Coordination in multiagent reinforcement learning systems by virtual reinforcement signals[1]

M.A.S. Kamal[a,*] and Junichi Murata[b]
[a]*Department of Mechatronics Engineering, International Islamic University Malaysia, PO Box 10, 50728 Kuala Lumpur, Malaysia*
[b]*Department of Electrical and Electronic Systems Engineering, Kyushu University, 744 Motooka, Nishi-Ku, Fukuoka 819-0395, Japan*

Abstract. This paper presents a novel method for on-line coordination in multiagent reinforcement learning systems. In this method a reinforcement-learning agent learns to select its action estimating system dynamics in terms of both the natural reward for task achievement and the virtual reward for cooperation. The virtual reward for cooperation is ascertained dynamically by a coordinating agent who estimates it from the change in degree of cooperation of all agents using a separate reinforcement learning. This technique provides adaptive coordination, requires less communication and ensures agents to be cooperative. The validity of virtual rewards for convergence in learning is verified, and the proposed method is tested on two different simulated domains to illustrate its significance. The empirical performance of the coordinated system compared to the uncoordinated system illustrates its advantages for multiagent systems.

1. Introduction

Multiagent systems (MAS) consist of a number of autonomous agents and form a particular type of distributed artificial intelligence systems that are different from single agent systems in the sense that there is no global control and globally consistent knowledge [1]. The most important reason to use MAS is that some domains require it due to distributed resources, handling a deluge of information, multiple goals and distributed or parallel processing requirements in the system. Even when distributed approach is not required, multiple agents may still provide an excellent way of scaling up by streamlining the search through the space of possible policies. Distribution brings up inherent advantages such as scalability, fault-tolerance, parallelism, etc. In MAS, an action of an agent should be usually good apparently for its own goal, but it may make the environment-state worse to other agents. Therefore, personal goal-oriented actions of all agents make them competitive, and ultimately reduce their individual success as well as combined success. To ensure their cooperative actions a coordination technique should be adopted in MAS. Coordination is the process of effectively managing interdependencies between activities distributed across agents so as to derive maximum benefit from them [3,13]. The degree of coordination is the extent to which the agents avoid extraneous activity by reducing resource contention, avoiding deadlock and livelock, and maintaining applicable safety conditions. Coordination among non-antagonistic agents is called cooperation [26]. But how agents actions can be coordinated is challenging, especially in environments where autonomous agents are motivated by personal goals. To have better coordinated efforts from agents, predefined coordination strategy in the form of rules and restrictions can be embedded in agents [2,21].

[1] This research was conducted while the corresponding author was in Kyushu University.

[*] Corresponding author. Tel.: +603 6196 4417; Fax: +603 6196 4433; E-mail: maskamal@ieee.org.

ISSN 1327-2314/07/$17.00

Such a system may work satisfactorily for a particular problem, but it requires exact knowledge of the system in designing coordination strategy and may have lack of adaptiveness to changing situations.

Reinforcement learning (RL) is a simple technique whereby an autonomous agent with its goal embedded in an environment learns how to transform any environmental state into another that contains goal [8,20]. This learning technique of the autonomous agent is useful for designing an MAS. Still how agents efforts can be efficiently coordinated in a multiagent reinforcement learning system is a fundamental issue. Researchers in both human and computational organizational theories have pointed out that there is no single organization of coordination protocol that is the best for all environments [3,4,15,19]. In a simple coordination scheme in multiagent RL, an agent treats the other agents as a part of the environment and they do not communicate with each other [2,16,23]. But it can only be applied to the environment having only few agents, since the state size expands exponentially with the number of agents and actions of other agents change the environment into dynamic one. Agents can be designed to share perception information to overcome perceptual limitations or communicate policy functions learned through reinforcement learning [6,21]. It may boost the learning speed and performance where the sharing of meta-level information is needed, but it does not make explicit the notion of local policy in a more global, abstract situation [15]. In collective intelligence framework a payoff function called ‘wonderful life utility’ has been derived as an alternative to a team-game payoff utility for the distributed agents that work in different subworlds. Choosing appropriate parameters of the payoff functions, distributed agents could maximize the world utility functions [24,25]. An online coordination in multiagent RL system was proposed in [9]. The coordinator evaluates overall system dynamics in term of global state value and provides it to all agents. Instead of estimated value of next state, estimated value of the global state is used as the target value of individual agents in policy update unlike any RL algorithm. The method worked satisfactorily for one-step RL optimization task, but it cannot be applied in multi-step task, because it is almost impossible to define global state space that properly reflects and matches the dynamics of all agents at any time throughout their individual state spaces. In this paper the above complicated technique is completely modified so that it can be applied to both one-step and multi-step tasks using simpler learning rules.

The proposed method uses conventional RL algorithm with a new coordination method, and is useful to the class of problems that requires cooperative actions of agents, especially where individual agents have their own goal which cannot be achieved without their proper cooperation or where they may vie with each other for the goal. To make them cooperative rather than competitive in such a system, agents should be motivated by both the personal goal and the goal of other agents. In the proposed coordination scheme, a coordinating agent estimates their degree of cooperation and provides a virtual reinforcement to the individual agent based on it. Each agent updates task-oriented policy using both the natural reinforcement from the environment and the virtual reinforcement from the coordinator. The natural reinforcement makes the agent greedy for its own benefit, whereas the virtual reinforcement from the coordinator guides the agent to be cooperative with others. This virtual reinforcement gives a critic of the selected action on how much it is good for all the agents. These dual reinforcements let an agent attain better coordinated behavior through its policy keeping a balance of its own benefit and their combined benefit.

