Simultaneous Learning to Acquire Competitive Behaviors in Multi-Agent System based on Modular Learning System

Yasutake Takahashi[1,2], Kazuhiro Edazawa[1], Kentarou Noma[1], and Minoru Asada[1,2]

[1] Dept. of Adaptive Machine Systems, Graduate School of Engineering,
[2] Handai Frontier Research Center,
Osaka University
Yamadagaoka 2-1, Suita, Osaka, 565-0871, Japan
{yasutake, eda, noma, asada}@er.ams.eng.osaka-u.ac.jp
http://www.er.ams.eng.osaka-u.ac.jp

Abstract. The existing reinforcement learning approaches have been suffering from the policy alternation of others in multiagent dynamic environments. A typical example is a case of RoboCup competitions since other agent behaviors may cause sudden changes in state transition probabilities of which constancy is needed for the learning to converge. The keys for simultaneous learning to acquire competitive behaviors in such an environment are

- a modular learning system for adaptation to the policy alternation of others, and
- an introduction of macro actions for simultaneous learning to reduce the search space.

This paper presents a method of modular learning in a multiagent environment, by which the learning agents can simultaneously learn their behaviors and adapt themselves to the situations as consequences of the others' behaviors.

1 Introduction

There have been an increasing number of approaches to robot behavior acquisition based on reinforcement learning methods [1, 4]. The conventional approaches need an assumption that the state transition is caused by an action of a learning agent so that the learning agent can regard the state transition probabilities as constant during its learning. Therefore, it seems difficult to directly apply the reinforcement learning method to a multiagent system because a policy alteration of other agents may occur, which dynamically changes the state transition probabilities from the viewpoint of the learning agent. RoboCup provides such a typical situation, that is, a highly dynamic, hostile environment, in which agents must obtain purposive behaviors.

There are a number of studies on reinforcement learning systems in a multiagent environment. Kuhlmann and Stone [8] have applied a reinforcement learning system with function approximater to the keep-away problem in the situation of RoboCup simulation league. In their work, only the passer learns his policy to keep the ball away from the opponents. The other agents, receivers and opponents, follow fixed policies given by the designer beforehand.

Asada et al. [2] proposed a method that sets a global learning schedule in which only one agent is specified as a learner and the rest of agents have fixed policies. Therefore, the method cannot handle the alternation of the opponent's policies. Ikenoue et al. [3] showed simultaneous cooperative behavior acquisition by fixing learners' policies for a certain period during the learning. These studies suggest that it is enable to acquire a reasonable behavior in a multi-agent environment if the learner can see the environment including the other agents almost fixed because the others keep their policies for a certain time. In the case of cooperative behavior acquisition, both agents do not have any reason to change their policies while they successfully acquire positive rewards with the result of the cooperative behavior for each other. The agents tend to update their policies gradually so that the state transition probabilities seem almost fixed from the view point of the other learning agents.

In case of competitive behavior acquisition in a multiagent environment, however, it is unlikely that the agent tends to select the action that causes positive rewards for the opponents and a negative reward for itself. The punished agent tends to change drastically its policy so that it can acquire a positive reward by which it gives a negative reward to the opponents. This policy alternation causes dynamic changes in the state transition probabilities from the viewpoint of the learning agent therefore it seems difficult to directly apply the reinforcement learning method to a multiagent system.

A modular learning approach would provide one solution to this problem. If we can assign multiple learning modules to different situations, respectively, in each of which the state transition probabilities can be regarded as constant, then the system could show a reasonable performance. Jacobs and Jordan [7] proposed the mixture of experts, in which a set of the expert modules learn and the gating system weights the output of the each expert module for the final system output. This idea is very general and has a wide range of applications (ex. [11, 9, 14, 13, 5]).

We adopt the basic idea of the mixture of experts into architecture of behavior acquisition in the multi-agent environment. Takahashi et al. [12] have shown preliminary experimental results of behavior acquisition in the multi-agent environment, however, the learning modules were assigned by the human designer. In this paper, first, we show how it is difficult to directly introduce a multi-module learning system for even single agent learning in a multi-agent environment because of its complexity and introduce a simple learning scheduling which makes it relatively easy to assign modules automatically. Second, we introduce macro actions to realize simultaneous learning in multiagent environment by which the each agent does not need to fix its policy according to some learning schedule.

