OxBlue2009 (2D) Team Description

Jie Ma

Oxford University Computing Laboratory,
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
jie.ma@comlab.ox.ac.uk
http://www.comlab.ox.ac.uk

Abstract. OxBlue2009 (2D) is a robot football team for RoboCup 2D simulation. In this paper, the decision structure of our team will be presented. Under this structure the fundamental cooperative behaviour *formation* is briefly presented. It is accompanied by a instance-based learning method that approximates opponent formation functions. As to the high-level cooperation, PSP (Policy Search Planning) that we proposed in 2008 is reviewed. Policy Search is used to find an optimal policy for selecting plans from a *plan pool*; it extends an existing gradient-search method (GPOMDP) to a MAS domain.

1 Introduction

OxBlue2009 (2D) is a robot football team for RoboCup 2D simulation and it derives from Apollo05(2D). Jie was the team leader of Apollo and lead the team won the champion of China RoboCup in 2004 and 2nd place of US RoboCup Open as well as 7th place of RoboCup in 2005. Since 2006 OxBlue has been developed in Oxford University computing laboratory and we were top 8 team in RoboCup 2008. OxBlue2009 is implemented in C++ and it is developed based on the released code of Helios2008 and the corresponding base library **librcsc**, which are developed by HELIOS team [1].

The main purpose of our development of OxBlue2009 is to verify and promote our research on multi-agent systems (MAS). In particular, as cooperative skills essentially differentiate MASs from single-agent intelligence, we are interested in applying Machine Learning methods to yield cooperative behaviours in MAS scenarios. HELIOS was ranked 3rd in RoboCup 2008 and with their decent released code we can concentrate our efforts on the cooperative learning scenarios rather than maintaining low-level models.

In §2, a layered decision architecture that is used in OxBlue2009 is presented. Under this structure the fundamental cooperative behaviour *formation* is briefly presented §3, and how to modeling opponents' formation functions using instance-based learning is also discussed. In regarding to the high-level cooperation, our novel method PSP (Policy Search Planning) and its applications in RoboCup are briefly reviewed in §4. It is followed by the conclusion in §5.

2 Decision Architecture

In order to reduce the learning space of cooperative skills, most of today's MASs tend to adopt *vertical layered architectures* [2, 3]. This architecture is also adopted in OxBlue2009. In RoboCup soccer, pure reaction is required on some occasions, such as in a corner kick situation, where most agents don't need to make complicated decisions but to move to stationary positions. In other words, before world models are fully generated, actions will be directly sent. In more sophisticated decisions, however, such as stopping the ball from losing it, then although world models have been created emergency actions will be directly sent to the executor without comparing different skills in the arbitrator. This is in-between decisions. In most cases, decisions are pure deliberation – local issues such as interception and dribbling can be solved in the individual skill module, while global problems including formation and team strategies can be dealt with by advanced methods such as planning. Our decision structure is given in Figure 1.

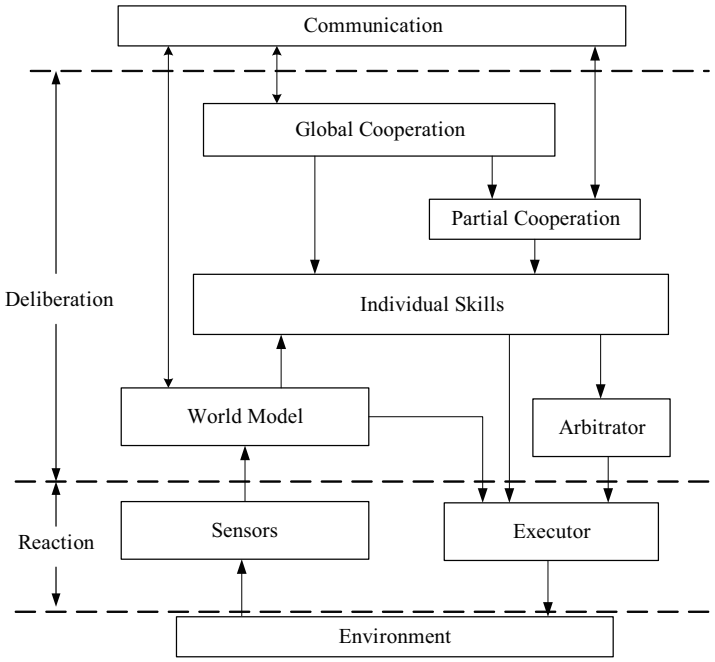

Fig. 1. Decision Architecture of OxBlue2009

3 Formation

In football games, *formation* is the term used to describe how players are positioned in the field. It plays a critical role both in human and robot football games.