The proposed algorithm is tested on two different multiagent domains: the hunter-prey domain where the hunters seek to catch a prey; and a domain of taxi service control, which is a partially distributed partially centralized system of agents. The agents have shown better coordinated efforts than the non-coordinated agents and agents following other coordination strategies, and they have maximized their combined success without worsening the individual achievement in both cases.

The rest of the paper is organized as follows. The proposed coordination mechanism is introduced in Section 2, after a review of the reinforcement learning theory. The first testbed ‘hunter-prey domain’ is presented in Section 3, followed by an implementation procedure of the proposed learning algorithm and simulations. Section 4 describes the second testbed ‘taxi service control’, its implementation and simulations. Conclusions and future research directions are included in Section 5.

2. Coordinated multiagent reinforcement learning

2.1. Reinforcement Learning

Reinforcement learning (RL) is a process of trial-and-error whereby an agent seeks to find the combination of actions that maximizes the reinforcements as its performance feedback [8,20]. Markov decision process (MDP) is the most widely accepted mathematical model of RL [8,17,18]. An MDP is a quadruple of $\langle S, A, P, R\rangle$ and also called multistage decision task, where for a finite set of all possible states S of the environment and a finite set of all possible actions A in each state, the state transition P and an expected scalar reward R are the functions of state action pair, $P : [S \times A \rightarrow S]$ and $R : [S \times A \rightarrow R]$, respectively. At each time step, an agent observes its environment and selects the next action based on that observation. In the next time step, the agent obtains the new observation that may reflect the effects of its previous action and a reward indicating the quality of the selected action. The process is repeated and based on this cycle, the agent learns the optimal policy.

One of the most commonly used reinforcement learning method is Q-learning [22]. This algorithm does not need a model of the environment and directly computes the approximate function of the optimal action-value independent of the policy followed. The updating rule of this off-policy RL is as follows:

$$\begin{aligned} Q(s_t, a_t) \leftarrow{}& (1-\alpha)Q(s_t, a_t) \\ &+\alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)], \end{aligned} \tag{1}$$

where α is the learning rate, γ is the discount factor, r_{t+1} is the achieved reward at time $t+1$, and $Q(s_t, a_t)$ is the value of action a_t at state s_t. The convergence of RL is proven under the assumption that each state-action pair is visited infinitely often.

2.2. Coordinated multiagent reinforcement learning

Agents in an MAS can follow the above RL rules independently and concentrate on their own target with or without considering dynamics due to other agents in the system. So without additional coordination, the agents may be partly or fully competitive rather than cooperative since they are directed by personal rewards. To ensure the cooperation, we propose a new method for policy coordination where agents are given two kinds of rewards: natural rewards from the environment, and virtual rewards for cooperation which is estimated by an additional learning. In the proposed coordinated multiagent reinforcement learning (CMRL) system, an agent plays the role of coordinator to estimate the degree of cooperation achieved and provides the virtual reinforcements to the agents based on it. To learn the pattern of combined cooperative behavior of agents, a new state called 'state of cooperation' is defined. The 'state of cooperation' (SOC) s^c is composed of concise information of all agents related to their goal. The information constituting SOC can be a part of information perceived by all the agents. So observing SOC requires no or small amount of communication among agents. The coordinator updates the value (goodness) of the SOC, $V(s^c)$, after each transition using the rewards for achieving the goal but ignores local rewards of all agents. This also reduces the amount of communication. So its value reflects the degree of cooperation achieved by the agents at any stage. The following standard RL algorithm is used to update the values of SOC:

$$\begin{aligned} V(s^c_t) \leftarrow{}& (1-\alpha)V(s^c_t) \\ &+\alpha[\textstyle\sum r_{t+1} + \gamma^c V(s^c_{t+1})], \end{aligned} \tag{2}$$

where the agents take actions at time t which altogether cause the transition of SOC from s^c_t to s^c_{t+1} and one or some agents attain the goal and receive a reward r_{t+1}. If two or more agents receive the rewards, their sum is used to update $V(s^c_t)$. Since SOC is composed of concise information, the learning problem described by Eq. (2) is not large. The SOC s^c_t mainly represents the status of all agents concerning their goal and its value indicates how much their overall situation is good to reach the goal by the agents. Dislike the rewards that are available at only particular states, $V(s^c_t)$ tells the goodness of the state for each and every state, which will successfully guide the agents to described state through their state transitions. This guidance is made by giving the agents virtual rewards based on $V(s^c_t)$. Any transition to a better SOC ($\Delta V = V(s^c_{t+1}) - V(s^c_t)$ is positive) indicates the action is good for the society although it may not be good to a particular agent in the short run. Similarly, any transitions to a worse SOC (ΔV is negative) indicates the action is not good for the society although it may be very good to a particular agent. Therefore the coordinator gives a virtual reinforcement signal $r^c = \Delta V$ to the agents after each action as an indication how good the action just taken was from the whole system's viewpoint. So the learning equation for an agent can be written as:

$$\begin{aligned} Q(s_t, a_t) \leftarrow{}& (1-\alpha)Q(s_t, a_t) \\ &+\alpha[r_{t+1} + \beta r^c_{t+1} \\ &+\gamma \max_a Q(s_{t+1}, a)], \end{aligned} \tag{3}$$

where β is the coordinating factor. If β equals zero, the learning fully concentrates on non-cooperative mode and an agent becomes greedy to its own rewards. If β is large, the agent takes cooperative actions by trading off its own rewards for reward for the cooperation. Adjusting the coordinating factor β the individual behavior of agents can be set in between competitive or partial cooperative to full cooperative modes.

