The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Cooperative Multi-Robot Reinforcement Learning: A Framework in Hybrid State Space

Xueqing Sun, Tao Mao, Jerald D. Kralik, Laura E. Ray

Abstract—**In the area of autonomous multi-robot cooperation, much emphasis has been placed on how to coordinate individual robot behaviors in order to achieve an optimal solution to task completion as a team. This paper presents an approach to cooperative multi-robot reinforcement learning based on a hybrid state space representation of the environment to achieve both task learning and heterogeneous role emergence in a unified framework. The methodology also involves learning space reduction through a neural perception module and a progressive rescheduling algorithm that interleaves online execution and relearning to adapt to environmental uncertainties and enhance performance. The approach aims to reduce combinatorial complexity inherent in role-task optimization, and achieves a satisficing solution to complex team-based tasks, rather than a globally optimal solution. Empirical evaluation of the proposed framework is conducted through simulation of a foraging task.**

I. Introduction

Considerable attention and growing interest has been given to the problem of cooperation in multi-robot systems due to the wide range of applications. On-going challenges in searching for effective learning algorithms in a large state space have been defined by many researchers, e.g., [1-5]. With multiple robots performing joint actions in a shared environment with various degrees of uncertainties, coordinating actions in the presence of other robots becomes a key issue in searching for optimal behaviors of individual robots based on the shared task objective. The difficulties encountered are intensified by little or no communication during real-time execution. However, the fact that humans and other intelligent animals can perform quite well in such settings sheds light on developing effective collaboration strategies.

Besides humans, coordinated behavior is seen in many other animal species, especially during pack or group hunting; and although less prevalent in the animal kingdom, collaborative behavior—in which individuals assume different roles—has been documented in species ranging from Harris Hawks to humans. Perhaps the strongest examples of collaborative behavior in non-human animals are seen with both dolphins and chimpanzees. Dolphins perform different collaborative fishing strategies, such as driving fish toward an awaiting line of ambushers [6]; chimpanzees appear to take on several roles when hunting colobus monkeys, including a driver that keeps the monkey moving forward, side blockers, chasers and an ambusher [7]. Although the underlying abilities required for cooperation appear to be innate, it is clear that there is a strong learning component as well.

The theory and techniques of Reinforcement Learning (RL) have been widely adopted and applied in a variety of problem domains including robotics [8-12]. In some applications, such as group foraging [1] or motor primitive planning [12], it has proven to be an effective methodology by which agents learn by taking actions and receiving reward or punishment from their environment. RL algorithms provide optimization of a long term reward based on the interaction between the agent and environment. A classic issue in RL is the tradeoff between exploration and exploitation, in which agents must balance exploring the unknowns and exploiting "the current best practice" based on the past experience. Although similar tradeoffs exist in both single- and multi-agent settings, it poses a greater challenge for the latter in the sense that each agent's exploration-exploitation choice not only influences its own but also the other agents' environments.

Another important issue in RL is the size of the state space. With increased complexity of the task, the state space grows excessively, requiring unreasonable computation. Furthermore, if a task is complex enough, it may require agents to choose from a set of behavior types (i.e., roles) dynamically in order to reach the goal, which can be referred to as role emergence. Traditionally role emergence is treated as an independent reinforcement learning problem as in certain game theory and behavior-based robotics applications [3, 13-15].

In this paper, we propose a general hybrid state space that accounts for both the task-related objective state and the agent role state in one unified framework. This general state space representation enables the integration of learning for both action choices and role emergence. The main RL module within the framework utilizes Q-learning [16] because it is a model-free, asynchronous and anytime algorithm with simplicity and speed. As the number of object and robot entities increase, it imposes a challenge to learn in a large state space. To reduce the state space, a neural perception module is developed to categorize objects (tasks) into clusters and group robots by role types. As the level of state abstraction increases, certain details, such as individual robot identities or the optimality based on these details are abstracted to search for a feasible, sub-optimal but satisficing solution. This neural perception algorithm is

Manuscript received Feb 28, 2009. This work was supported by the Office of Naval Research under Grant No. N00014-08-1-0693.

Xueqing Sun, Tao Mao, and Laura E. Ray are with Thayer School of Engineering, Dartmouth College, Hanover, NH 03755 USA (e-mail: {xueqing.sun, tao.mao, laura.e.ray}@dartmouth.edu)

Jerald D. Kralik is with Department of Psychological & Brain Sciences, Dartmouth College, Hanover, NH 03755 USA (e-mail: jerald.d.kralik@dartmouth.edu)

joined by a progressive rescheduling policy for performance enhancement and uncertainty adaptation.

