attack/defende/pass/dribble/shoot and score)

In this paper, we focus first and foremost on dynamic coordination in such a soccer team. The concept "dynamic coordination" herein must be understood as a combination of

[*] Corresponding author. Tel.: 84-4-7547615.
E-mail: thiennd@vnu.edu.vn

traditional coordination techniques in the multi-agent system [2] and dynamic tackling strategies which shall be applied subject to current status of environment.

Remainding of this paper is composed of basic concepts of coordination in multi-agent system as shown in part 2. Following in-depth introduction to the multi-agent system of robocup soccer simulation in section 3.1, we shall specify the way to access of dynamic coordination in section 3.2 and experimentation results in section 4 as well. And the final section of this paper is for evaluation of any resulted related.

2. Coordination in Multi-agent System

Coordination is one among three important factors [1] during reaction process between agents in a Multi-agent System (MAS). According to difinition by M. Wooldridge [1], coordination between agents has close relationship with inter-dependencies among activities of agents. There are many different ways of accesses in carrying out coordination in a MAS, such as *Coordination through partial global planning, Coordination through joint intentions, Coordination by mutual modeling, Coordination by norms and social laws, etc* [1]. The typical method out of those listed above is namely based on Nash-equilibria.

The essential point of Nash-equilibria is that in case of large number of agents, it is very sophisticated and takes time to calculate and determine "balance" action for each agent [3]. For that reason, subdivision of acting space of agents to be analyzed becomes effective. Considering problem of robot soccer simulation, coordination between agents has an intimate relation with information collected from environment of simulated robot. Hence, coordination graph is proposed in order to intensify possibility between mutual coordination among agents and with coach agent. In this section, following 2.1 for explanation on this method, we shall also mention another method based max-plus algorithm in section 2.2.

2.1. Coordination graph and Variable Elimination

In a multi-agent system, each agent shall take indibidual action, of which results, however, are under influence of behavior of other agents. In such a multi-agent system with mutual coorperation between agents [4] (a simulated robot soccer team for instance), set **A** includes individual behaviors $\mathbf{A_i}$ of every agents $\mathbf{a_i}$ and creates a joint action satisfactory with optimization conditions of global ***pay-off function***.

During process of implemention, each agent must choose a reasonable individual action to optimize joint-action of the whole system (for instance, based on Nash-equilibria). However, number of joint-actions increases in accordance with exponential function and number of agents, and this causes the determination of balance statuses non-feasible in case of large number of agents.

For pupose of solutions to this problem, *coordination graph – CG* and *Variable Elimination - VE* are applied by Guestrin et al. [5] who consequently brought solutions to sophistication degree on process for

Definition : *Coordination graph (CG) G = (V, E) is a directional graph, of which each dot of V is an agent and a certain side of E is dependent to coorperation of two end-agents* [6].

[1] Three these factors are : cooperation, coordination and negotiation.

Naturally, at a certain point of time, only agents connected with others and shown on CG should be coordinated with those agents. For example, see Fig.1 below which demonstrates a CG with 4 agents. IN this example, A1 must coordinate with both A2 and A3 while A2 must coordinate with both A1, and A3 coordinates with A4 and A1, whereas A4 coordinates with A3.

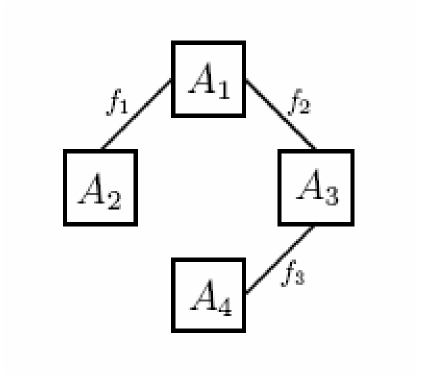

Fig. 1. Coordination graph of four agents.