If the goal state is far from the beginning state, in this proposed learning mechanism an agent often receives virtual reinforcement signals depending on the change in the value of SOC. So the agent may tend to collect all of these rewards instead of going to the goal state in the optimum way or it may fail to converge at all. So an investigation is needed whether the proposed algorithm of cooperation works well or not. For simplicity, let us assume a case of a single agent system where the reward r is given only at the final goal state, and both individual state and SOC are the same, say s. Then, the value of SOC is determined by the number of transitions to reach the final goal state. Let us denote the number of state transitions from the current state s to the final goal state by M. Then the value of SOC s is equal to $\gamma_c^M r$ where γ_c is the discount factor for estimating the value of SOC. If the state transitions are optimal or they follow the optimal path from the current state to the goal state, then the virtual reward (the difference of the values between two consecutive SOCs) is:

$$\begin{aligned} r^c &= \Delta V = V(s^c_{t+1}) - V(s^c_t) \\ &= (\gamma_c^{M-1} - \gamma_c^M)r. \end{aligned} \tag{4}$$

If this virtual reward r_c is used for each agent along with reward r for the task achievement, then the value of the state perceived by the agent can be calculated as:

$$\begin{aligned} V_{\text{agent}} &= \beta[(\gamma_c^{M-1} - \gamma_c^M)r \\ &\quad + \gamma(\gamma_c^{M-2} - \gamma_c^{M-1})r \\ &\quad + \ldots + \gamma^{M-1}(1-\gamma_c)] + \gamma^M r \\ &= \beta(1-\gamma_c)(\gamma_c^{M-1} + \gamma\gamma_c^{M-2} \\ &\quad + \ldots + \gamma^{M-1})r + \gamma^M r, \end{aligned} \tag{5}$$

where γ is the individual discount factor. If γ is chosen to be $\kappa\gamma_c$, then the local value given above will be (regarded as a function of M):

$$\begin{aligned} V_{\text{agent}}(M) &= \beta(1-\gamma_c)\gamma_c^{M-1} \\ &\qquad (1 + \kappa + \ldots + \kappa^{M-1})r + \gamma_c^M \kappa^M r \\ &= \beta(1-\gamma_c)\gamma_c^{M-1}(\frac{1-\kappa^M}{1-\kappa})r \\ &\quad + \gamma_c^M \kappa^M r. \end{aligned} \tag{6}$$

For the convergence in learning this value must be a monotonically decreasing function of M, since M is the distance (measured in the number of state transitions) between the current state and the final goal state and the value must be higher when closer to the goal. Assuming continuous M, we have

$$\begin{aligned} \frac{dV_{\text{agent}}(M)}{dM} &= \frac{r\gamma_c^{M-1}}{1-\kappa}[\beta(1-\gamma_c)\log\gamma_g \\ &\quad + \kappa^{M-1}\log(\kappa\gamma_c) \\ &\quad (1 - \kappa - \beta\kappa + \beta\kappa\gamma_c)]. \end{aligned} \tag{7}$$

This derivative is negative for $M \geqslant 1$ if the following conditions are met:

$$(1 - \kappa - \beta\kappa + \beta\kappa\gamma_c) > 0. \tag{8}$$

Thus,

$$\kappa < \frac{-1}{\beta(\gamma_c - 1) - 1} = \frac{1}{\beta(1-\gamma_c) + 1}. \tag{9}$$

So for the case $\beta = 0$: $\kappa < 1.0$; and for the case $\beta = 1$: $\kappa < \frac{1}{2-\gamma_c}$. Therefore, if the above condition is met, the virtual reinforcement can be valid. So the selection of discount factors for both learning is the key point for the convergence. The above condition is for a single agent system where both states change simultaneously and both virtual and natural reinforcement signals are used in individual learning. For some problems the state for coordination can be different from the individual state, so their transitions are not synchronous. The above condition may be shifted slightly depending on the asynchronous states transitions.

The proposed method indirectly coordinates agents actions through their policy and can be applied to a number of multiagent systems where autonomous agents have to work as a team and a coordinator can at least partially observe the agents activity in the system. This method can be applied to multiagent task scheduling, survival world of agents and the systems where the agents have to work against uncertainty, dynamism or intelligent opponents. This coordination technique is adaptive, requires simple computation and less communication among the agent. But it needs a coordinator and proper design of SOC that represents status of all agent in a concise form.

3. Hunter-prey domain

A hunter-prey domain is chosen as our first testbed, which is a special case of pursuit-evasion problems [7,

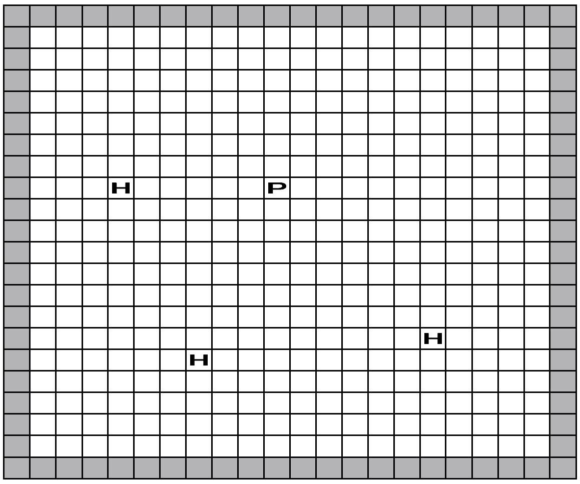

Fig. 1. The hunter-prey world with three hunters (H) and one prey (P).