As a motivating example, consider a simplified multi-robot foraging application where a robot team is assigned to a field to collect scattered objects. There are different roles that heterogeneous robots can take on dynamically during execution with little or no communication. Each robot needs to decide not only on the most efficient path to collect the objects, but also on which role(s) to take on in order to reach the team goal with optimal or near-optimal performance – shortest completion time, for example.

The paper is organized as follows. Section II gives a brief overview of RL and Q-learning in particular. Section III describes our hybrid state space representation. Section IV presents the structure of the multi-robot cooperation framework and algorithm. Section V presents simulation results and compares the performance of this framework with another approach and in different settings. Finally, Section VI provides conclusions and discusses future work.

II. REINFORCEMENT LEARNING

Reinforcement learning (RL) is a computational approach focused on goal driven learning through an agent's interaction with its environment [17]. RL can be carried out online by receiving a reward or penalty signal from an agent's environment with or without a model. The agent's objective is to maximize reward or minimize penalty in the long run. In the RL domain, an environmental state containing all relevant information for the learning process is said to have the Markov property. For example, in a chess game, the current positions of all chess pieces would serve as the Markov state. A reinforcement learning process is a Markov Decision Process (MDP) if its states satisfy the Markov property. Formally, a MDP includes a state set S, action set A and reward set R, in which a learning sequence is comprised of a current environmental state $s \in S$ from which an action $a \in A$ is taken by the agent generating a reward or penalty signal $r \in R$ received by the agent from the environment or intrinsic source. A value function V is modified by r and the next action is selected based on criteria related to the value function.

RL is different from other areas of machine learning in that an agent must learn from interaction through its own experience; to maximize its long term reward, an agent must not only exploit what it already knows but also explore unknowns while risking lower immediate reward. A large state space and environmental uncertainties pose major challenges in utilizing RL. Q-learning is used here because of its proven success, simplicity and asynchronous learning strategy, which provides an anytime algorithm.

Watkins introduced Q-learning [16], and its convergence was rigorously proven in [18] and more generally in [19, 20]. The core of the Q-learning algorithm is the update of state-action value Q at the end of a sequence of distinct states or time steps [18] as

$$Q_t(s,a) = (1-\alpha)Q_{t-1}(s,a) + \alpha[r + \gamma V_{t-1}(s')] \quad (1)$$

$$V_{t-1}(s') \equiv \max_{a'}\{Q_{t-1}(s',a')\} \quad (2)$$

where s is the current state at time step t, a is the action taken at state s, r is the reward, $\alpha \in (0,1]$ is the learning rate, and $\gamma \in [0,1]$ is the reward discount factor. The learned value function approximates the optimal value function as the algorithm iterates. It has been proven that if all action-value pairs are visited indefinitely often and if

$$\lim_{T\to\infty}\sum_{t=0}^{T}\alpha_t = \infty \quad and \quad \lim_{T\to\infty}\sum_{t=0}^{T}\alpha_t^2 < \infty \quad (3)$$

then the probability of Q_t converging to the optimal action-value function Q^* is one [19, 20]. The optimal policy is defined by,

$$\pi^* = \arg\max_a Q(s,a) \quad (4)$$

Even if all value functions are not updated enough times, Q-learning is an "anytime algorithm", which means we can interrupt the algorithm as needed based on a computation budget and will simply obtain the best policy/solution thus far. This is an important feature compared to methods like Dynamic Programming which sweeps through the whole state set, performing a costly *full backup* operation on each state. Additionally, Q-learning is exploration insensitive, which means a clear model of the environment is not required and value functions Q will converge to the actual values, regardless of the strategy used to determine the actions taken [21].

III. HYBRID STATE SPACE

In multi-robot cooperation, each robot not only needs to deal with task related objects but it also interacts with other robots in order to achieve the team goal. In other words, in each robot's environment, there are two types of states - object states and robot internal states [22]. In this section, we propose a hybrid state space representation integrating both object states and robot role states for reinforcement learning. This allows using one generic Q-learning algorithm to achieve both task planning and role emergence simultaneously. Our definition of a role is a specific set of behaviors that a robot carries out during task execution. Roles can change voluntarily or involuntarily over time (i.e., role emergence). It is different from a robot's capability in the sense that capability is a static set of behaviors that a robot could assume, while a role is the set of behaviors actually assumed. We will show in the next section how robots change their roles voluntarily through Q-learning based on the team goal.