Elfwing et al. [6] introduced macro actions to acquire a cooperative behavior with two real rodent robots. The exploration space with macro actions becomes much smaller than the one with primitive actions, then, the macro action increases the possibility to meet cooperative experiences and leads the two agents to find a reasonable solution in realistic learning time. We show the introduction of macro actions enable the agents to learn competitive behaviors simultaneously. We have applied the proposed multi-module learning system to soccer robots which participate in RoboCup competition and show experimental results on computer simulation and real robot implementation.

2 Tasks and assumption

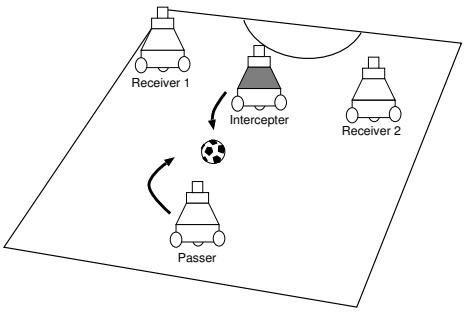


Fig. 1. Task : 3 on 1

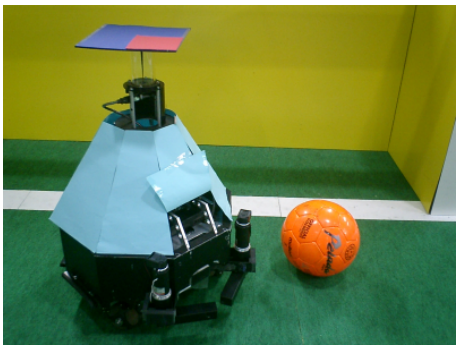


Fig. 2. A real robot

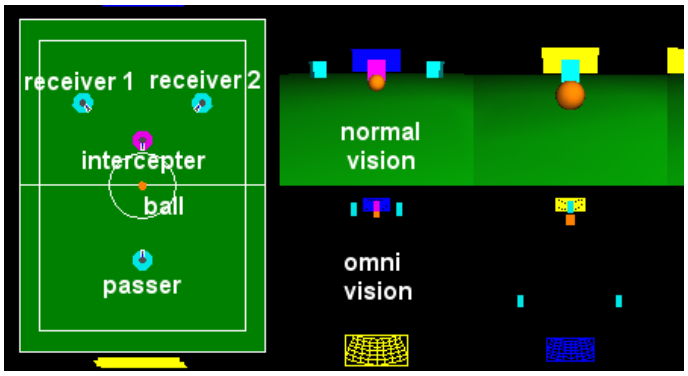


Fig. 3. Viewer of simulator

Fig.1 shows a situation which the learning agents are supposed to encounter. The game is like a three on one; there are one opponent and other three players. The player nearest to a ball becomes to a passer and passes the ball to one of the teammates (receivers) while the opponent tries to intercept it.

Fig. 2 shows a mobile robot we have designed and built. Fig. 3 shows the viewer of our simulator for our robots and the environment. The robot has an omni-directional camera system. A simple color image processing is applied to detect the ball, the interceptor, and the receivers on the image in real-time (every 33ms). The left of Fig. 3 shows a situation in which the agent can encounter and the right images show the simulated ones of the normal and omni vision systems. The mobile platform is an omni-directional vehicle (any translation and rotation on the plane).

3 Multi-Module Learning System

3.1 Basic Idea

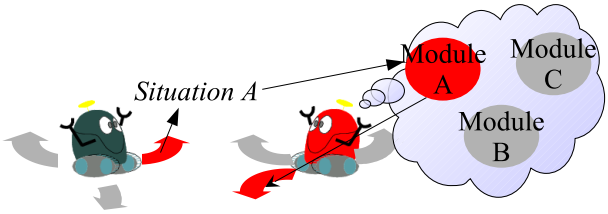


Fig. 4. Adaptive behavior selection based on Multi-module learning system

The basic idea is that the learning agent could assign one behavior learning module to one situation which reflects other agent's behavior and the learning module would acquire a purposive behavior under the situation if the agent can distinguish a number of situations in each of which the state transition probabilities are almost constant. We introduce a modular learning approach to realize this idea (Fig. 4). A module consists of a learning component that models the world and an action planner. The whole system follows these procedures:

- select a model which represents the best estimation among the modules,
- update the model, and
- calculate action values to accomplish a given task based on dynamic programming.