14]. In such a domain, hunters move around the environment trying to catch the prey, and the prey tries to evade the hunters. Pursuit-evasion tasks are interesting because they are ubiquitous in the natural world, and offer a clear objective that requires complex coordination considering the nature of the environment, goal of the other agents and adversarial agents [5].

In this paper, the environment is a square maze of dimension 22 × 22 and it has an obstacle boundary in its periphery, Fig. 1. The domain has one prey and three hunters (later four-prey three-hunter environment is considered). Both hunters and the prey can move one step in north, south, east, west cell or stay stand still in a unit time step. The hunters seek to capture the prey but the prey tries to flee away from the hunters. The prey examines the fear of hunters in the north, south, east, west and its current cells, and always moves in the cell of least fear of hunters. Fear of a single hunter is defined as 3, 2, 1, or 0 for a cell if the cell is 1, 2, 3 or more than 3 steps far from the hunter, respectively, and fear of 1 if it is one step away from any obstacle. The prey sums up the fear of all hunters in a cell considering nearby hunters and obstacles wall. Figure 2 shows how fear of hunters is calculated in the encircled cell. Sum of fear at the cell is 4 that is calculated by summing up F_{H1} (fear of hunter H1), F_{H2}, F_{H3} and F_{OB} (fear of obstacle). Since the prey has this intelligence to flee away from a hunter with the same speed, a single hunter cannot catch it at all. We choose this complex and dynamic domain to focus on the capability of the CMRL technique.

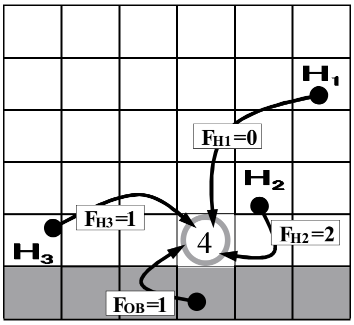

Fig. 2. Fear from hunter H1, H2, H3 and obstacle at the encircled cell.

3.1. Implementation

To ensure the best efforts as a team of hunters, the proposed CMRL technique can be used as follows. Each hunter agent follows its own policy to catch the prey, and one hunter plays an additional role of coordinator to feedback the virtual reward for cooperation. The state of individual hunter consists of the following information of the environment:

- status of eight neighbor cells of the agent (vacant or not), 2^8 possible values;
- relative direction of the prey (sign of relative $x - y$ coordinates: negative, zero, positive; 3 × 3 possible value;
- distance of the prey (unsigned relative $x - y$ values: 22 × 22 possible values);
- the existence of either any hunter or a wall of the obstacle boundary in four zones (the surrounding cells within three steps in distance from the prey towards all directions are divided into four quadrants, Fig. 3): yes or no (2^4 possible values).

A hunter agent is considered to have caught the prey when it enters the same cell of the prey, and it receives a reward of 1.0. It receives a punishment -1.0 if it tries an illegal move (going to a cell having obstacle or a hunter). All other actions are not given any reinforcement (reinforcement is 0). A virtual reinforcement ΔV is given from the coordinator after the action as described below. It updates its policy according to Eq. (3).

Besides the above learning, the coordinator-hunter estimates the value of 'state of cooperation' SOC of all hunters by Eq. (2). The prior knowledge about the problem helps to select appropriate information in constituting SOC. If the prey is properly surrounded by hunters and obstacle boundary then it cannot flee away from them. The existence of either any hunter or

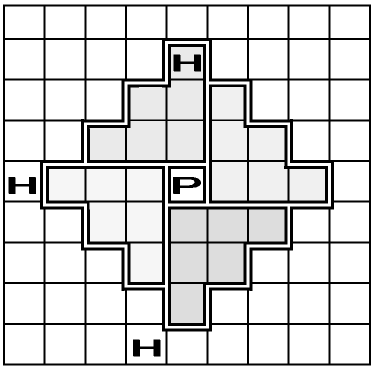

Fig. 3. Four surrounding zones within 3 steps from the prey.

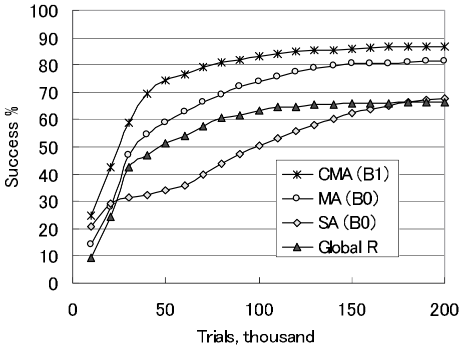

Fig. 4. Learning of four systems in term of average success rates of catching the prey.

a wall of the obstacle boundary in four zones (Fig. 3) is used as SOC (yes or no: 2^4 possible values). If hunters and obstacles boundary are in different zones then the SOC will have the highest value since by the way they surround the prey properly and the prey can be caught easily. If a hunter fills up an empty zone, then value of SOC increases, they receive a positive virtual reward. Similarly, if the number of empty zones increases it decreases the value of SOC and they receive negative virtual reward. The value of SOC is updated after each transition. The individual reward of all agents for catching the prey (1.0) is used for this update but other reinforcement signals (e.g. punishment for an illegal move) to the individual hunter are ignored. After an action of agents, the coordinator hunter estimates ΔV equals to the change in values of SOC and provides it as a virtual reinforcement to the corresponding hunters.