Assume all environment states have Markovian properties. We define a hybrid state space $\tilde{s} \in \mathbb{S}$ as a set of states including both object state o and role state e as

$$\mathbb{S} = \{\tilde{s}_0, \tilde{s}_1, ...\tilde{s}_t, ...\tilde{s}_n\} = \{[o_0, e_0], [o_1, e_1], ..., [o_t, e_t], ...[o_n, e_n]\} \quad (5)$$

Fig. 1 illustrates an example of a hybrid state space construction and state transition that integrates both object states o and role e for unified Q-learning in a multi-robot group foraging task. Here, a team of heterogeneous robots

dynamically assume different roles to collaborate in sweeping and collecting actions in order to achieve the shortest team completion time in collecting and depositing scattered objects. The left section of each state representation in Fig. 1 is the object entities. For example, the values in the circles represent the number of objects for each entity at a certain state, where each entity can be a single object or cluster of objects. The section on the right represents the dynamic robot roles, e.g., P2 is a role in which a robot can pick up two objects or less at once, P1 is a role in which a robot can pick up a single object only, but can also emerge as a sweeper S when needed to cluster objects. The number of robots assuming that role is given in parenthesis, e.g., P2(1) means there is one P2 robot at certain state. In order to reduce the state space and generalize the learning process, we do not distinguish between homogeneous robots, and the location of objects or robots is tracked but is not included in the state space.

IV. A FRAMEWORK IN HYBRID STATE SPACE

In this section, we present a framework for multi-robot cooperation including a neural perception module using a Competitive Learning neural network for state space abstraction, Q-learning using Boltzman distribution for action selection, and a progressive rescheduling policy for refined task allocation to maximize team utility and online re-learning for environmental uncertainties. Our framework is inspired by the structure and function of mammalian brain circuitry in which the operations emerge from the interaction of multiple brain regions [23-25]. The posterior cortex (PC) receives and processes sensory inputs via dorsal thalamus (T), and anterior cortex (AC) produces motor outputs while interacting closely with elements of striatal complex (S, striatum; and P pallidum) [23-25]. Fig. 2 depicts the schematic diagram of our framework. The components follow the general sensation, perception, decision, action, reward feedback cycle, and thus include modules for Environmental Sensation (ES), Neural Perception (NP), Decision Process (DP), Execution (EX) and Reward Sensation (RS). There are primary connections between these individual modules acting as the pathways for input and output signals. Generally similar to how the human brain functions, this framework provides an architecture for robots to sense and reason about how to collaborate in a joint task space with environmental uncertainties towards maximizing global team utility. Each functional module has an analogous brain structure as described below.

ES: In the Environmental Sensation (ES) module, robots sense the outside world, including the task related object states and the behavior related role states of other robots. Based on observation of individual robots or information shared through limited communication among robots, the sensation signals of both object states and role states are packaged and passed to a Neural Perception module for hierarchical representation and hybrid state space construction. The detailed state space description was presented in section II. The ES module is roughly analogous to the sensory processing areas of thalamus (T) in the brain, such as the early processing areas of visual cortex of mammals.

NP: The Neural Perception (NP) module is inspired by knowledge of how the human brain receives, processes and represents its environment through efficient, hierarchical data structures [26]. In this module, the object's physical locations are passed in as input vectors to a Competitive Learning neural network to be classified as single objects vs. clusters, which can be further abstracted into subcategories such as shape and orientation based on processing camera images [27]. Learned categories from this perception module are not only important for the later decision process, but also provide for state space reduction.

The Competitive Learning neural network [28] is an unsupervised learning network with the output layer known as the "competitive layer" (Fig. 3). During learning, the input vector is compared with the weight vector of each neuron that leads to the competitive layer. The neuron with a weight vector most closely matching the input vector has its weight vector adjusted in a "winner takes all" fashion. The process continues until no further weight adjustment is required and

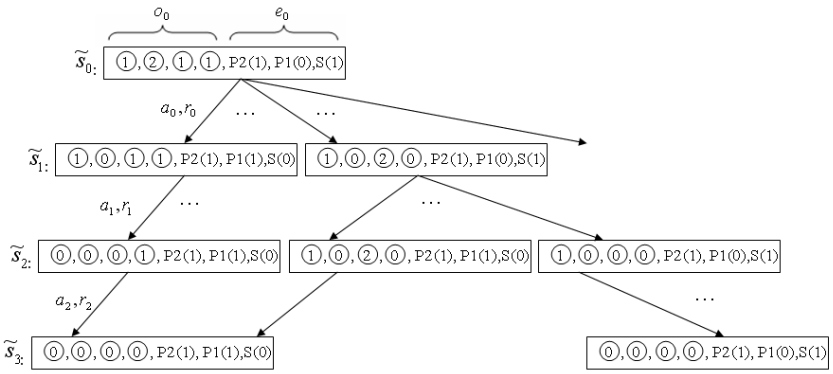

Fig. 1. Illustration of hybrid state space and transition in a group foraging task with two robots and dynamic role emergence.