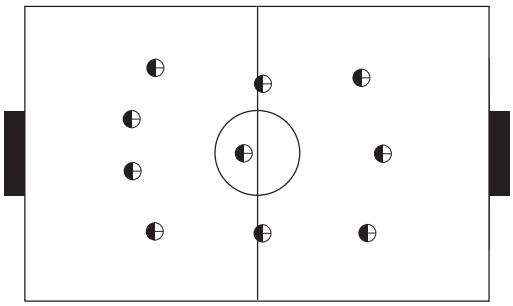

Fig. 2. 4-3-3 Formation

A Situation Based Strategic Positioning (SBSP) method has been proposed by a world champion team, FC Portugal 2000 [4]. Similar approaches have also been employed in many other successful teams including another world champion, UVA 2003 [5]. Recently Akiyama proposed a novel agent positioning mechanism using Delaunay Triangulation; due to the space limitation details can be found in [6]. In OxBlue2009 we use Delaunay Triangulation (which is implemented by **librcsc**) to generate our players' formation and use SBSP to approximate the formation of other teams.

Basically, SBSP establishes a ball-based function $Formation\,(\boldsymbol{P}_{ball})$ to calculate formation positions from a local perspective. Before a game, each agent is granted a unique role, which determines a strategic home position, an activity limitation and a ball attraction factor. During a game, dynamic formation points are calculated by simple vector operations. Every agent is just performing its own role without taking teammates into account, thus this cooperation is self-interested. Such cooperation is essentially embodied in predefined roles, which leads to distribution of agents in appropriate positions. The SBSP algorithm framework is shown in *Algorithm* 1.

3.1 Instance-based learning and Opponent Modelling

Since a team formation is highly dependent on the position of the ball, *instance-based learning* can be used in predicting the formation pattern of each opponent. Such learning is well known as *opponent modelling*, which is particularly useful in adversarial multi-agent systems, such as combat simulations [7] and the robot sheepdog [8, 9]. Learning opponent formation models is useful in RoboCup simulation because an agent's positions are only partially observable, so such models can help an agent to predict the positions of the other agents that cannot be seen.

Three-layered backward propagation networks (BPNs) are used to approximate individual agents' movements: two units in the input layer, which is the vector of the ball; two units that determine position in the output layer, plus a 3-unit hidden layer. All the units employ *sigmoid* activation functions. During a

Algorithm 1 SBSP algorithm

$\underline{Formation\,(\boldsymbol{P}_{ball})}$

$\boldsymbol{P} = \boldsymbol{P}_0 + \boldsymbol{r} \times \boldsymbol{P}_{ball}$
{$\boldsymbol{P}_0$ and $\boldsymbol{r}$ are pre-defined home position and ball-attraction factor respectively}
if possible_offside **then**
  $\boldsymbol{P}_{.x} = \boldsymbol{P}_{ball.x}$
end if
if $\boldsymbol{P}_{.x} > x.\max$ **then**
  $\boldsymbol{P}_{.x} = x.\max$ {the limitation of x}
end if
if $\boldsymbol{P}_{.x} < x.\min$ **then**
  $\boldsymbol{P}_{.x} = x.\min$ {the limitation of x}
end if
return $\boldsymbol{P}$

game, the coach builds 11 backward propagation networks for all the opponents, which are trained using online position information.

In our experiment, each opponent agent generates 1000 position samples per 100 seconds for *instance-based learning*(see *Fig.*3). The samples are collected in *play-on* mode only, and unexpected position changes (e.g. caused by **move** command) are excluded. After 8 epochs, all the 11 agent models converge, having average errors less than 0.048. Interestingly, with an average error of 0.0156, it is much easier to predict the position of opponent agent 1 (the goalie). The reason is clear: the goalie stays only in a very small area during the game.

