Combining Policy Search with Planning in Multi-agent Cooperation

Jie Ma and Stephen Cameron

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

Abstract. It is cooperation that essentially differentiates multi-agent systems (MASs) from single-agent intelligence. In realistic MAS applications such as RoboCup, repeated work has shown that traditional machine learning (ML) approaches have difficulty mapping directly from cooperative behaviours to actuator outputs. To overcome this problem, vertical layered architectures are commonly used to break cooperation down into behavioural layers; ML has then been used to generate different low-level skills, and a planning mechanism added to create high-level cooperation. We propose a novel method called *Policy Search Planning* (PSP), in which Policy Search is used to find an optimal policy for selecting plans from a *plan pool*. PSP extends an existing gradient-search method (GPOMDP) to a MAS domain. We demonstrate how PSP can be used in RoboCup Simulation, and our experimental results reveal robustness, adaptivity, and outperformance over other methods.

Key words: Policy Search, Planning, Machine Learning, Multi-agent Systems

1 Introduction

Cooperation is one of the most significant characteristics of multi-agent systems (MASs). Compared with a single agent, cooperating agents may gain over autonomous agents in *efficiency*, *robustness*, *extensibility* and *cost*. Sometimes cooperation is necessary to achieve goals due to the observation or action limitations of a single agent.

In order to reduce the learning space of cooperative skills, most of today's MASs tend to adopt *vertical layered architectures* [1, 2]. Such structures can arguably balance decision accuracy and speed, and simplify the learning process for high-level (deliberative) skills — such as cooperation — by decomposing them into lower-level skills. Those lower-level skills can be further decomposed until the lowest-layer reaction skills.

Although machine learning approaches have shown advantages in solving low-level skills [3, 4, 5], there still remain two difficulties in learning cooperation. Firstly, when the number of agents increases, state and action spaces become too

large for current machine learning approaches to converge. In the MAS domain, previous machine learning methods showed very limited cooperation and most of that cooperation was demonstrated in a stationary environment with only 2 players. Secondly, in an adversarial MAS application such as RoboCup and Combat Simulation, cooperation is usually very complex and highly related to its opponent, and repeated work has shown that it is difficult to yield such cooperation directly through machine learning [6, 2]. Although planning methods have arguably represented such cooperation, few of these methods have shown successful generalised adaption.

We propose a novel method called Policy Search Planning (PSP) which combines machine learning with a symbolic planner. We claim it can increase cooperation quality in POMDPs (Partially Observable Markov Decision Processes). Plenty of human knowledge on complex cooperation can be presented in a symbolic planner that allocates subtasks to appropriate agents; policy search is then used to find an optimal cooperation pattern in an unknown environment.

To evaluate our method, we employed RoboCup Soccer 2D Simulation as our test-bed, which is essentially a dynamic adversarial MAS environment. In this domain, a number of low-level individual skills such as *Positioning* [4], *Interception* [3], and *Dribbling* [5] were addressed and solved by machine learning approaches. Under high-level cooperation however, although some attempts have been made to present and decompose cooperation using planning, few of them have demonstrated how to learn to select plans (cooperative tactics) to maximise overall performance in an unknown environment. Our experimental results show adaptive cooperation among 11 agents and a significant improvement in performance over pure planning methods.

In §2 we discuss previous work, and we then present our PSP method in §3, where we consider a MAS as a generalised POMDP. The details of PSP in our test-bed RoboCup 2D Simulation are provided in §4, and we conclude in §5.

2 Related Work

Accompanied by the booming research on MAS, increasing interest has been shown in extending machine learning (ML) to the multi-agent domain. There are mainly three kinds of ML: *supervised*, *unsupervised* and *reinforcement learning*. In terms of learning cooperation, reinforcement learning is the most appropriate one among the above three because the mappings from cooperative actions to a global goal are usually obscure thus supervised and unsupervised learning is not suitable for yielding cooperative strategies.

Under reinforcement learning (RL), Q-Leaning is the most common learning approach in MAS scenarios. Recently however, there has been an increasing interest in another method of reinforcement learning, namely policy search. As a reasonable alternative to Q-learning, it can arguably promote performance in POMDP (as shown below).