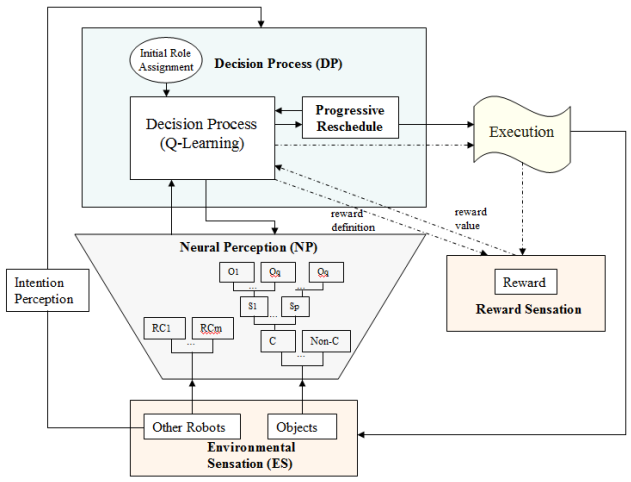

Fig. 2. Schematic diagram of a multi-robot cooperation framework

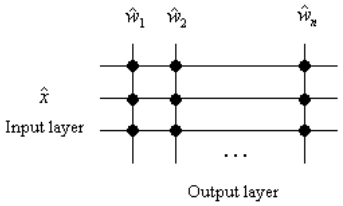

Fig. 3. Competitive learning NN used in clustering.

the final learned weight vectors identify the centers of the clusters. The performance rule for finding winning node j based on a randomly selected normalized vector of object coordinates $\hat{x}$ is given by

$$j \leftarrow \arg\max_{j} \hat{w}^{T} \hat{x} \tag{7}$$

where the learning rule for updating the normalized winning $\hat{w}_j$ with rate η is

$$\hat{w}_j \leftarrow \hat{w}_j + \eta(\hat{x} - \hat{w}_j) \tag{8}$$

Therefore, with each random sample $\hat{x}$, $\hat{w}_j$ is updated and converges to one cluster's center. The NP module is roughly analogous to later perceptual processing areas in mammalian posterior cortex (PC).

DP: With a learned hierarchical representation of the world from the NP module, the Decision Process (DP) is performed in this core module utilizing Q-Learning based on trial and error of different role and object collecting actions with the objective of achieving optimal or near-optimal team utility: for example, the shortest team completion time. This module is closely coupled with the Execution and Reward Sensation module to achieve unified role emergence and path planning (dashed lines). There are two main components in this module. The first is a recursive Q-learning algorithm with a Boltzman distribution, and the second is a progressive rescheduling algorithm that leads to task allocation and online re-learning due to environmental uncertainties. These two components interleave to provide dynamically adapted schedules as robots execute their best found policies in a stochastic environment. The details of the algorithm suite inside the DP module are shown in Table I. The algorithm suite in Table I is a unified decision process that includes both task planning and role emergence in one reasoning engagement. When this algorithm suite is distributed to a system of robots that "think alike," minimal communication is required because each robot already knows paths or roles other robots assume based on the same reasoning strategy as its own. In other words, if each robot has identical state information and brain architecture, each will arrive at the same set of policies for task completion and will learn each other's role and task. While complete, error-free state and policy knowledge without communication is unrealistic, these assumptions can be then relaxed individually to evaluate the effect on learning.

We define a learning budget T_b by which Q-learning will pause learning and return the best policy found so far to the execution module. This is necessary under the circumstances of restricted learning time and it takes the advantage of Q-learning being an anytime algorithm that can be interrupted

TABLE I
ALGORITHM SUITE IN DP MODULE

Algorithm Suite

Initialize:
Let t=0, re-learning flag=TRUE
Neural Perception: clustering by Competitive Learning NN
Initialize hybrid state $\tilde{s}_0 = [o_0, e_0]$
For all $\tilde{s} \in S$, $a \in A$,
let $Q(\tilde{s}, a) = 0$, $V(\tilde{s}) = 0$
Initialize Q -learning budget T_b
While current object state O_t != completion state O_n
If re-learning flag is TRUE
Q-learning:
While under budget T_b
Recursive learning until O_n :
Choose action a at state $\tilde{s}$ based on the probability from Boltzman distribution,

$$\Pr\{a_t = a\} = \frac{e^{Q(\tilde{s},a)/\tau}}{\sum_{b=1}^{n} e^{Q(\tilde{s},b)/\tau}}$$