The overall average error is also calculated: after 8 epochs, it converges to 0.036. This is a normalised value, which equals to around 5 metres in the real field. The result is satisfactory: according to the noise model, even the information from sensors will contain noise errors equivalent to 1 metre. The experiment is processed using our legacy OxBlueCoach08 and currently there still remains some work to do to embed it into OxBlueCoach09 because the current *librcsc* only supports single-thread computation, but hopefully we can use this technique in RoboCup 2009 in Graz. We have to admit that the learning result may be not precise for every cycle but it can be very helpful to predict the opponents' position and identify unnumbered opponent players.

4 Policy Search Planning (PSP)

In complex MASs, particularly in a system with hybrid individual architectures, planning plays a different role compared with that in traditional domains. In a simplified single-agent system, planning is used to directly find a goal. In dynamic MASs, however, the goal is usually difficult to achieve, or sometimes it is difficult to describe the goal. In addition, the traditional action effects will lose their original meaning: environmental state can also be changed by other agents at the same time, or sometimes it continually varies even without any actions.

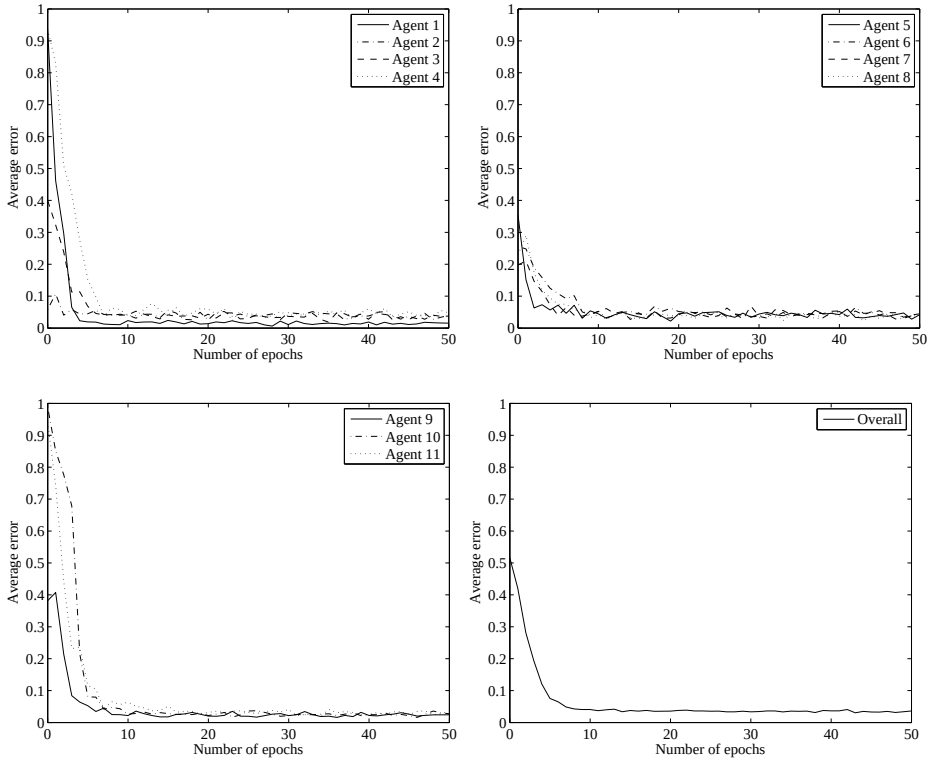

Fig. 3. Opponent Formation Learning Process

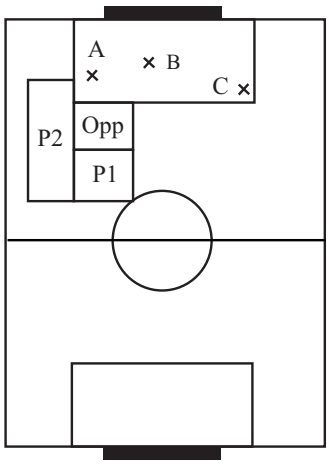

Fig. 4. A Planning scenario in RoboCup

For example, consider a scenario from RoboCup as shown in Figure 4: $P1$ and $P2$ are two team members with $P1$ controlling ball, *Opp* is an opponent, and they are all located in different areas. A traditional planner might construct a plan in which $P2$ dashes to point A and then $P1$ passes the ball. However, in this situation, points B and C are also potential target points for $P2$. Even from a human's perspective it is difficult to say which plan is better before fully knowing the opponent's strategies. Therefore, in multi-agent systems planning tends to be regarded as a "tutor" to increase cooperative behaviours so as to improve overall system performance. Expert knowledge can be embodied in such planning, without which agents mainly execute individual skills.