In this section, we review how policy search was used in single-agent and multi-agent applications. Moreover, how reinforcement learning can be used to promote or simplify cooperative *planning* is also reviewed.

2.1 Policy Search

In a large POMDP problem, such as RoboCup, traditional Q-Learning sometimes has difficulty in approximating the Q-functions [7]. Especially in the MAS domain, since agents have to broadcast their local Q-function to the other agents, Q-Learning is significantly restricted by storage space and communication bandwidth. In recent years, Policy Search that directly finds the optimal policy is stepping into the spotlight.

Since the policy is usually parameterised, the optimal policy can be found by searching the parameter space θ, and using gradient-based search (also known as *Policy Gradient*) can substantially increase the search speed. Under policy gradient, the Boltzmann [1] distribution is widely used for computational simplicity.

In a single-agent POMDP system, as the state space cannot always be completely observed, *biased estimation* using an eligibility trace has been proposed by Kimura et al [9]. Baxter and Bartlett further suggested a GPOMDP (Gradient of Partially Observable Markov Decision Process) method, which was proved to converge under some assumptions [10, 11]. GPOMDP is essentially a biased and estimated gradient using the *eligibility trace* method [12]. The learning equations are given as follows:

$$\begin{array}{l} z_{t+1}\left(\gamma\right) \leftarrow \gamma z_t\left(\gamma\right) + \frac{\nabla \mu_{U_t}(\theta, Y_t)}{\mu_{U_t}(\theta, Y_t)} \\ \Delta_{t+1} = \Delta_t + \frac{1}{t+1}[R(U_t) z_{t+1} - \Delta_t] \end{array} \tag{1}$$

where z_t is an eligibility trace and Δ_t is a gradient estimate. For each observation Y_t, control U_t and its reward $R(U_t)$, $\mu_{U_t}(\theta, Y_t)$ represents the probability distribution with parameter θ at time t. $\gamma \in [0, 1)$ is a discount factor, where γ close to 1 yields a smaller bias but a larger variance. The contribution of GPOMDP is that the action transition probabilities and the probability distribution over the observation space are not necessarily required. Due to the space limitation here, the details of generalised GPOMDP algorithm can be found in [10, 11].

In the MAS domain, on the other hand, only a little work has been done using policy search. Tao et al suggested a possible way to adopt GPOMDP in network routing [13]. Routers are regarded as agents, each of which has a set of local parameters $\theta_i = \left(w_1^i, w_2^i, \cdots, w_m^i\right)$, and each parameter controls an action. Essentially, every agent learns its local parameters θ_i from a local perspective with the global rewards $R\left(\overrightarrow{U_t}\right)$. This algorithm has also been employed in a simple cube-pushing game in recent research [14]. Experimental results supported the performance of GPOMDP. Although this method has shown strong robust performance in some circumstances, it has two particular limitations: the action

[1] Essentially the same as Softmax and Gibbs distributions [8].

space in previous experiments was small, and the cooperation among agents was not complex.

Another relatively more complex MAS application has been demonstrated by Peshkin et al in [6]. Their algorithm essentially extends the REINFORCE algorithm [15] to an MAS domain. It can guarantee convergence to local optimality in a parameterised policy space. This algorithm has been adopted in a simple football game, and experiments have demonstrated outperformance over Q-learning in a partially observable environment. However, when the agent population grows the state space will become too large to be practical.

2.2 Combining Reinforcement Learning with Planning

Planning enables an agent to automatically achieve a goal by searching a set of actions. It is a significant way to undertake deliberative reasoning. In single-agent systems, there exist some mature planning methods, such as STRIPS (Stanford Research Institute Problem Solver), ADL (Action Description Language), HTN (Hierarchical Task Network) and PDDL (Planning Domain Definition Language).

The usages of planning in MASs are different from those in single-agent systems. In a single-agent system, planning is mainly used to find an action sequence to directly achieve a goal. In the MAS domain, however, planning tends to generate advanced cooperation. According to the taxonomy of Marinova [16], there are mainly three types of multi-agent planning: *centralised planning for distributed actions*, *distributed planning for centralised actions* and *distributed planning for distributed actions*. Today, most planning approaches in MASs are of the type centralised for distributed actions.