Major idea of this access depends on the global pay-off function U(A) to be disintegrated into sum of global pay-off function which relates to some agents only. For purpose of determining optimal action for every agent, Variable Elimination is used by Guestrin in similar way with variable elimination in Bayesian network [1,2]. According to [6], this algorithm operates in two phases: eliminating variables and determining optimal actions as folows:

• Phase 1: Variable Elimination

▪ B1: Select agent, a_i, and determine pay-off functions u_j from all neighbor agents of a_i (neighbor agents - NA_i, is obviously determined through Coordination Graph).

▪ B2: optimize decision of a_i depending on action combinations available in set NA_i and transmit results to its neighbor agent a_j (belonging to NA_i).

▪ B3: Eliminate a_i out of Coordination Graph and repeat B1 till there is only one agent left in Coordination Graph. This agent shall select optimal action from sets of actions available for its.

• Phase 2: carried out in resverse sequence of agents according to phase 1. Each agent determines its optimal action based on actions determined by its neighbor agents before.

For further illustration of performance process of variable elimination, let's consider an example shown in Fig. 1 with four agents above. In this example, pay-off function of every joint-action of four agents shall be determined with function

$$U(a) = f_1(a_1, a_2) + f_2(a_1, a_3) + f_3(a_3, a_4) \quad (1)$$

(here, we consider a_i as action of agent A_i and a as joint-action of all agents)

Firstly, let's eliminate agent A_1. This agent depends on two functions f_1 and f_2 and maximum value of U(A) shall be determined through formula:

$$\max_{a} U(a) = \max_{a_2,a_3,a_4} \left\{ f_3(a_3,a_4) + \max_{a_1} \left[f_1(a_1,a_2) + f_2(a_1,a_3) \right] \right\} \quad (2)$$

From A_1, we have a new pay-off function $f_4(a_2, a_3) = \max \{ f_1(a_1, a_2) + f_2(a_1, a_3) \}$ in accordance with a_1. This is function that brings relevant value with its best-response in combination of any action available of a_2 and a_3 (signalized as $B_1(a_2, a_3)$). At that time, function f_4 is completely dependent from a_1 and a_1 is eliminated from graph.

Apply above-mentioned process to eliminate a_2, now there only left with f_4 depending on action of agent a_2 and replacing by function $f_5(a_3) = \max \{ f_4(a_2, a_3) \}$ in accordance with a_2. Next, we eliminate a_3 by replacing function f_3 and f_5 with function $f_6(a_4)$. Hence, max U(a) in accordance with $a = f_6(a_4)$ according to a_4. At this time, A_4 shall be the optimal action of a^*_4 itself.

After selecting action of A_4, optimal actions of remaining agents shall be carried out in reverse sequence. In this example, action of A_3 shall be determined through the best-response function related to a^*_4: $a^*_3 = B_3(a^*_4)$. Similarly, $a^*_2 = B_2(a^*_3)$ and $a^*_1 = B_1(a^*_2, a^*_3)$.

In any even of an agent having more than one best-response action, it shall randomly select one of them. This selection shall not affect joint-action because that selection shall inform its neighbor agents.

Effect of Variable Elimination algorithm does not depend on order of elimination, and always brings optimal joint-action of agents. Yet, performance time of this algorithm depends on the sequence of variable elimination and sophistication degree of exponential function for width of CG. Furthermore, this shall only take effect only when phase 2 completely finishes and that is why it is unreasonable for any MAS to deal with real-time, taking robot soccer simulation as an example (each player must determine its next action after every 100ms). Solutions to these weak points of CG and VE shall be mentioned in next part, based on Max-plus algorithm which was proposed by J. Kok and N. Vlassis in 2005 [7].

2.2. Max-plus Algorithm

Another very effective algorithm in improving the coordination between agents has been studied and successfully applied by UvA Trilearn for TriLearn 2005 Multi-agent System. This algorithm namely depends on CG, yet, despite of VE, [6] is used with max-plus algorithm thanks to which the main idea is to determine maximum a posteriori in non-directed graph.