3.2. Simulations

The performance of the team of hunters is tested in terms of successful trials rate and average time to catch a prey in successful trials. Here a successful trial means the prey is caught by a hunter within the given time of 50 steps. A constant learning rate α, discount factors γ_c and γ are set at 0.03, 0.93 and 0.96, respectively. ϵ-greedy action selection with decreasing value of ϵ is used in learning. The initial value of ϵ is set at 0.1, it decreases to 0.01 in one million trials. According to the necessary condition of convergence using the virtual reinforcement, γ must be smaller than γ_c. In this settings, the transition in SOC does not take place at each action of the hunter and we have found after about every 3.97 (average value) actions of a hunter the SOC changes. It means only the every fourth (approximately) action is given a non zero virtual reward (ΔV). Therefore the γ acts internally at a value of $0.96^{3.97} = 0.85$, which satisfies the required condition of convergence.

We considered the case of three hunters and one prey in the domain. At the beginning of each trial the hunters and the prey are set at different random locations in the world. To ensure a minimum level of difficulty, the hunters are allowed to start their moves 3 time-steps later than the prey. At first, learning of the proposed coordinated multi-agent (CMA) system, in which each agent considers existence of other agents (in four zones) as part of the environment and receives virtual rewards from coordinator ($\beta = 1$), is compared with other three systems in Fig. 4. In multi-agent (MA) system, similar to [2,16,23], each agent considers existence of other agents (in four zones) as part of environment (but there is no virtual rewards provided by coordinator). In single agent (SA) system, agents ignore each other completely. In Global Reward (GR) system, each agent considers existence of other agents (in four zones) as part of environment and reward of catching prey is global to all agent (all agents receive the same reward if anyone of them catches the prey). Taking information of other agents in state as in MA system and sharing the same rewards as in GR system are the well-known coordination techniques in multi-agent RL problems. The success percentage of each 10000 trials are plotted after taking average of 30 independent tests. GR system has the worst success rate, since receiving a reward without cooperating others makes a hunter idle or confused. Although a single hunter cannot catch the prey, SA system could achieve 67.9% success rate since their independent greedy actions partly help each other. The MA system have much better success rate (81.2%) than the SA systems, since the hunters partly consider others in their states that makes them cooperative as proposed by some researchers. Still the hunters are motivated by personal goal. CMA system overcomes

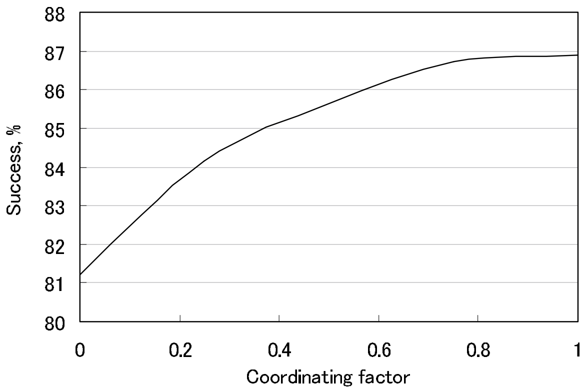

Fig. 5. Average success rates of catching the prey for different coordinating factor β.

this by providing virtual rewards that enhances their cooperative performance to a success rate of 86.9%.

The virtual rewards to the individual hunters are tuned by the coordinating factor β. Figure 5 shows the percentage of successful trials versus the coordinating factor β. Values are taken after convergence in learning and averaging the results of 1 million trials. A factor $\beta = 0$ means the agents actions are not coordinated similar to MA system, so they are not fully cooperative but partly competitive to catch the prey. However their partly competitive actions help them to have a success rate of 81.2%. The success increases with β and at $\beta = 1$ it became 86.9%. The average time required to catch the prey at each trial are 28.7 time units for $\beta = 0$ and 27.5 time units for $\beta = 1$. Thus the proposed coordination mechanism has positive impact on both the success rate and average time required to catch the prey.

The above results are obtained for the prey taking 100% intelligent action to avoid the hunters. We examined a situation where the prey takes 10% random (non-intelligent) moves. In this easier case, we found a success rate of 93.2% and 93.3% for β equals 0 and 1, respectively. The average time required to catch the prey also reduced to 24.7 and 24.5 time units for β equals 0 and 1, respectively. From this result, it is obvious that for an easier case additional coordination is not so important. But for the difficult cases, fully cooperative behaviors by the additional coordination gives better results.

The value of SOC can be used not only for generating virtual rewards but also for other purposes. In above results, SOC is based on the prey and surrounding hunters. If there are many preys then each of them has an SOC (according to its definition in this domain). If different preys are targeted by different hunters then they might not be able to catch them efficiently. So it is better to target a single prey for all hunters and value of SOC can be used for the prey selection. The Table 1 shows the success rates of catching a prey in a 4-prey 3-hunter domain using different learning and targeting methods. Here each trial ends when a prey is caught. Each column shows the performance of each system for different prey selection for hunting. In first three rows, the coordinator specifies a single prey to all hunters. Coordinator specifies a prey that SOC has the highest value (Vmax). The prey with the highest value of SOC is judged every time step (maxV at every step), or the target prey is fixed at beginning and they do not change the target until the end of the trial (maxV at beginning). Fixing a random prey at beginning by coordinator is given in third row. In the last two rows, individual hunters select the nearest prey either at every step (Nearest at every step) or fix a target prey at beginning (Nearest at beginning). The proposed method with prey selection by Vmax at every step provides the best results. By this prey selection using Vmax other methods are also benefited. Table 2 shows the average time-step required to catch the prey.