Pecora and Cesta proposed a hierarchical structure to apply PDDL planning [17] to MASs. The HTN method has been employed in MASs by Obst and Boedecker [18, 19]. Their method can arguably promote expert knowledge in dynamic POMDPs and speed up the planning process due to its hierarchical planning structure, though the role mappings in this method are rather stationary. To extend the flexibility of role mapping, Fraser and Wotawa presented a possible way to apply traditional STRIPS to MAS domains [20]. Before a plan starts, an agent can select its role, and broadcast it to others.

In the planning process, RL has been adopted in two ways. Firstly, RL is used to learn advanced individual skills, and so planners are able to search these skills instead of the actuators at the lowest-level. Secondly, some attempts have been made to directly promote planning decisions by using RL.

Using the first combination, recent work of Grounds and Kudenko [21] supports this approach using an example problem of an agent navigating through a grid. In a single agent grid square, Q-learning is used to generate low-level behaviours (choosing the direction of motion), while a STRIPS-based planner encodes high-level knowledge. Experimental results reveal that their PLANQ method performs better than pure Q-learning in a small domain, but on the other hand it will lose its strength when the state space grows.

Strens and Windelinckx employed Q-learning in a multi-robot task allocation problem [22]. Experimental result showed significant energy saving compared with a greedy method. Their main contribution is extending the action space to plan space. However, there are two limitations of this algorithm. Firstly, at any state, the active plans have to be the same for all the agents. Secondly, this algorithm requires a pure planning decision structure (pure deliberation architecture). Therefore, the algorithm is difficult to apply to generalised MASs.

Recently, Buffet and Aberdeen proposed have a planner called Factored Policy Gradient (FPG) [23, 24, 8]. Their algorithm combines a single-agent planner (PDDL) with the aforementioned policy search method, GPOMDP. Although the authors did not mention this, its potential advantage is that a pure deliberative architecture is not always needed. But the drawback is that all the plans have to share the same action space.

3 Policy Search Planning (PSP)

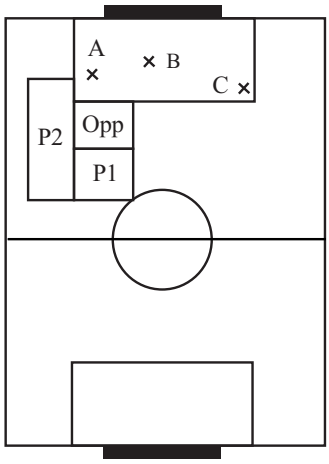

Fig. 1. A Planning scenario in RoboCup

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. For example, consider a scenario from RoboCup as shown in Figure 1: $P1$ 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 planner for distributed actions. In the example of Figure 1, 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. One possible presentation, a PDDL-like planner, is shown in Figure 2.

Compared with original PDDL, *:goal* will not be included as PSP aims not to achieve it directly, and *:effect* is not needed unless it is used for parameters in policy search. The concept of *stage* is introduced, which makes complex cooperation possible, whereby if and only if the success condition of the current stage is met a planner moves to the next stage; and *role mapping* formulae are introduced to find the most appropriate agents to implement actions

```
(define (PLAN_NAME)
   (:plan_precondition CONDITION_FORMULA)
   ((:agentnumber INTEGER(N))
    (ROLE_MAPING_FORMULA(1))
    (ROLE_MAPING_FORMULA(2))
     ...
    (ROLE_MAPING_FORMULA(N)))
   ((:stagenumber INTEGER(M))
    ((:stage_1_precondition   CONDITION_FORMULA)
     (:stage_1_success        CONDITION_FORMULA)
     (:stage_1_failure        CONDITION_FORMULA)
     (:stage_1_else           CONDITION_FORMULA)
     (:action1                ACTION_FORMULA)
     (:action2                ACTION_FORMULA)...)
    ((:stage_2_precondition   CONDITION_FORMULA)
     ...)   ...
    ((:stage_M_precondition   CONDITION_FORMULA)
     ...))
  [(:effect EFFECT_FORMULA)])
```