We propose a novel method called Policy Search Planning (PSP) for POMDPs, which is essentially a centralised plan-

ner for distributed actions. In the example of Figure 4, PSP can try to find the most appropriate policy for selecting a plan even without the opponent's model. Specifically, it can represent a number of complex cooperative tactics in the form of plans. Plans are shared by all the agents in advance, and policy search is used to find the optimal policy in choosing these plans. As a plan is not designed to find the goal directly but to define cooperative knowledge, the style of it is not very critical.

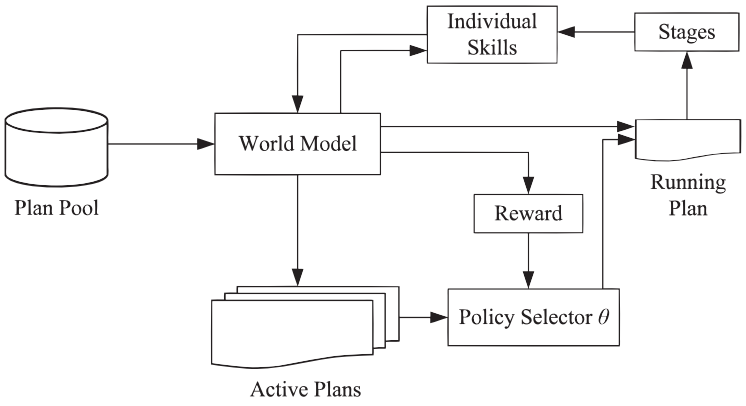

Fig. 5. Learning Process in PSP algorithm

In the PSP algorithm, a plan is actually a cooperative strategy. We can define plenty of offline plans to establish a *plan pool*, which is essentially an expert knowledge database. If the external state satisfies the precondition of a plan, the plan will be called an *active plan*. At time t, if there is only one active plan, it will be marked as the *running plan* and actions will be executed stage by stage. However, along with the growth of the plan pool, multiple active plans may appear at the same time.

Previous solutions chose a plan randomly, which is clearly a decision without intelligence. Q-learning is apparently a wiser approach, but unfortunately Q-learning is difficult to adopt in generalised decision architectures because all the plans cannot guarantee activation. In this paper we employ another reinforcement learning method, policy search, to overcome this difficulty. The learning framework is illustrated in Figure 5. Due to the space limitation, we are unable to extend the details of PSP method, more details can be found in [12].

The performance comparison of OxBlue2009 with and without PSP is shown in *Fig.*4. The average goal difference(AGD) is used as the quantitative performance criteria and the each statistical result is measured based on 10 games. We use the released binaries of RobCup 2008 teams as our opponents and employ about 50 plans in the plan pool and these plans have been refined by the PSP based on OxBlue2007. The *Fig.* 4 shows PSP increases the performance