Observe reward r and next state $\tilde{s}'$
Update $Q(\tilde{s}, a)$ such that,
$Q_t(\tilde{s}, a) = (1-\alpha) Q_{t-1}(\tilde{s}, a) + \alpha[r + \gamma V_{t-1}(\tilde{s}')]$
$V_{t-1}(\tilde{s}') \equiv \max_{a'} \{Q_{t-1}(\tilde{s}', a')\}$
Record current best known policy,
$\pi^* = \arg\max_{a} Q(\tilde{s}, a)$
End
Progressive Reschedule:
Allocate tasks based on best known policy π^* to individual robots using Least Workload algorithm to achieve optimal/sub-optimal completion time
End
If perception at t and t-1 is inconsistent
Trigger re-learning by setting relearning flag=TRUE
Else
Set relearning flag=FALSE
End
End

at any point. When interrupted, it returns the best solution for execution found to the point of interruption.

The purpose of the Progressive Reschedule task allocation block is to further generate a specific task list from the best learned policy for individual robots within the same role category. The goal is to allocate tasks so as to minimize overall team completion time without introducing excessive computation. Since the overall team completion time is determined by the slowest team member (i.e., the longest completion time for an individual task), the objective becomes finding an allocation policy that yields $\min_j(\max_i(T_{ij}))$ where T_{ij} is the individual completion time of the *ith* robot under *jth* allocation policy. In computational complexity theory, finding an optimal solution for this combinatorial optimization problem is NP-hard [29-31]. We propose a simple and fast approximation algorithm that sorts the non-sequential tasks by their cost in descending order and allocates them from top-down to the individual robot with the current "least workload." In the next section, we compare the results between this Least Workload algorithm and traditional Brute Force combinatorial optimization on both overall team completion time and computation.

Finally, inside the Progressive Reschedule relearning block, relearning is triggered by setting a flag based on the detection of an inconsistent perception between the current and previous time steps; for example, an object may move from its original location if it is hit by a robot, or uncertainty in sensor measurements may cause inconsistencies in perception. Relearning allows for execution uncertainty; future addition of heterogeneity among robots, both in perception of the state space and in learned policies; and asynchronous learning and associated relaxation of the assumption that all robots "think alike." In the mammalian and especially the primate brain, the DP module is analogous to higher-order cortical areas such as posterior parietal and anterior cortex (AC), which are involved in decision making. These areas are well-positioned for decision making in that they receive perception and reward information and project to motor control areas.

EX and RS: At the planning stage, the Execution (EX) model takes pseudo actions based on the Q-learning policy and Reward Sensation (RS) module feeds back reward signals for recursive learning (dashed lines). Besides the feedback path, there is another path from the DP to the RS module to adjust the reward definition in case of a potential team goal change, such as changing from shortest team completion time to lowest energy use; at the execution stage, the task lists generated by Progressive Rescheduling are passed to individual robots for execution (solid lines). For the planning stage, EX and RS may be analogous to prefrontal cortical areas, which are used in planning. At the same time, especially during execution, EX is analogous to motor control areas of the brain, such as premotor and primary motor cortex; and RS would be analogous to the dopamine reward system (of the midbrain and ventral striatum) that appears to underlie reward learning.

V. Experimental Results and Discussion

We have conducted a number of Monte Carlo experiments with random environmental configurations to evaluate the effectiveness of our hybrid state space formulation and algorithm suite in a multi-robot foraging task and have also tested extensively in a physically realistic simulation environment using the Webots simulator [32] as shown in Fig. 4 and in the accompanying video.

The first set of experiments is conducted to measure the performance of a multi-robot team in three different role configurations: a Homogenous team of three robots with a Single role of P2 (HoSr), a Heterogeneous but Fixed-role team of two P2 and one S (HeFr) and a Heterogeneous team of two P2 and one S/P1 with dynamic Role emergence (HeRe). In this experiment, each robot can obtain state information provided by Webots simulator about the locations of objects and robots, and it uses the framework described in section IV. Simple policies for lower level motor behaviors, such as right-of-way and manipulating objects are assumed.