4. Taxi service control

Online coordination of taxi service control (TSC) for a small city serves as our second test bed (Fig. 6) [9], which contains five taxi stands and ten taxies for a moderate traffic profile approximately 125 rides (passengers or group of passengers) per day. All taxis maintain a queue in each stand to get a passenger and they have a link for limited communication with the control center. The control center has the information of all empty taxi and their waiting stands. Passengers come to the stand seeking a taxi, but some may use other transportation if a taxi is not available. Some passengers (non-stand passengers) call the taxi control center asking for a taxi from anywhere in the city. The destinations of the passengers can be anywhere in the city. Assigning taxis to the non-stand passengers and other coordination among the taxis are conducted by the control center. The objective of TSC system as a whole is to maximize the profit of the company by capturing maximum total passengers. But each taxi-agent tries to get many passengers to maximize its own profit only. So it may go to a stand of relatively higher traffic density, although some taxis are already there, keeping some other stands empty that is not good for the com-

Table 1
Success rates of systems with different prey selection rules

Targeted by	Method	CMA (B1)	MA (B0)	SA (B0)
Coordinator	maxV at every step	92.3%	87.2%	83.3%
	maxV at beginning	89.6%	86.2%	72.3%
	Random at beginning	88.4%	82.2%	71.8%
Hunter	Nearest at every step	75.5%	73.6%	70.1%
	Nearest at beginning	68.4%	66.7%	56.8%

Table 2
Average time-steps required to catch a prey by different systems for different prey selection rules

Targeted by	Method	CMA (B1)	MA (B0)	SA (B0)
Coordinator	maxV at every step	17.6	22.3	22.0
	maxV at beginning	23.9	25.0	25.6
	Random at beginning	25.5	27.2	27.7
Hunter	Nearest at every step	24.8	25.1	25.8
	Nearest at beginning	24.7	25.1	25.8

pany. Virtual reward can help to overcome this conflict of interest.

This kind of multiagent transportation system poses high-dimensionality with a number of hidden states and complex dynamics due to unknown and stochastic passenger patterns. The performance objective has multiple aspects that make this system more complex. First is ensuring taxi availability at each stand and second is sending taxis as quickly as possible to non-stand passengers. If the system chooses the nearest taxi to a non-stand passenger (minimum waiting time) that makes the stand empty sometimes then it may lose the future passengers at that stand. The system has to trade-off these two contradictory aspects to find a suitable balance in operation in maximizing systems achievement. If the company increases the number of taxis and the number of stands throughout the city, then it may be able to catch almost all passengers providing best services. But it will not be cost effective since the passengers per day are limited. This testbed is chosen to focus the applicability of the proposed CMRL technique in such a complex real world problem.

4.1. Implementation

Task-oriented reinforcement learning (TORL) method [10,11] has been applied successfully to Elevator Group Control (EGC) [12]. TSC is similar to EGC as both systems have more than one learning tasks in carrying passengers. But TSC poses further difficulties as a distributed systems. Passengers may get in/out a taxi anywhere and some of them originally supposed to use a taxi may use other alternative transporting services due to unavailability of a taxi in the stand. In this problem the whole task can be divided into two main sub-tasks. First task is to select an appropriate taxi for the non-stand passengers. An agent in the taxi center carries out this learning when a passenger asks for a taxi. The whole city is divided into $6 \times 4 = 24$ zones, as shown in Fig. 6. The taxi availability at each taxi stand and calling zones in the city are used as the state for this task. The agent is given a reward after each action that equals to the reciprocal of waiting time of the passenger. Second task, carried out by each taxi agent when it becomes empty, is to select a suitable taxi stand to wait for a passenger. The numbers of taxis at five taxi stands are used as the state of this stand selection task. The taxi-agent receives a reward when a passenger gets in it. The reward is the reciprocal of its waiting time in the stand.

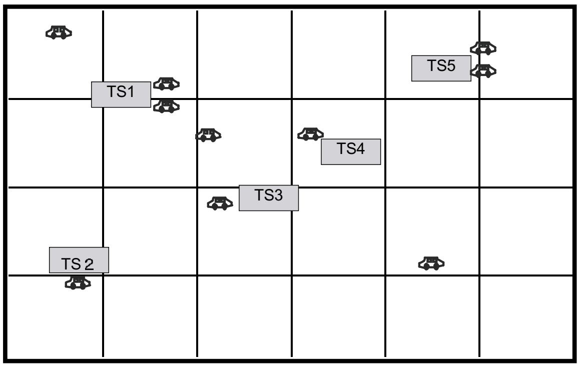

Fig. 6. Schematic diagram of the zones and taxi stands (TS) in a simulated city.