Above-proposed method relies on sending again and again messages $\mu_{ij}(a_j)$, which is considered as optimal global pay-off function between two agents i and j in a side of CG [8,9]. This allows an approach an optimal action of each agent after every two certain repeat [6].

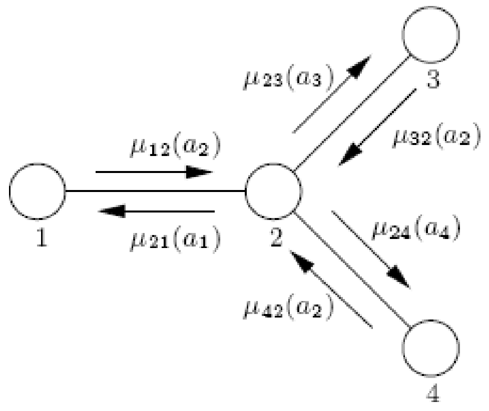

Fig. 2. Illustration of Max-Plus Algorithm.

Consider non-directed graph G = <V,E> of which, |V| is number of points, |E| is number of sides of graph. The global pay-off function U(A) is calculated as follows:

$$U(a) = \sum_{i \in V} f_i(a_i) + \sum_{(i,j) \in E} f_{ij}(a_i, a_j)$$

Of which, demonstrates costs for action a_i of A_i and f_{ij} as action mapping pay-off function (a_i,a_j) of two agents $i, j \in E$ near to a real number $f_{ij}(a_i,a_j)$, aiming at finding the best joint-action a* with (3) in maximum.

Each agent i is sending repetitively message μ_{ij} to its neighbor points $j \in \Gamma(i)$, of which μ_{ij} maps action a_j of agent j to a real number following formula:

$$\mu_{ij}(a_j) = \max_{a_i}\{f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i)\} + c_{ij} \quad (4)$$

Of which, $\Gamma(i) \setminus j$ is all neighbor points of agent i except agent j, and c_{ij} is normalized vector. This message can be understood as appropriate value of maximum pay-off value to which agent i reaches with any actions of agent j, and is calculated as grand sum (through actions of agent i) of pay-off functions f_i , f_{ij} and all messages sent to agent i except those sent from j. Messages are exchanged until they bring together again as $g_i(a_i) = f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ij}(a_i)$. At that time, every agent i shall select its optimal action as follows: $a_i^* = \arg\max_{a_i} g_i(a_i)$. If only one action reaches maximum for all agent i, optimal joint-action a*= arg $\max_a(U(a))$ is the unique with element a*= (a_i^*).

3. Multi-agent System for RoboCup Soccer Simulation

3.1. RCSS Competiton

RCSS (RoboCup Soccer Simulation) is a competition among RoboCup Soccer Simulation teams, of which each must establish a multi-agent system with every of its agents to be considered as a client, and performed in the same environment against a competitor [10]. The whole evnvironment of MAS is managed by a server (RoboCup Soccer Server - RCSS), which controls environment and is a element managing competition rules as well [6]. Correspondingly, RCSS has been coded and required compliance of all soccer teams. There is no limit in the way teams are built up, the only requirement is that instrument used to develop a team must be supported by a client-server through UDP/IP socket. Each client is a process independently linked to server with a separate portal. Rules applied for this competition is governed by FIRA[2] with 11 players in maximum.

[2] For further information, visit http://www.fira.net/

In the most recent time, at the competition held in May 2006 in Germany, the championship was won by WrightEagle of China University of Science and Technology, and followed by Brainstormers of Germany Osnabrueck University and Ri-one of Japan Ritsumeikan University[3].

3.2. Dynamic coordination of robot simulation agents

RoboCup Soccer Server provides a dynamic and discrete environment, which modelizes many real environmental factors such as movement noise, interference sensor, limited physical abilities and restrainted conmmuniations.

However, in RoboCup Soccer Server, there is only one agent, at every point of time, permissible to communicate with Server. Thence, agents must observe and store the environmental status inside. IN the event of no action coordination, a player moves to a position where he observes the ball's speed change. Before the ball's speed changes, that player has no notion he will practically receive the ball and thus does not coordinate with the passing-ball-player.