Fig. 5 compares the average overall team completion time from 200 trials for each of the three different role configurations based on randomly generated test cases of 12-24 scattered objects. The raw data are shown in black points, and the quartile and mean of each group are indicated by the red box and center line of the green diamond respectively. The HoSr team has the lowest performance compared to the other team configurations. This is mainly due to the lack of diversified roles, highlighting the importance of collaboration. The results also show that teams with dynamic role emergence (HeRe) perform significantly better than teams with static roles. Also note that the best policy after learning budget T_b can be sub-optimal in a large state space, and role emergence will create more possible role states than fixed role scenarios. Nonetheless, HeRe outperforms HoSr and HeFr teams with static roles on average even with a restricted Q-leaning

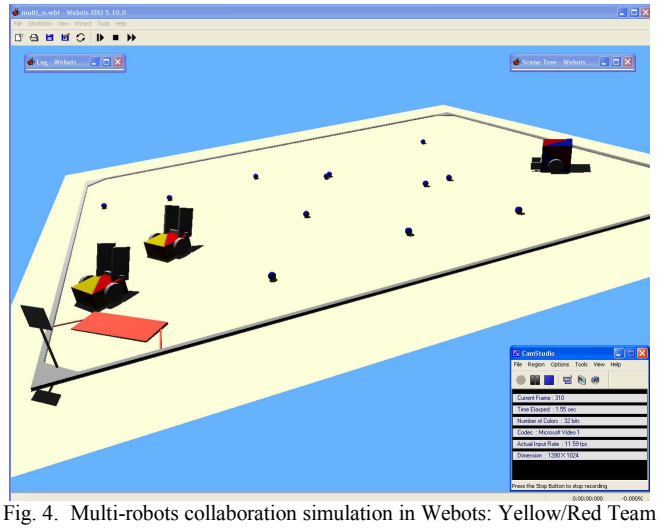

Fig. 4. Multi-robots collaboration simulation in Webots: Yellow/Red Team is composed of two robots that can pick up 1-2 objects at a time; Blue/Red Team is one robot that can either sweep or pick up 1 object.

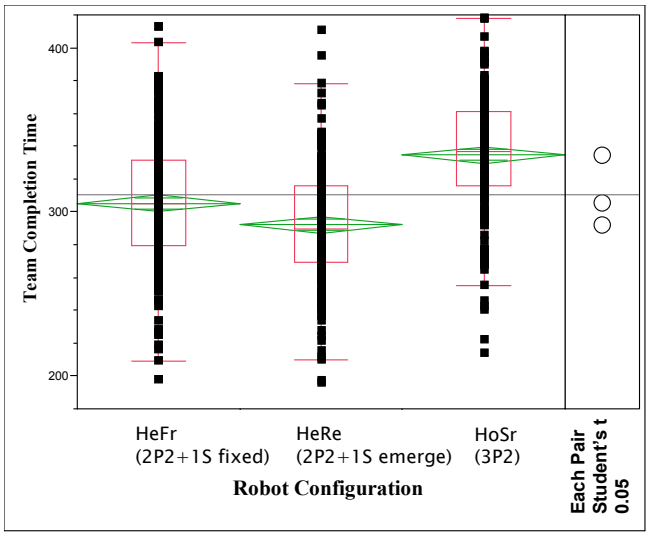

Fig. 5. Overall team completion time for three difference robot team configurations: a heterogeneous but fixed-role team of P2 and S (HeFr), a heterogeneous team of P2 and S/P1 with dynamic role emergence (HeRe) and a homogenous team of single role of P2 (HoSr).

budget T_b that forces Q-learning to stop and simply return the best policy

In the second set of Monte Carlo experiments, we evaluate and compare the Least Workload algorithm with Brute Force optimization in task allocation for robots with the same role type, based on the best found policy from the Q-learning algorithm. To justify the efficacy of the approach, we first focus on a two-robot team. We generate 100 randomly initialized trials for each case of total number of tasks ranging from 1-15 tasks (i.e., 1500 trials in total). Within each trial, a random number$\in$ (0,100) is generated to represent the individual task completion cost in seconds. Fig. 6 shows the mean of overall team completion time and computation cost (on a logarithmic scale) from this set of experiments. As seen in Fig. 6(a), the overall team completion time from the Least Workload approach falls within 2.5% of the optimal solution found by Brute Force independent of the number of tasks. Also, as seen in Fig. 6(b), as the number of tasks to be allocated increases, the computation cost from Least Workload is significantly lower than that of the Brute Force approach which has exponential time complexity.

We also extend the robot team from two robots to five with total number of tasks ranging from 5-15 and total number of trials in a normal distribution. Similar to Fig. 6, Fig. 7 shows that with the increase in number of robots, the overall team completion time from the Least Workload approach still falls within 2.5% of the optimal solution, and the computation cost remains significantly lower than the Brute Force algorithm.