Fig. 2. A PDDL-like Plan Structure in PSP

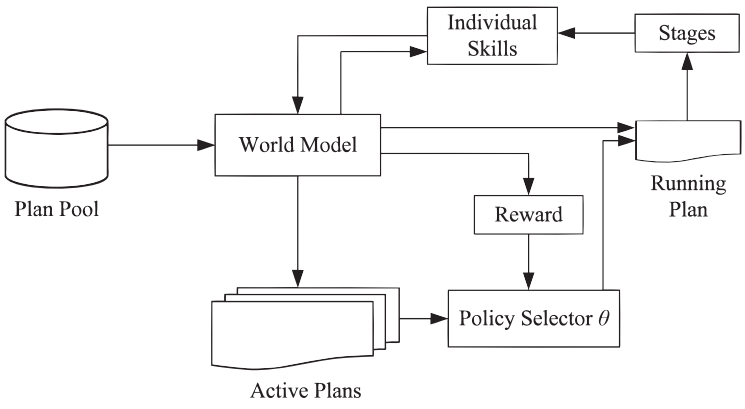

Fig. 3. Learning Process in PSP algorithm

In the PSP algorithm, a plan is actually a cooperative strategy. We can define plenty of offline plans by hand 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 [18, 19] 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 3. Say there exist n agents, which are organised in a hybrid architecture. From a global perspective, although all the actions are executed by the lowest level, decisions come from two different directions: individual skills and cooperative planning. Then the global policy value that can be evaluated by the accumulated reward can be represented as:

$$\begin{aligned}\rho\left(\theta, \theta^1, \theta^2 \cdots \theta^n\right) &= \sum_{s\in S, \phi\in\varphi} R_{\phi,\overrightarrow{a}} P\left(\theta, \phi\right) \\ &= \sum_{s\in S, a^i\in A^i, i\in[1..n]} R\left(\overrightarrow{a}\right) P\left(\theta^i, a^i\right) + \sum_{s\in S, \phi\in\varphi} R\left(\phi\right) P\left(\theta, \phi\right)\end{aligned} \tag{2}$$

where: φ is the action plan pool in state s; $P\left(\theta, \phi\right)$ is the probability distribution of selecting plan ϕ under plan parameter vector θ; $\theta^1, \theta^2 \cdots \theta^n$ are policy parameter vectors of each agent respectively; $\sum_{s\in S, a^i\in A^i, i\in[1..n]} R\left(\overrightarrow{a}\right) P\left(\theta^i, a^i\right)$ is the accumulated reward of individual skills; and $\sum_{s\in S, \phi\in\varphi} R\left(\phi\right) P\left(\theta, \phi\right)$ is the accumulated reward of cooperative planning.

In hybrid architectures, however, we cannot distinguish where the reward come from, thus along with three assumptions of GPOMDP [10, 11], two additional assumptions need to be satisfied:

Assumption 1. *Individual policies $\theta^1, \theta^2 \cdots \theta^n$ are independent of planning policy θ.*

Assumption 2. *During the observation of policy value $\rho\left(\theta\right)$ over active plan pool φ, probability distributions over individual actions under local policies $P\left(\theta^i, a^i\right)$ are stationary and individual actions yield a stationary accumulated reward: $\sum_{s\in S, a^i\in A^i, i\in[1..n]} R\left(\overrightarrow{a}\right) P\left(\theta^i, a^i\right) = C$*

Under the above two assumptions, the global policy value (Equation 2) can be rewriten as:

$$\rho\left(\theta, \theta^1, \theta^2 \cdots \theta^n\right) = \sum_{s \in S, \phi \in \varphi} R_{\phi, \overrightarrow{a}} P\left(\theta, \phi\right) = C + \sum_{s \in S, \phi \in \varphi} R\left(\phi\right) P\left(\theta, \phi\right) \quad (3)$$