In order to finish coordination, all agents firstly are assigned with particular role in accordance with current context. Then, these roles shall be coordinated by applying Variable Elimination algorithm with value principles defined so as to make the best use of joint-action and environment variables. Below is detailed description how to use coordination graph to coordinate the ball-passer and ball-receiver, the first ball-receiver and second ball-receiver, i.e. who receives ball from the first ball-receiver.

[3] For further details, visit http://ssil.uni-koblenz.de/RC06/2D_Ranking.html

Goalkeeper, central defender, sweeper, wing defender, central midfielder, wing midfielder, wing attacker và central attacker

First of all, a Role Assignment Function shall be applied to assign roles for players: the defender, the ball-passer and the ball-receiver, etc. depending on infromation from environment. This role assignment can be calculated directly from information on current context. For instances, a player standing nearest to the ball can be assigned as a defender when it is impossible for him to kick the ball, and as a ball-passer when possible for him to kick it. All roles assigned to agents are arranged subject to position of ball. Any players, who have not been assigned with any role, are in passive state. This role assignment helps build up structure of Coordination Graph on which roles of saving, passing or receiving ball are linked together [11]. And this role assignment can be changed in conformity with the change of ambient environment.

At that time, all agents linked together must coordinate their actions. E.g. an agent can choose one of following actions:

- *PassTo (i, dir)*: pass the ball to a certain position with a fixed distance from agent *i* in direction of *dir*, D={center, n, nw, w, sw, s, se, e, ne}

- *MoveTo(dir)*: move in direction of *dir*

- *Dribble(dir):* bribble the ball in direction of *dir*

- *Score:* attempt to drive the ball into the contender's goal

- *ClearBall:* strongly kick the ball among the contender's players towards the contender.

- *MoveToStratPos:* move to strategic position of agent (according to situation of home team and current position of ball)

Existing rules can be directly modified or rubbed out, creating possibility to change strategies of team when playing against many different kinds of contenders.

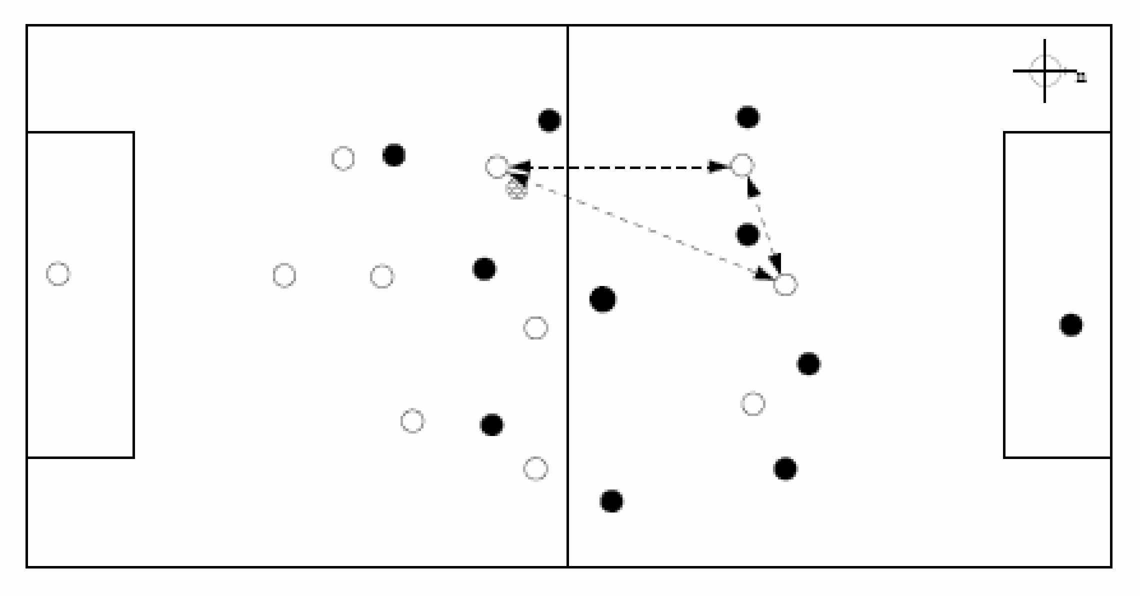

Fig. 3. Coordination graph based on roles of agents.