As an experimental task, we suppose ball passing with possibility of being intercepted by the opponent (Figs. 1 and 3). The problem for the passer (interceptor) here is to select one model which can most accurately describe the interceptor's (passer's) behavior from the viewpoint of the agent and to take an action based on the policy which is planned with the estimated model.

3.2 Architecture

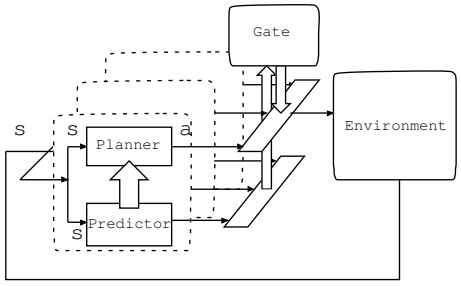


Fig. 5. A multi-module learning system

Fig. 5 shows a basic architecture of the proposed system, that is, a multi-module reinforcement learning system. Each module has a forward model (predictor) which represents the state transition model, and a behavior learner (action planner) which estimates the state-action value function based on the forward model in a reinforcement learning manner. This idea of combination of a forward model and a reinforcement learning system is similar to the H-DYNA architecture [10] or MOSAIC [5]. The system selects one module which has the best estimation of a state transition sequence by activating a gate signal corresponding to a module while deactivating the gate signals of other modules, and the selected module sends action commands based on its policy.

Predictor Each learning module has its own state transition model. This model estimates the state transition probability $\hat{\mathcal{P}}^a_{ss'}$ for the triplet of state s, action a, and next state s':

$$\hat{\mathcal{P}}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{1}$$

Each module has a reward model $\hat{\mathcal{R}}^a_{ss'}$, too:

$$\hat{\mathcal{R}}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \tag{2}$$

We simply store all experiences (sequences of state-action-next state and reward) to estimate these models.

Planner Now we have the estimated state transition probabilities $\hat{\mathcal{P}}^a_{ss'}$ and the expected rewards $\hat{\mathcal{R}}^a_{ss'}$, then, an approximated state-action value function $Q(s, a)$ for a state action pair s and a is given by

$$Q(s,a) = \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma \max_{a'} Q(s', a') \right] , \tag{3}$$

where γ is a discount rate.

Module Selection for Action Selection The reliability of the module becomes larger if the module does better state transition prediction during a certain period, else it becomes smaller. We assume that the module which does the best state transition prediction has the best policy against the current situation because the planner of the module is based on the model which describes the situation best. In the proposed architecture, the reliability is used for gating the action outputs from modules. We calculate an execution-time reliability ${}^{exec}g_i$ of the module i as follows:

$$ {}^{exec}g_i = \prod_{t=-T+1}^{0} e^{\lambda p_i^t} $$

where p_i is an occurrence probability of the state transition from the previous $(t-1)$ state to the current (t) one according to the model i, and λ is a scaling factor. The T indicate a period (step) to evaluate the reliability of the module and we set T as 5 in the following experiments. The agent continues to use the module for a certain period, for example 5 step or 1 second, after it changes the module in order to avoid oscillation of the policies.

Module Selection for Updating Models We use an update-time reliability ${}^{update}g_i$ of the module for updating modules. The calculation of this reliability contains the future state transition probabilities:

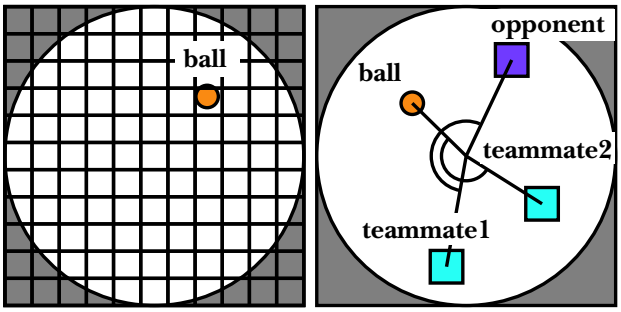
$$ {}^{update}g_i = \prod_{t=t-T}^{t+T} e^{\lambda p_i^t} \quad . $$

4 Behaviors Acquisition under Scheduling

As we mentioned in 1, first, we show how it is difficult to directly introduce the proposed multi-module learning system in the multi-agent system. We introduce a simple learning scheduling in order to make it relatively easy to assign modules automatically.