From the equation above, we find that under individual policies $\theta^1, \theta^2 \cdots \theta^n$ the accumulated reward of the individual skills is independent of the global policy value $\rho\left(\theta, \theta^1, \theta^2 \cdots \theta^n\right)$. In other words, under our assumptions the global policy value is only determined by the planning policy value, and thus we can directly use the global policy value to evaluate the planning policy value:

$$\rho\left(\theta\right)' = \rho\left(\theta, \theta^1, \theta^2 \cdots \theta^n\right) = \sum_{s \in S, \phi \in \varphi} R_{\phi, \overrightarrow{a}} P\left(\theta, \phi\right) \quad (4)$$

Therefore, we can extend the GPOMDP (Equation 1) to a MAS planning domain. Agents adjust the planning parameters independently without any explicit communication. The adjustment is based on the local observation and a shared planning control. For an agent i, the PSP learning equations are as follows:

$$\begin{aligned} z_{t+1}^i\left(\gamma\right) &\leftarrow \gamma z_t^i\left(\gamma\right) + \frac{\nabla \mu_{\overrightarrow{U_t}}\left(\theta, Y_t^i\right)}{\mu_{\overrightarrow{U_t}}\left(\theta, Y_t^i\right)} \\ \Delta_{t+1}^i &= \Delta_t^i + \frac{1}{t+1}[R(\overrightarrow{U_t}) z_{t+1}^i - \Delta_t^i] \end{aligned} \quad (5)$$

where θ is a planning parameter; and z_t^i and Δ_t^i are the eligibility trace and the gradient estimate for the agent i respectively at time t. For each local observation Y_t^i, global planning control $\overrightarrow{U_t}$ and its global reward $R(\overrightarrow{U_t})$, $\mu_{\overrightarrow{U_t}}\left(\theta, Y_t^i\right)$ represents the probability distribution with the planning parameter at time t.

4 Application in RoboCup

RoboCup simulation is a suitable application to evaluate the PSP algorithm because of the following three reasons: firstly, humans have knowledge about football and need to apply it to robots; secondly, intelligent cooperation is useful; and thirdly, there exist no universal optimal policy and so adaptive and adversarial strategies are required. In order to show universality, our plans were constructed in a RoboCup coach language, *CLang* [25]; however, due to space limitations, we are unable to show the *CLang* form of the plan here.

We created two opponent teams *OppA* and *OppB* which tend to defend from side and centre respectively. All the plans have three features, which are the extents to which a plan will change the ball or player positions towards the left sideline, right sideline, and the opponent goal respectively. They are calculated by vector operations, and selected under a policy π. The policy π is parameterised by a vector of weights $\theta = (w_1, w_2, w_3)$. The probability for selecting plan ϕ with parameter vector θ is taken from a Boltzmann distribution.

Our experiments consist of three parts. In our first experiment, 30 plans were defined in the plan pool, and agents play against *OppA*. In order to verify the

robustness of PSP, an additional 15 plans were defined in our second experiment under the legacy policy from the first experiment. *OppB* is also used to test adaptivity in our third experiment.

During the PSP learning, we used a discount factor of $\gamma = 0.95$; when agents score a goal $R_{\phi,\overrightarrow{a}} = 1$; and the average number of goals per 100 seconds were calculated every 15 minutes. The PSP learning processes lasted for 20 hours, and was compared with a non-planning method and a planning without learning method. Experimental results are shown in Figures 4–7.

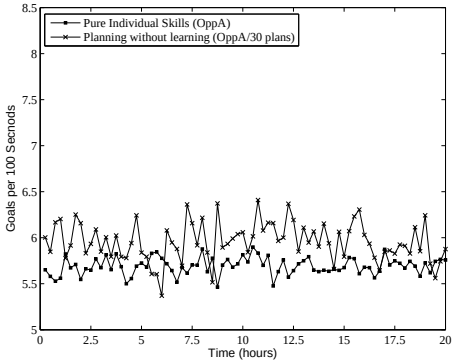

Fig. 4. Performance of pure individual skills and planning without learning.