Those rules contain many ties of the environment that are described by state variable. In fig. 3, we will simplify coordination graph based on roles of agents if there are more environment variables which are suitable with following value rules (Supposed that there are only environment variables!isPassBlocked(1,2,s)and!isPassBlocked(2,3,nw) with value True):

A1: (p_1 ^{passer}; a_1 = passTo(2,s) ^ a =moveTo(s): 50)
(p_2 ^{passer}; a_1 = dribble(n) : 30)
(p_3 ^{passer}; $a_1$earBall : 10)
A2: (p_7 ^{receive}r; a_2 = moveToStratPos : 10)
A3: (p_6 ^{receiver}; a_1 = passTo(2, dir) ^ a_2 = moveTo(dir) ^ a_3 = moveTo(nw): 30)
(p_7 ^{receiver}; a_3 = moveToStratPos : 10)

By applying exception algorithm, each agent will be eliminated from the graph by value maximums that return locality (Pay-Off). In case agent A 1 is eliminated first, it will collect all value rules that contain a_1 and report its strategies to paternal agents:

(p1 ^{passer}; a2 = moveTo(s) ^ a3 –moveTo(nw) : 80)
(p1 ^{passer}; a2 = moveTo(s) ^ a3 = moveTo(nw): 50)
(p3 ^{passer}; a2 = !moveTo(s) : 30)

After agents A2 and A3 have fixed their strategies, agent G2 will implement the action *passTo(2,s)*, agent G2 will implement the action *moveTo(s)* to take ball, and agent G3 will implement the action *moveTo(nw)* to receive possible ball passed from agent G2. In case of unexpected possibilities such as the first ball passed from G1 to G2 fails, coordination graph will automatically be updated in conformation with new event.

Strategies of proposed dynamic coordination

UvA Trilearn Team is the championship one of simulation competition in 2003 and the source code has been proclaimed to help people for learning and research. In UvA Trilearn, 8 real players are defined: goalkeeper, central defender, sweeper, wing defender, central midfielder, wing midfielder, wing attacker và central attacker. Moreover, the source code of *UvA Trilearn team* is clearly and understandably written, and it has been used as the development basis of many international teams such as FC Portugal with championship in 2004. This is the main reason that group of authors select it as the development basis of the team.

Based on Trilearn 2003, we have proposed some strategies related to action of each agent/player. These strategies are to advance the effect of dynamic coordination between agents in team. From that, following actions can be earliest and most effectively identified.

a. Ball handling strategy

Ball handling strategy of each player in Trilearn 2003 is simply set up. Moreover, this strategy has some weak points as follow: If kickable, player will kick it (towards an accidentally kicking direction). Otherwise, considering whether that player is possible to approach the ball at the quickest time. If possible, let him approach the ball. Or else, let him move back to strategic position.

Our strategy aims at correcting above-mentioned weak points. Specifically, at the start of the match, player's formation will be arranged and player number 9 kicks off the ball at maximum speed (but in accidental direction) towards the goal. In the match, if players does not see the ball, it will be searched with *method searchBall*. In case the ball (under possession) s kickable, it will be kicked towards the goal. In case of no ball possessed, players will measure whether they are the quickest ones to approach the ball. If possible to approach the ball, do it. Otherwise, move the player back to strategic position, which is determined by current position of the player, role of the player (midfielder, defender, etc) and ball position.

b. Ball searching strategy

Unfortunately, Trilearn 2003 is just only simple ways of ball searching. Specifically in function *searchBall*, players' visual sector is default with half-space before their eyes. This provides a better observation to players. However, this is not substantially true to players. Each player has his own characteristic so-called *viewAngle*. I.e. each player only sees ball within cone domain under *viewAngle*. If not seeing the ball, they will move towards ball direction that they last see it. Following *searchBall* method, players will move in slanting direction of 60^0 from previous position. This is not totally optimal because in many cases players can see the ball with only a little change of *viewAngle*; what's more, it may be impossible for them to see the ball even *viewAngle* has been directed to 60^o angle.