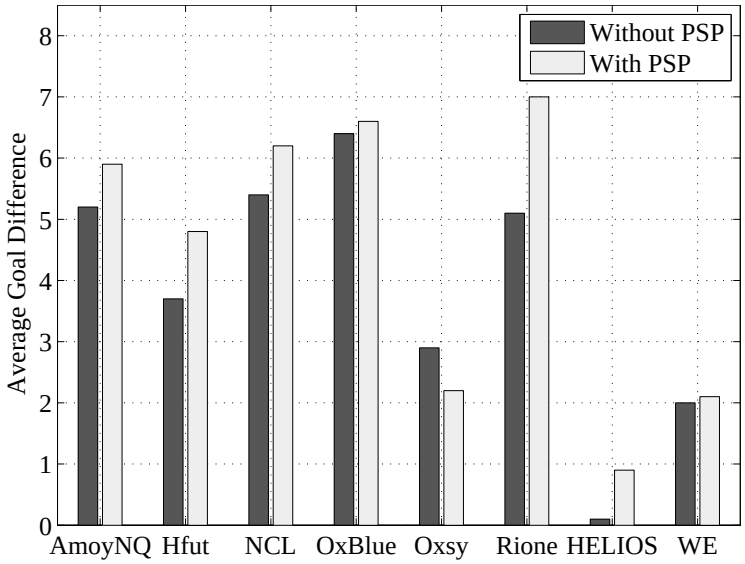

Fig. 6. Performance Comparison Of OxBlue2009 with and without PSP

of OxBlue2009 for most teams. Under Ri-one the AGD explicitly increases by about 2 goals; and for AmoyNQ, Hfut and NCL, the PSP contributes about 1 goal increase for each game on average. However for Oxsy, PSP will lead to the performance decrease. It is mainly because the plan pool does not contains the particular plans targeting this team and the plans were not learned based on Oxsy. In the next section, we will further discuss this problem and introduce our future work to solve this problem.

5 Conclusion and Future Work

In this paper, we briefly reviewed the roadmap and the previous achievement of OxBlue2009. We employed a layered decision architecture in OxBlue2009, under which we reviewed the currently most common formation algorithm SBSP. An instance-based learning is proposed to approximate opponent formation functions. Also, a novel method called PSP are also proposed in a generalised POMDP scenario, in which a large selection of cooperative skills can be presented in a plan pool; and policy search is used to find the optimal policy to select among these plans.

Though PSP can arguably increase the performance, its main limitation is the training time — according to our previous research the learning costs about 20–30 hours to converge for a particular team; and when a team changed its strategies the learning result is obscure. Furthermore it is also time-consuming to generate a well-designed plan pool for each team. In order to solve this problem,

we are currently working on a more targeting planning method. We detect the opponent strategies and use PSP to learn to conquer each strategy rather than the overall team. We are currently working on this method and hopefully we can use it in RoboCup 2009.

References

[1] Akiyama, H., Shimora, H., Noda, I.: Helios2008 team description. In: 12th RoboCup International Symposium. (July 2008) (CD Supplement).

[2] Perraju, T.S.: Multi agent architectures for high assurance systems. In: American Control Conference. Volume 5., San Diego, CA, USA (1999) 3154–3157

[3] Stone, P., Veloso, M.: Layered learning and flexible teamwork in robocup simulation agents. In: RoboCup-99: Robot Soccer World Cup III. (2000) 65–72

[4] Luis Paulo Reis, N.L.: Fc portugal team description: Robocup 2000 simulation league champion. Robocup 2000: Robot Soccer World Cup IV (2000)

[5] de Boer, R., Kok, J.: The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. PhD thesis (2002)

[6] Akiyama, H., Noda, I.: Multi-agent positioning mechanism in the dynamic environment. In Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup 2007: Robot Soccer World Cup XI, Lecture Notes in Artificial Intelligence. Volume 5001., Springer (2008) 377–384

[7] Yang, A., Abbass, H.A., Sarker, R.: Landscape dynamics in multi-agent simulation combat systems. In: AI 2004: Advances in Artificial Intelligence. (2004) 39–50

[8] Vaughan, R., Sumpter, N., Frost, A., Cameron, S.: Robot sheepdog project achieves automatic ock control. International Conference on Simulation of Adaptive Behaviour (1998)

[9] Vaughan, R., Sumpter, N., Henderson, J., Frost, A., Cameron, S.: Experiments in automatic flock control. Journal of Robotics and Autonomous Systems **31** (2000) 109–117

[10] Obst, O.: Using a planner for coordination of multiagent team behavior. Programming Multi-Agent Systems **3862/2006** (2006) 90–100

[11] Obst, O., Boedecker, J.: Flexible coordination of multiagent team behavior using htn planning. In: RoboCup 2005: Robot Soccer World Cup IX. (2006) 521–528

[12] Ma, J., Cameron, S.: Combining policy search with planning in multi-agent cooperation. In: Proceedings of RoboCup 2008: Robot Soccer World Cup XII, Lecture Notes in Artificial Intelligence, Springer (2009) to appear.