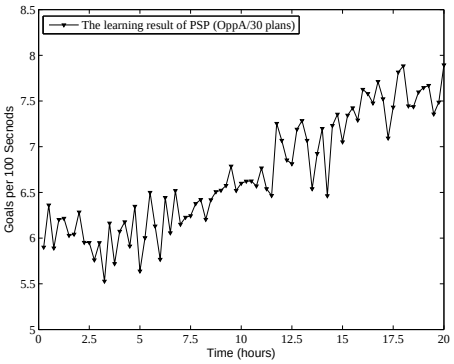

Fig. 5. The learning result of PSP (OppA/30 plans).

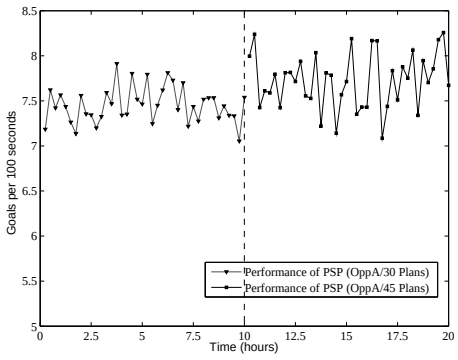

Fig. 6. Performance of PSP (OppA/30 and 45 plans).

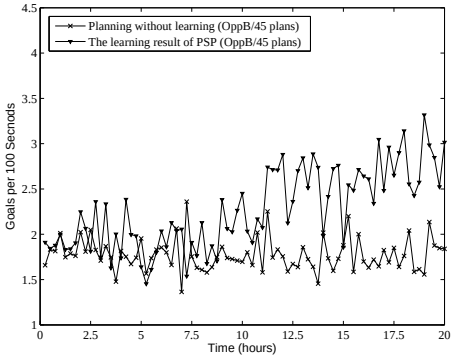

Fig. 7. The learning result of PSP (OppB/45 plans).

Figures 4 and 5 show that multi-agent planning causes a larger fluctuation in the average reward, but can promote overall performance. It supports previous research on MAS planning [26, 20, 19, 18]. However, randomly selecting active

plans without learning leads to only a slight performance improvement in our first experiment. In the first 4 hours, learning speed was very slow; the performance difference between learning and non-learning is not very clear. It is followed by a steady reward increase from around 6 to more than 7.5 goals per 100 seconds. The figures illustrate that PSP leads to a notable increase in system performance compared with non-learning planning and non-planning methods.

The result of our second experiment (Figure 6) shows slightly better performance with 15 additional plans using the legacy planning policy. It suggests that PSP is reasonably robust — with the same opponent, legacy policy is also compatible with new plans that can lead to further improvement of the decision quality.

Although the legacy policy may not be useful when playing against a completely new opponent, Figure 7 shows that through reinforcement learning agents will finally find a policy to beat their new opponent, namely *OppB* which has a stronger defence strategy. Without learning, pure planning can only make fewer than 2 goals per 100 seconds; after about 8 hours' learning, PSP increases the average goal rate to just below 3. Therefore, adaptive cooperations are established without knowing the opponent model, which supports our theoretical analysis.

5 Conclusion and Future Work

Cooperative behaviours has long been regarded as one of the most important features of MASs. The main difficulty in establishing cooperation using machine learning is the large learning space. Today, layered learning frameworks are widely used in realistic MAS applications. Under this architecture, ML is used to establish low-level skills while planning is used to define high-level tactics.

Under a layered decision architecture, we proposed a novel method called PSP 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. The innovations of our PSP method include:

1. enabling agents to learn to find the most efficient cooperation pattern in an unknown environment;
2. the design of a learning framework to yield robust and adaptive cooperation among multiple agents; and
3. extending GPOMDP to a MAS domain where complex cooperation is useful.

We demonstrated why and how PSP can be used in RoboCup 2D Simulation, and experimental results show explicit robustness, adaptivity and outperformance over non-learning planning and non-planning methods. In our more general (unquantified) experience with PSP it appears able to find solutions for problems that cannot be solved in a sensible timescale using earlier methods.

PSP is our first attempt to learn the optimal cooperation pattern amongst multiple agents. Our future directions are two-fold. Under RoboCup we are planning to define more plans and more features for PSP in our OxBlue 2D and 3D

teams to further verify the robustness of our method. Meanwhile, third party developed opponents will be used to evaluate the performance of PSP. As to a generalised POMDP, we are keen to explore how the different architectures of a planner can effect learning quality, and to verify its generality in other MAS applications.