VI. Conclusion and Future Work

In this paper, we have described a hybrid state space representation, framework, and learning algorithm inspired by human brain function. Using reinforcement learning in this framework enables robots to solve role emergence and task allocation simultaneously and to relearn in the face of environmental uncertainties that arise during task execution. The experimental results presented here demonstrate the importance of role emergence and the efficiency of our algorithm on enhancing team performance in a collaborative task.

While the framework is general, our current reinforcement learning process defines the end of each episode by the subtask with longest completion time (i.e., the slowest team member), which could introduce noise in the reward calculation. We plan to investigate a sub-state representation in the reward calculation that is asynchronous which will

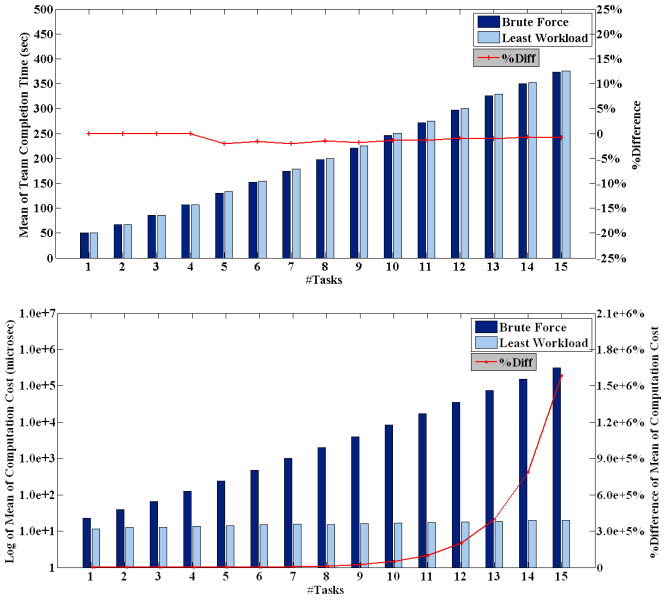

Fig. 6. Comparison of Brute Force vs. Least Workload (2 Robot Team) (a) Mean of team completion time by # tasks (1-15), (b) Log of Mean of computation cost by # tasks (1-15)

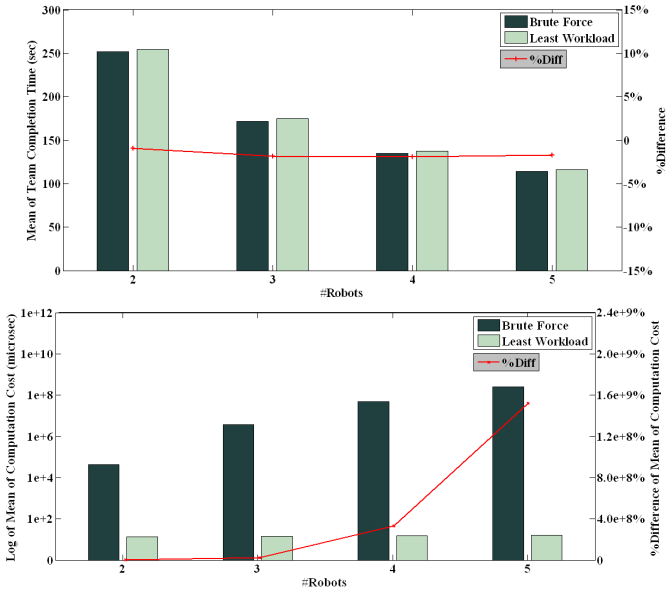

Fig. 7. Comparison of Brute Force vs. Least Workload (2-5 Robot Team) (a) Mean of overall team completion time by number of robots, (b) **Log** of Mean of computation cost by number of robots(2-5)

refine the reward calculation, albeit at risk of increasing the size of the state space.

Another area for future work is scalability of the current learning process with uncertainty. In the present representation, every robot "thinks alike" without error or uncertainty. Future modifications will allow for uncertainty in each of the ES, NP, and DP processes, allowing for heterogeneity in learning among robots and a distributed version of the framework. Subsequent work will implement and conduct experiments with Pioneer robots to evolve the method to accommodate asynchronous learning and uncertainty, such as error in perception of object or role states or location.

REFERENCES

[1] Matarić M. J. (1997). Reinforcement Learning in the Multi-Robot Domain. Autonomous Robots 4, pp. 73-83.

[2] Goldsmith S. & Robinett. R. (1998). Collective search by mobile robots using alpha-beta coordination. Conference on Collective Robotics Workshop, Paris, France, 4-7 Jul 1998, pp. 136–146.