4.1 Configuration

The state space is constructed in terms of the centroid of the ball on the image, the angle between the ball and the interceptor, and the angles between the ball and the receivers (see Figs. 10 (a) and (b)). We quantized the ball position space 11 by 11 as shown in Fig. 10 (a) and the each angle into 8. As a result, the number of states becomes $11^2 \times 8 \times 8 \times 8 = 61952$. The action space is constructed in terms of desired three velocity values (x_d, y_d, w_d) to be sent to the motor controller (Fig. 7). Each value is quantized into three, then the number of action is $3^3 = 27$. The robot has a pinball like kick device, and it automatically kicks the ball whenever the ball comes to the region to be kicked. It tries to estimate



(a) state variables (position) (b) state variables (angle)

Fig. 6. State variables

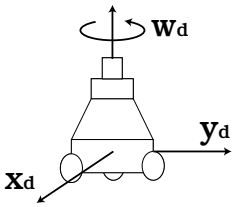


Fig. 7. Action variables

the mapping from sensory information to appropriate motor commands by the proposed method.

The initial positions of the ball, the passer, the interceptor, and receivers are shown in Fig. 1. The opponent has two kinds of behaviors; it defends the left side, or right side. The passer agent has to estimate which direction the interceptor will defend and go to the position in order to kick the ball to the direction the interceptor does not defend. From a viewpoint of the multi-module learning system, the passer will estimate which situation of the module is going on, select the most appropriate module to behave. The passer acquires a positive reward when it approach to the ball and kicks it to one of the receivers.

4.2 Learning Scheduling

We prepare a learning schedule composed of three stages to show its validity. The opponent fixes its defending policy as right side block at the first stage. After 250 trials, the opponent changes the policy to block the left side at the second stage and continues this for another 250 trials. Then, the opponent changes the defending policy randomly after one trial.

4.3 Simulation Result

We have applied the method to a learning agent and compared it with only one learning module. We have also compared the performances between the methods with and without the learning scheduling. Fig. 8 shows the success rates of those during the learning. The success indicates that the learning agent successfully kick the ball without interception by the opponent. The success rate indicates the number of successes in the last 50 trials. The "mono. module" in the figure indicates "monolithic module" system and it tries to acquire a behavior for both policies of the opponent. The multi-module system with scheduling shows a better performance than the one-module system. The monolithic module with scheduling means that we applied learning scheduling mentioned in **4.2** even

though the system has only one learning module. The performance of this system is similar with multi-module system until the end of first stage (250 trials), however, it goes down at the second stage because the obtained policy is biased against the experiences at the fist stage and cannot follow the policy change of the opponent. Since the opponent takes one of the policies at random at the third stage, the learning agent obtains about 50% of success rate. "without scheduling" means that we do not applied learning scheduling and the opponent changes its policy at random from the start. Somehow the performance of the monolithic module system without learning scheduling is getting worse after the 200 trials. The multi-module system without learning schedule shows the worst performance in our experiments. This result indicates that it is very difficult to recognize the situation at the early stage of the learning because the modules has too few experiences to evaluate their fitness, then the system tends to select the module without any consistency. As a result, the system cannot acquire any valid policies at all.

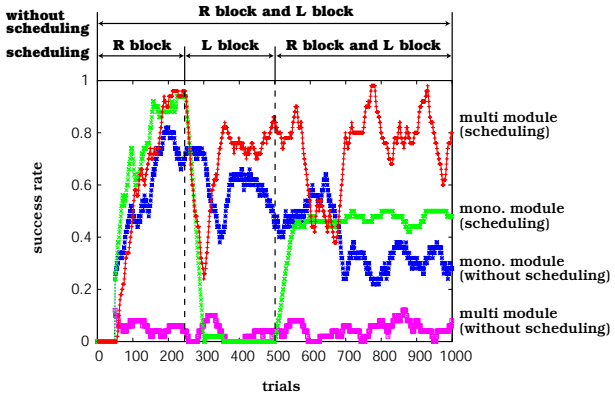


Fig. 8. Success rate during the learning

5 Simultaneous Learning with Macro Actions

We introduce macro actions to realize simultaneous learning in multiagent environment by which the each agent does not need to fix its policy according to some learning schedule. In this experiment, the passer and the interceptor learn their behaviors simultaneously. The passer learns behaviors for different situations which are caused by the alternation of the interceptor's policies, that is, blocking left side or right one. The interceptor also learns behaviors for different situations which are caused by the alternation of the passer's policies, that is, passing a ball to left receiver or right one.