Both tasks are completely different from each other and there is no direct impact of an action to other agents or its future actions. So an agent takes an action and updates its policy after getting the reward with-

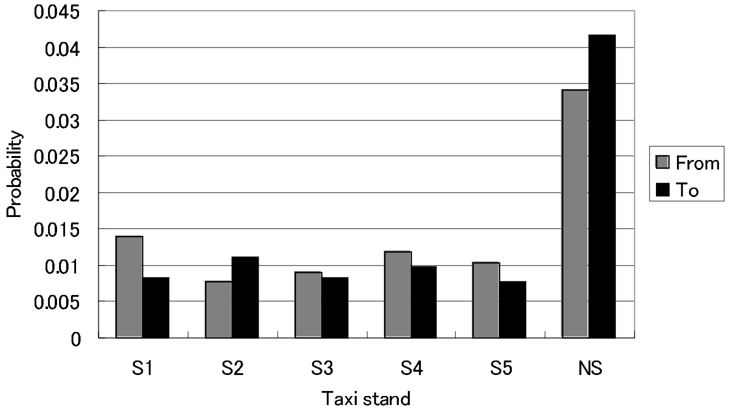

Fig. 7. Probability of passengers for every minute at all stands and non-stand.

out taking future dynamics. But from the view point of the TSC in maximizing its passengers, both actions have direct impact to its global objective. So the proposed CMRL is used to provide a virtual reward in both learning to make the agents cooperative in maximizing global benefit of TSC. The agent in the taxi control center can play the additional role as the coordinator to estimate the values of SOC by Eq. (2). It is apparent that availability of taxis in all stands ensures getting passengers who comes at any stand in the system. So the number of taxi stands with available taxis can be used to constitute SOC. SOC for no empty stand will have the highest value (function of rewards from all taxi agents) followed by SOC for one empty stand and so on. If a taxi agent goes to an empty stand then value of SOC increases (good for the company), and virtual reward, a positive ΔV, will encourage the agent to do so next time. Similarly, the control center agent will try to choose the taxi for non-stand passengers in such a way that the number of empty stand does not increases (that worsens the value of SOC). A passenger arrival gives a reward equals to the reciprocal of the elapsed time from the last passenger arrival in TSC. If any event (stop or start of a taxi) occurs, the coordinator agent updates the value of SOC and informs its change ΔV to the corresponding individual agents as a virtual reinforcement which indicates how much the action was good for getting future passengers.

4.2. Simulations

A typically simulated non-uniform passenger arrival pattern is used to generate the passenger data on 24-hours of a day. Figure 7 shows the stochastic distribution of passengers or group of passengers per minute for the taxi stands $TS1$ to $TS5$ and non-stand (NS) passengers. Here a from-passenger wants to get in a taxi at the stand, while a to-passenger arrives at the stand in a taxi. The numbers of passengers at the stands are not the same (non-uniform profile), which further enhances the difficulty. The performance of the CMRL system is measured in terms of waiting time of the passengers and percentage of passengers who could get a taxi at a stand. A fixed learning rate of $\alpha = 0.03$, discount factor $\gamma_g = 0.9$ and ϵ-greedy policy of action selection are used in the simulations. The value of ϵ decreases from an initial value of 0.1 to 0.03 in 1 million actions. In both tasks, individual actions are given immediate reward without estimating the future action values. It means the discount factor γ for an individual agent is zero which fulfills the condition in CMRL for convergence.

Figure 8 shows the average passenger rate who could get a taxi for different values of the coordinating factor β after convergence in learning. In this learning the passenger pattern shown in Fig. 7 is used. Without coordination ($\beta = 0$) the system could manage 82.1% of the total passengers (who actually want to use a taxi but some of them cannot use it due to unavailability of taxi in the stands) whereas for $\beta = 1$ it becomes 88.1%. But the system has to sacrifice its service level to the non-stand passengers since the waiting time increases from 13.0 min to 15.0 min, Fig. 9.

The above results are for passenger pattern of Fig. 7. The performance of the system has also been tested for uniform traffic profile (a much easier case) that is the passenger arrival/departure probability is uniform over all stands, keeping the number of passengers the same as 125 rides per day. The average passenger rate who could catch a taxi per day is found 91.9% for $\beta = 1$, and 89.6% for $\beta = 0$, whereas average waiting time of the passengers are 14.8 min for $\beta = 1$ and 12.9 min for $\beta =$

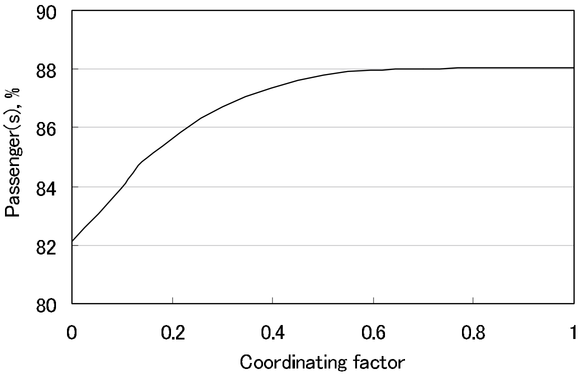

Fig. 8. Average percentage of passengers who could use a taxi for different coordinating factor β.

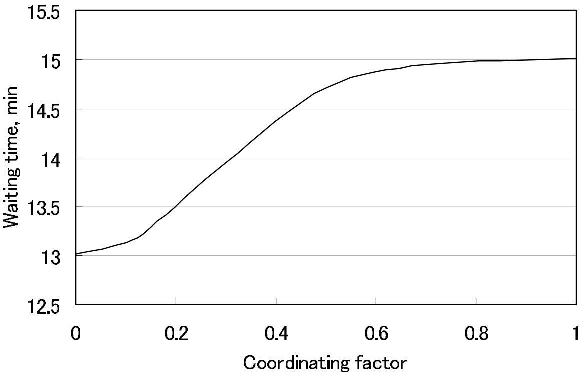

Fig. 9. Average waiting time of non-stand passengers for different coordinating factor β.

0. Still the fully coordinated system has slightly better overall outcome, but both systems have close values in both percentage of passengers and waiting time in the uniform traffic pattern. From these results, it is obvious that for uniform pattern the non-cooperative system can perform at satisfactory level. But performance goes down if the pattern is non-uniform or having uncertainty.