Our improvements are to provide additional characteristics of players' *viewAngle*: visual sector of each player is cone angle called *viewAngle*. If the ball is out of cone angle, players can not see it. With new method, in order to see the ball, players will observe in different directions. Players will observe in a direction slanted with a small angle from the previous one. This observation will be done many times until players see the ball. Like this, players will observe in different directions until they see it.

c. Strategy of ball possession

Dribble method in Trilearn 2003 requires parameter called dribble direction, which is just calculated but unrealized yet. This means players can not dribble the ball in case rivals prevent it.

Accordingly, we suggest a strategy of possessing ball as follow: player will observe to specify rivals in front of him and the ball; then the ball possessor shall dribble the ball in a direction different from the rival's or dribble it fast across the rival. Such advanced dribble method helps define the opposite player. Therfore, dribble direction can be determined whether different from direction towards the opposite player, or the same the direction towards the opposite player but with *DRIBBLE_FAST*. Improved dribble method only weighs dribble direction. It recalls former function by providing parameter of this calculated dribble direction.

d. Strategy of ball passing

In Trilearn 2003, when the match starts, formation of players will be arranged, the player number 9 will kick off the ball at maximum speed (but in accidental direction towards the goal). During the match: in case player does not see the ball, it will be searched using *method searchBall*. When the ball (under possession) is kickable, it will be kicked in an accidental direction towards the goal. In case of no ball, player will measure his posibility to be fastest in approaching the ball. When the ball is approachable, do it. Otherwise, move the player back to strategic position, which is determined in accordance with formation of 433, 442, etc, current position of player and role of players (midfielder, defender,etc), ball position. Obviously, this is not optimal at all.

Passing strategy suggested is displayed as follow: If the match starts, formation of players will be done, the player number 9 will pass the ball to player number 7 who, in turn, will handle the ball (as his position is more favorable than player 9). During the match: In case player does not see the ball, it will be seeked for (with method *RearchBallApp* instead of method *searchBall*). In case players are tackling a free kick (the ball must be kicked or passed), the ball will be passed to the player in favourable position (with method *passBall*). If the ball is searched, it will be passed. Otherwise, the ball will be kicked towards the goal with maximum strength. In case the ball (under possession) can be kicked through a reasonably small distance from the player to the goal (smaller than 90% * (ability of the furthest shoot by this time is based on physical force of players)), the ball will be shot towards the goal. Defender, if seeing no rival player within a radius of 5 measurement units, decides to dribble ball. If facing against a rival, he will choose to clear the ball (using method *clearBall*). Non-defender without any rival player in front shall dribble the ball towards the goal. In other cases, the ball will be passed (to any fellow favorable to receive the ball) or strongly kick the ball ahead (if no player found able to receive the ball). In case of no ball: continue to approach the ball, if possible, or move to the strategic position.

e. Formation

The main formation information is the way to arrange and organize formation, including: Information about different formation methods' such as 433, 442,… (stored in the file *Fomations.conf*). The way of arranging players into formation (i.e. position – coordinate of players). The main procession in this module is method *getStrategicPosition* because it will determine strategic position to which player with no ball must move.

In Trilearn 2003, the method *getStrategicPosition* is applied in working out strategic position of players by taking *home position* of each player present in formation in combination with position of the ball that uses gravitational value. Strategic coordinate of players is calculated following formula: Current value of players + current value of the ball * gravitational value. If this co-ordinate is smaller than the minimum one or bigger than maximum one, it will be set to that minimum or maximum value.