5.1 Macro Actions and State Spaces

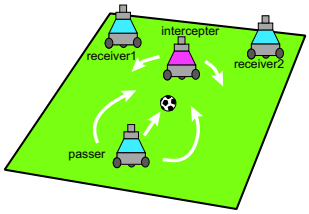
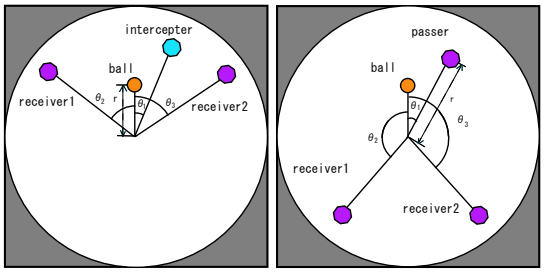


Fig. 9. Macro actions



(a) passer (b) interceptor

Fig. 10. State variables

Fig. 9 shows the macro actions of the passer and the interceptor. The macro actions for the interceptor are blocking the pass way to the left receiver and right one. On the other hand, the macro action for the passer are turning left, turning right around the ball, and approaching to the ball to kick it. A ball gazing control is embedded to the both learners. The number of the actions reduced from 27 (see 4.1) primitives to 2 or 3 macro actions. The state space for the passer is constructed in terms of the y position of the ball on the normal image, and the angle between the ball and the centers of interceptor, the ones between the balls and the two receivers on the image of omni-directional vision. The number of the states reduced from 61952 (see 4.1) to 3773 because the set of macro actions enable us to select smaller number of state variables and coarser quantization. The state space for the interceptor is constructed in terms of the y position of the passer on the image of normal vision system, and the angle between the ball and the passer and the ones between the ball and the two receivers on the image of omni-directional vision. The number of the states is 2695.

5.2 Experimental Results

We have checked how the simultaneous learning of the passer and interceptor works on our computer simulation. Both agents start to learn their behaviors from scratch and have 1500 trials without any scheduling. Fig. 11 show the success rates during the simultaneous learning of the passer and the interceptor. This figure shows that the interceptor has higher success rate at the beginning of learning, the passer is getting to acquire the appropriate behaviors corresponding to the interceptor's behaviors, and the both agents have almost equal success rate at the end of learning stage. The sum of the both success rates is not 1 because the both player sometimes failed to pass or intercept simultaneously.

In order to check if the both learners acquire appropriate behaviors against the opponent's behaviors, we fixed one agent's policy and check that the other

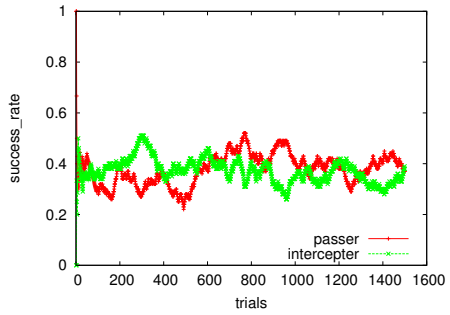


Fig. 11. Sequence of success rates during simultaneous learning

can select an appropriate behavior and its success rate. Table 1 shows these results. The both players have two modules and assign them to appropriate

Table 1. Success rates for passer and receiver in different cases

passer	interceptor	passer's success rate[%]	interceptor's success rate [%]	draw rate[%]
LM0,LM1	LM0	59.0	23.0	18.0
LM0,LM1	LM1	52.7	34.3	13.0
LM0	LM0,LM1	25.6	55.0	19.4
LM1	LM0,LM1	26.0	59.3	14.7
LM0,LM1	LM,LM1	37.6	37.3	25.1

situations by themselves. LM and the digit number right after the LM indicate Learning Module and index number of the module, respectively. For example, the passer uses both LM0 and LM1 and the interceptor use only LM0, then the passer's success rate, interceptor's success rate, and draw rate are 59.0 %, 23.0%, and 18.0%, respectively. Apparently, the player switching multi-modules achieves higher success rate than the opponent using only one module. These results show that the multi-module learning system works well for both.