References

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

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

[3] Nakashima, T., Udo, M., Ishibuchi, H.: A fuzzy reinforcement learning for a ball interception problem. In: RoboCup 2003: Robot Soccer World Cup VII. (2004) 559–567

[4] Bulka, B., Gaston, M., desJardins, M.: Local strategy learning in networked multi-agent team formation. Autonomous Agents and Multi-Agent Systems **15**(1) (August 2007) 29–45

[5] Ma, J., Li, M., Qiu, G., Zhang, Z.: Q-learning in robocup individual skills. In: China National Symposium on RoboCup. (2005)

[6] Peshkin, L., Kim, K.E., Meuleau, N., Kaelbling, L.P.: Learning to cooperate via policy search. In: Sixteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, Morgan Kaufmann (2000) 307–314

[7] Kok, J.R., Vlassis, N.: Collaborative multiagent reinforcement learning by payoff propagation. J. Mach. Learn. Res. **7** (2006) 1789–1828

[8] Buffet, O., Aberdeen, D.: FF+FPG: Guiding a policy-gradient planner. In: The International Conference on Automated Planning and Scheduling. (2007)

[9] Kimura, H., Yamamura, M., Kobayashi, S.: Reinforcement learning by stochastic hill climbing on discounted reward. In: ICML. (1995) 295–303

[10] Baxter, J., Bartlett, P.: Direct gradient-based reinforcement learning. Technical report, Research School of Information Sciences and Engineering, Australian National University (1999)

[11] Baxter, J., Bartlett, P.: Infinite-horizon policy-gradient estimation. Journal of Artificial Intelligence Research **15** (2001) 319–350

[12] Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. (1990) 81–93

[13] Tao, N., Baxter, J., Weaver, L.: A multi-agent policy-gradient approach to network routing. In: ICML. (2001) 553–560

[14] Buffet, O., Dutech, A., Charpillet, F.: Shaping multi-agent systems with gradient reinforcement learning. Autonomous Agents and Multi-Agent Systems **15**(2) (October 2007) 197–220

[15] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**(3) (May 1992) 229–256

[16] Marinova, Z.: Planning in multiagent systems. Master's thesis, Department of Information Technologies, Sofia University (2002)

[17] Micalizio, R., Torasso, P., Torta, G.: Synthesizing diagnostic explanations from monitoring data in multi-robot systems. In: AIA 06, IASTED International Conference on Applied Artificial Intelligence, Anaheim, CA, USA, ACTA Press (2006) 279–286

[18] Obst, O.: Using a planner for coordination of multiagent team behavior. Programming Multi-Agent Systems **3862/2006** (2006) 90–100
[19] Obst, O., Boedecker, J.: Flexible coordination of multiagent team behavior using htn planning. In: RoboCup 2005: Robot Soccer World Cup IX. (2006) 521–528
[20] Fraser, G., Wotawa, F.: Cooperative planning and plan execution in partially observable dynamic domains. In: RoboCup 2004: Robot Soccer World Cup VIII. (2005) 524–531
[21] Grounds, M., Kudenko, D.: Combining reinforcement learning with symbolic planning. In: Fifth European Workshop on Adaptive Agents and Multi-Agent Systems. (2005)
[22] Strens, M.J.A., Windelinckx, N.: Combining planning with reinforcement learning for multi-robot task allocation. In: Adaptive Agents and Multi-Agent Systems III. (2005) 260–274
[23] Aberdeen, D.: Policy-gradient methods for planning. In: Neural Information Processing Systems. (2005)
[24] Buffet, O., Aberdeen, D.: The factored policy gradient planner (IPC-06 version). In: Fifth International Planning Competition. (2006)
[25] Chen, M., Dorer, K., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Murray, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., Yin, X.: Robocup soccer server for soccer server version 7.07 and later (August 2002)
[26] Pecora, F., Cesta, A.: Planning and scheduling ingredients for a multi-agent system. In: UK PLANSIG Workshop. (November 2002) 135–148