Our improvements are proposed as follows: If player is right behind the ball, he then turns back to strategic position or ball position. Or else, strategic position will be calculated as follows: Position of players on strategic coordinate is calculated following formula: *current value of players + current value of the ball * gravitational value*. Next, check whether this value is out of touch-line (?) and assign it back to position at line if it is actually out of the touch-line. Concurrently, check if co-ordinate of player is behind the ball, and assign his coordinate at the same one of ball (in order to possess the ball).

4. Experimental result

Based on proposed dynamic coordination strategies, we have experimented by developing team called TrilearnA from source code of UvA Trilearn 2003. After setting up TrilearnA, we made an experiment by organizing matches between team TrilearnA and former UvA Trilearn 2003 and another match with team Brainstomer – the champion in 2005.

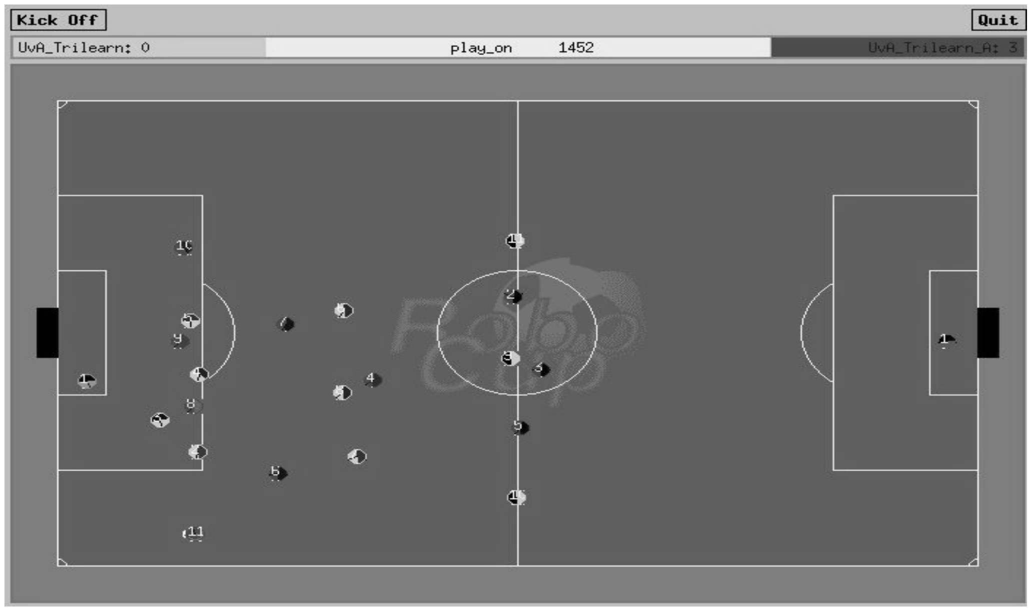

Fig. 4. Details of match between TrileanA (right) and Trilearn 2003 (left).

With experimental configuration of RAM 1GB, CPU 3.0 GHz system operating with a virtual machine Linux, results of matches between TrilearnA and two other teams are clear evidence for advantages of TrilearnA with advanced strategies, specifically:

- Three matches between TrilearnA and UvA Trilearn 2003, TrilearnA won completely with scores : 2-0, 3-1 and 3-0.

- Matches between TrilearnA and Brainstomer 2005, TrilearnA won 3 out of 4 matches with results 0-1, 1-0, 2-0 and 1-0 respectively.

Fig.5. Details of the match between TrileanA (left) and Brainstormer (right).

5. Conclusion

This paper focuses mainly on dynamic coordination problem in Multi-Agent System. From 2 approaching methods with application of coordination graph and max-plus algorithm, we have proposed improvements of dynamic coordination through connection with 5 strategies for particular context. Experimental results implemented with The RoboCup Soccer Simulation RCSS and open source code software UvA Trilearn 2003 have confirmed effects of our improvements. This is the basis for which group of authors intend to enhance further efficiency of the whole team in order to participate in matches of the Robot Cup Soccer Simulation 2008.