We have applied the same architecture to the real robots. Fig. 12 shows the one example behaviors by real robots. First, the interceptor tried to block the left side, then the passer approached the ball with intention to pass it to the right receiver. The interceptor found that it tried to block the wrong side and change to block the other side (right side), but, it is too late to intercept the ball and the passer successfully pass the ball to the right receiver.

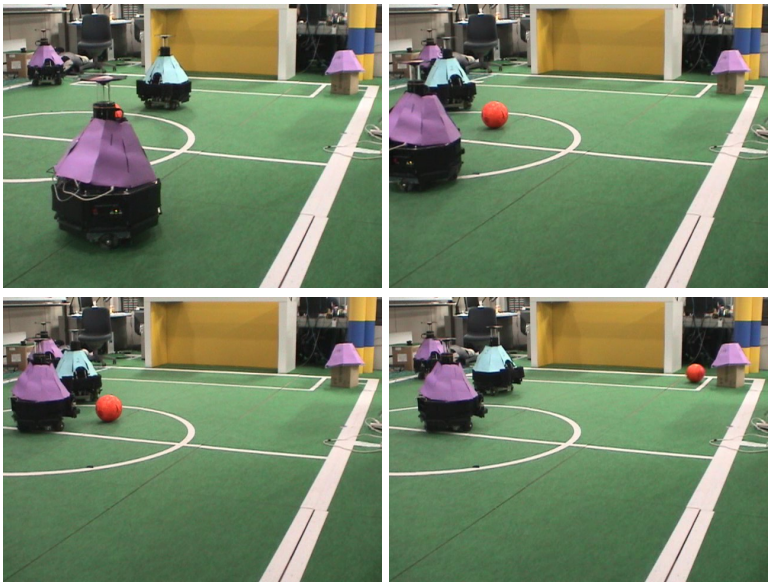


Fig. 12. A sequence of a behavior of passing a ball to the right receiver while the interceptor blocks the left side

6 Conclusion

In this paper, we proposed a method by which multiple modules are assigned to different situations which are caused by the alternation of the other agent policy and learn purposive behaviors for the specified situations as consequences of the other agent's behaviors.

We introduced macro actions to realize simultaneous learning of competitive behaviors in a multi-agent system. We have shown results of a soccer situation and the importance of the learning scheduling in case of none-simultaneous learning without macro actions and the validity of the macro actions in case of simultaneous learning in the multi-agent system.

References

1. M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.
2. M. Asada, E. Uchibe, and K. Hosoda. Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. *Artificial Intelligence*, 110:275–292, 1999.
3. Shoichi Ikenoue Minoru Asada and Koh Hosoda. Cooperative behavior acquisition by asynchronous policy renewal that enables simultaneous learning in multiagent environment. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 2728–2734, 2002.

4. Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.
5. Kenji Doya, Kazuyuki Samejima, Ken ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. Technical report, Kawato Dynamic Brain Project Technical Report, KDB-TR-08, Japan Science and Technology Corporatio, June 2000.
6. Stefan Elfwing, Eiji Uchibe, Kenji Doya, and Henrik I. Christensen1. Multi-agent reinforcement learning: Using macro actions to learn a mating task. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages CD–ROM, Sep 2004.
7. R. Jacobs, M. Jordan, Nowlan S, and G. Hinton. Adaptive mixture of local expoerts. *Neural Computation*, (3):79–87, 1991.
8. Gregory Kuhlmann and Peter Stone. Progress in learning 3 vs. 2 keepaway. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup-2003: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.
9. Satinder P. Singh. The effeicient learnig of multiple task sequences. In *Neural Information Processing Systems 4*, pages 251–258, 1992.
10. Satinder P. Singh. Reinforcement learning with a hierarchy of abstract models. In *National Conference on Artificial Intelligence*, pages 202–207, 1992.
11. Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.
12. Yasutake Takahashi, Kazuhiro Edazawa, and Minoru Asada. Multi-module learning system for behavior acquisition in multi-agent environment. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages CD–ROM 927–931, October 2002.
13. J. Tani and S. Nolfi. Self-organization of modules and their hierarchy in robot learning problems: A dynamical systems approach. Technical report, Sony CSL Technical Report, SCSL-TR-97-008, 1997.
14. Jun Tani and Stefano Nolfi. Self-organization of modules and their hierarchy in robot learning problems: A dynamical systems approach. Technical report, Technical Report: SCSL-TR-97-008, 1997.