[3] Martinson E. & Arkin R. C. (2003). Learning to role-switch in multi-robot systems. International Conference on Robotics and Automation, pp. 2727–2734.

[4] Powers R. & Shoham Y. (2005). New criteria and a new algorithm for learning in multi-agent systems. In Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS), Vancouver, Canada.

[5] Velagapudi P., Scerri P., Sycara K., Wang H., Lewis. M., Wang J. (2008). Scaling effects in multi-robot control. In Proceedings of Intelligent Robots and Systems (IROS).

[6] Wells, R. S. (2003). Dolphin social complexity: Lessons from long-term study and life history. In F.B.de Waal & P. L. Tyack (Eds.), Animal social complexity. Cambridge, MA: Harvard University Press.

[7] Boesch, C. (2003). Complex cooperation among Taï chimpanzees. In F.B.de Waal & P. L. Tyack (Eds.), Animal social complexity. Cambridge, MA: Harvard University Press.

[8] Littman M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, pp. 157-163.

[9] Hu J., Wellman M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufman, San Francisco, pp. 242-250.

[10] Chalkiadakis G. & Boutilier C. (2003). Coordination in multiagent reinforcement learning: a bayesian approach. In Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 709–716.

[11] Zhang X., Aberdeen D., Vishwanathan S.V.N. (2007). Conditional random fields for multi-agent reinforcement learning. In Proceedings of the 24th International Conference on Machine Learning (ICML), ACM International Conference Proceeding Series, Corvalis, Oregon, pp. 1143–1150.

[12] Peters J., Vijayakumar S., Schaal S. (2008). Natural actor-critic. Neurocomputing 71, pp. 1180-1190.

[13] Claus C. & Boutilier C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA.

[14] Asada M., Uchibe E., Hosoda K. (1999). Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. Artificial Intelligence, vol. 110, pp. 275-292.

[15] Liu, Z., Ang Jr., M. H., and Seah, W. K. G. (2005). Reinforcement learning of cooperative behaviors for multi-robot tracking of multiple moving targets. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1289–1294.

[16] Watkins, C.J.C.H. (1989). Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England.

[17] Sutton R. S. & Barto A. G. (1998). Reinforcement Learning. MIT Press, Cambridge, MA.

[18] Watkins C. J.C.H. & Dayan P. (1992). Technical Note: Q-Learning. Machine Learning, Volume 8, Numbers 3-4, May, pp. 279-292.

[19] Jaakkola, T., Jordan, M. I., Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. Neural Computation, 6(6), pp. 1185-1201.

[20] Tsitsiklis J. (1994). Asynchronous stochastic approximation and Q-learning. Machine Learning

[21] Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4, pp. 237-285.

[22] Jones C., Shell D., Matarić, M. J., Gerkey, B. (2004). Principled approaches to the design of multi-robot systems. In Proceedings of the Workshop on Networked Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[23] Granger R. (2006). Engines of the brain: The computational instruction set of human cognition, AI Magazine 27(2), pp. 15-31.

[24] Rodriguez A., Whitson J., Granger R. (2004). Derivation & analysis of basic computational operations of thalamocortical circuits. Journal of Cognitive Neuroscience 16. pp. 856-877.

[25] Shimono K., Brucher F., Granger R., Lynch G., Taketani M. (2000). Origins and distribution of cholinergically induced beta rhythms in hippocampal slices. Journal of Neuroscience 20. pp. 8462-8473.

[26] Hearn R. & Granger R. (2008). Learning Hierarchical Representations and Behaviors. Association for the Advancement of Artificial Intelligence.

[27] Nowak, S.C. (2009). Vision-Based Perception for Reinforcement Learning in Multi-Robot Systems. Diplomarbeit, Helmut-Schmidt University, Germany.

[28] Duda R. O., Hart P. E., Stork D. G. (2005). Pattern Classification. pp. 560-561

[29] Cormen, T. H., Leiserson, C. E., Rivest, R. L. (1990). Introduction to Algorithms. MIT Press/McGraw-Hill.

[30] Woeginger G. J. (2003). Exact Algorithms for NP-Hard Problems: A Survey", *Combinatorial Optimization – Eureka, You Shrink! Lecture Notes in Computer Science, vol. 2570*, Springer, pp. 185–207.

[31] Koes M., Nourbakhsh I., Sycara K. (2005). Heterogeneous multirobot coordination with spatial and temporal constraints. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*. AAAI Press, pp. 1292–1297.

[32] Webots Reference Manual. Cyberbotics Ltd. Professional Mobile Robot Simulation Software. http://www.cyberbotics.com.