Acknowledgements

This paper is partially funded from theme coded 203006 under the program Basic Reseach on Natural Science and partly from theme coded QC.06.03, Hanoi National Univesity.

References

[1] *Objective of RoboCup, http://www.robocup.org /overview/22.html*

[2] Michael Wooldridge, *An Introduction to Multi Agent Systems,* John Wiley, Sons, 2002,

[3] J. Laumonier, B. Chaib-draa. *Multiagent Q-Learning: Preliminary Study on Dominance between the Nash and Stackelberg Equilibriums***.** In Proceedings of AAAI-2005 Workshop on Multiagent Learning, Pittsburgh, USA, July 10, 2005.

[4] P. Stone, M. Veloso, *Multiagent System: A Survey from a Machine Learning Perspective, Ferbuary, 1997.*

[5] C. Guestrin, S. Venkataraman, D. Koller, *Context-specific multiagent coordination and planning with factored MDPs.* In: Pr006Fc. 8th Nation. Conf. on Artificial Intelligence, Edmonton, Canada (2002)

[6] José M. Vidal, *Learning in Multiagent Systems: An Introduction from a Game-Theoretic Perspective.* In *Adaptive Agents: LNAI 2636*, Eduardo Alonso editor, Springer Verlag, 2003, p. 202-215.

[7] J. Laumonier, B. Chaib-draa, *Partial Local FriendQ Multiagent Learning: Application to Team Automobile Coordination Problem***.** In Proceedings of the 19th Canadian Conference on Artificial Intelligence (AI'2006), Quebec, Canada, June 7-9, 2006

[8] J.R. Kok, N. Vlassis, *Using the max-plus algorithm for multiagent decision making in coordination graphs.* In: RoboCup 2005: Robot SoccerWorld Cup IX, Osaka, Japan (2005)

[9] M. Wainwright, T. Jaakkola, A. Willsky, *Tree consistency and bounds on the performance of the max-product algorithm and its generalizations.* Statistics and Computing 14 (2004) p.143-166.

[10] K.I. Tsianos, *Algorithms for optimal coordination of multiagent systems and applications*, Diploma thesis, National Technical University of Athens, Greece, June 2005.

[11] J.R. Kok, M.T.J. Spaan, N. Vlassis, *Non-communicative multi-robot coordination in dynamic environments.* Robotics and Autonomous Systems 50 (2005) 99.

Phối hợp động trong hệ đa tác tử mô phỏng robot đá bóng

Nguyễn Đức Thiện, Nguyễn Hoàng Dương, Phạm Đức Hải, Phạm Ngọc Hưng, Đỗ Mai Hương, Nguyễn Ngọc Hoá, Dư Phương Hạnh

Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, 144 Xuân Thủy, Hà Nội, Việt Nam

Mô phỏng robot đá bóng (The RoboCup Soccer Simulation- RCSS) được xem là một ứng dụng hữu ích trong việc nghiên cứu các hệ đa tác tử (Multi-Agent Systems - MAS). Với cách tiếp cận đa tác tử, mỗi đội bóng được mô phỏng như một MAS, trong đó có sự phối hợp giữa các cầu thủ cũng như với huấn luyện viên của đội –tác tử giữ vai trò điều phối cả đội. Các chiến thuật khác nhau nhằm cải thiện hiệu quả vai trò của các agent. Bài báo này chú trọng đến quá trình phối hợp động trong một đội bóng mà trong đó quá trình phối hợp giữa các tác tử sẽ được thực hiện thông qua các chiến thuật được xác định theo từng ngữ cảnh thực tế. Kết quả thực nghiệm thu được cho phép đánh giá khả năng áp dụng tốt của cách tiếp cận phối hợp động đề xuất trong mô phỏng robot đá bóng.

Từ khóa: Hệ đa tác tử, phối hợp động, đồ thị phối hợp.