5. Conclusions

An online coordination algorithm for multiagent reinforcement learning has been proposed in this paper that indirectly coordinates the actions of individual agents in maximizing the social benefit. This coordination method is unique in reinforcement learning and the encouraging results show the effectiveness of this mechanism in attaining complex objectives of the multiagent systems. This RL-coordination technique can also be introduced to other multiagent systems where the individual agents do not use RL learning so that the autonomous agents can work as a team against uncertainty, dynamism and intelligent opponents. The possible scope of this technique are in scheduling distributed tasks among the agents, processes coordination in the industry, traffic coordination in a highway, etc. It can be interesting to use this idea of coordination to make an agent more competitively against opponents in a game.

Acknowledgements

The research was partly supported by the Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (C), 16560354, 2004–2006.

References

[1] O. Abul, F. Polat and R. Alhajj, Multiagent Reinforcement Learning Using Function Approximation, *IEEE Trans on Syst, Man, and Cybern, Part C: Application and Reviews* **30**(4) (2000), 485–497.

[2] R.H. Crites and A.G. Barto, Elevator Group Control using multiple Reinforcement Learning Agents, *Machine Learning* **33**(2–3) (1998), 235–262.

[3] K.S. Decker and V.L. Lesser, *Designing a family of coordination algorithms*, in Proc. of the first Int. Conf. on Multi-Agent Systems, AAAI Press, 1995, 73–80.

[4] M.S. Fox, An Organizational view of distributed systems, *IEEE Trans on Syst, Man, and Cybern* **11**(19) (1981), 70–80.

[5] F. Gomez and R. Miikkulainen, Incremental evaluation of complex general behavior, *Adaptive Behavior* **5** (1991), 317–342.

[6] J.J. Grefenstette, The Evolution of Strategies for Multi-agent Environments, *Adaptive Behavior* **1**(1) (1992), 65–89.

[7] L.J. Guibas, J.-C. Latombe, S.M. LaValle, D. Lin and R. Motwani, A visibility-based pursuit-evasion problem, *Intl Journal of Computational Geometry and Applications* **9**(4–5) (1999), 471–478.

[8] L.P. Kaelbling, M.L. Litman and A.W. Moore, Reinforcement learning: A survey, *Journal of Artif Intell Res* **4** (1996), 237–285.

[9] M.A.S. Kamal and J. Murata, *Coordination in Multiagent Reinforcement Learning Systems,* Proc. of Int. Conf. on Knowledge-Based Intelligent Information & Engineering Systems (KES 2004), 1198–1204.

[10] M.A.S. Kamal, J. Murata and K. Hirasawa, Task Oriented Reinforcement Learning for Continuing Task in Dynamic Environment, Research Reports on Info. Sc. and Elect. Eng. of Kyushu University, Japan, vol. 9, No. 1, 2004, 7–12.

[11] M.A.S. Kamal, J. Murata and K. Hirasawa, *Task-Oriented Reinforcement Learning for Continuous Tasks in Dynamic Environment,* Proc. of the SICE Annual Conf., 2002, 932–935.

[12] M.A.S. Kamal, J. Murata and K. Hirasawa, *Task-Oriented Multiagent Reinforcement Learning Control for a Real Time High-Dimensional Problem,* Proc. of the Int. Symposium on Artificial Life and Robotics (AROB) vol. 2, 2003, 353–356.

[13] T. Malone, Kevin Crowston, Toward an interdesciplinary theory of coordination, Center for Coordination Science Technical Report 120, MIT Sloan School of Management, 1991.

[14] G. Miller and D. Cliff, Co-evolution of pursuit and evasion I: Biological and game-thoretic foundations, Techinical Report CSRP311, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1994.

[15] M.V.N. Prasad and V.R. Lesser, Learning situation-specific coordination in cooperative multi-agent systems, *Autonomous Agents and Multi-Agent Systems* **2** (1999), 173–207.

[16] S. Sen, M. Sekaran and J. Hale, *Learning to coordinate without sharing information,* Proc. of the twelfth National Conf. on Artificial Intel., AAAI Press, 1994, 426–431.

[17] J.W. Sheppard, *Multi-agent Reinforcement Learning in Markov Game,* Ph.D. dissertation, John Hopkins Univ. Baltimore, MD, 1997.

[18] S.P. Singh, *Learning to Solve Markov Decision Processes,* Ph.D. dissertation, Dept. Comput. Sci. Univ. Mass., Boston, 1994.

[19] T. Sugawara and V. Lesser, Learning to improve coordinated actions in cooperative distributed problem-solving environments, *Machine Learning* **33**(2/3) (1998), 129–154.

[20] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction,* MIT press, 1998.

[21] M. Tan, *Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents,* Proceedings of the Tenth International Conference on Machine Learning, 1993, 330–337.

[22] C.J.C.H. Watkins and P. Dayan, Q-learning, *Machine Learning* **8** (1992), 279–292.

[23] S.D. Whitehead, *A Complexity Analysis of Cooperative Mechanisms in Reinforcement Learning,* Proc. of AAAI, 1991, 607–613.

[24] D.H. Wolpert and K. Tumer, An Introduction to Collective Intelligence, Technical Report NASA-ARC-IC-99-63, NASA Ames Center, 1999.

[25] D.H. Wolpert and K. Tumer, Optimal Payoff Functions for Members of Collectives, *Advances in Complex Systems* **4**(2–3) (2001), 265–279.

[26] G. Weiss, ed., *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence,* The MIT Press, 1999.