Hierarchical Multi-Agent Reinforcement Learning for Dynamic Coverage Control

Hossein Hajimirsadeghi

Abstract—**In this study, an automatic dynamic coverage control problem is introduced and a hierarchical multiagent reinforcement learnig (HMRL) algorithm based on MAXQ framework is employed to solve the problem. Hierarchical reinforcement learning (HRL) is a state-of-the-art topic in domain of multi-agent reinforcement learning, and most of the investigations in HRL have been engaged with theoretical issues to develop the mathematical basics and clarify the concepts. In the present work, we consider an applied problem and investigate the performance of the proposed learning algorithms. Simulation results show that the presented hierarchical framework is much more efficient that the flat cases in both quality of learning and speed of convergence.**

Index Terms—**Hierarchical Reinforcement Learning, Multi-Agent Systems, Coverage Control, Cooperative Control.**

I. Introduction

Reinforcement learning which is defined as learning from interaction, has achieved high success in solving problems in machine learning. However, the tabular reinforcement learning methods suffer from the curse of dimensionality which means the exponential growth of computational and memory requirements with the number of state variables. It is more challenging when we are facing the problems in multiagent systems where the number of states increases drastically by the number of agents.

One of the approaches to come up with the curse of dimensionality is hierarchical representation of problems to changing the flat space to a hierarchy of simpler spaces and introduces mechanisms for abstraction and sharing of subtasks. Using a hierarchy in reinforcement learning has three main benefits. First, learning will be done in fewer trials because fewer parameters must be learned. In addition, abstraction can be used to ignore irrelevant information in states of subtasks (it can help in the case of partially observable agents or lack of complete sensing). Also, learned subtasks can be shared between some parent subtasks. Second, learned subtasks can be reused for a new problem. Third, the exploration will be improved due to existence of high-level actions that can search a larger space in one step.

Report was prepared March 6, 2009.

H. Hajimirsadeghi is a M.Sc. student of Control Engineering in the school of Electrical and Computer Engineering, University of Tehran, (Std. No. 810187579; e-mail: h.hajimirsadeghi@ece.ut.ac.ir).

In multiagent systems, the hierarchical framework can help to decrease the cost of coordination and communication, i.e., in many applications, the agents need to be aware of other agents' actions only in high levels of decision making. Also, the partially observability of the agents about states of each other can be addressed in a hierarchical representation.

As it is stated in [5] there has been insufficient experience in experimentally testing the effectiveness of the ideas in HRL on large applications. In this study, we employ hierarchical multiagent reinforcement learning procedure described in [1] and [2] in an automatic coverage control problem which is known to be an important problem in the field of cooperative control. Cooperative HRL and Selfish Multiagent HRL algorithms are used and their performance is evaluated in a test problem.

Section II provides a quick review of Hierarchical Reinforcemnt Learning (HRL) with more emphasis on MAXQ approach. In section III, the coverage control problem is described and the hierarchical framework for that is presented. In section IV, a test problem is introduced and the simulation results are summarized. Finally, the conclusions are drawn in section V.

II. Hierarchical Multi-Agent Reinforcement Learning

A. *Hierarchical Reinforcement Learning*

Research in HRL is based on the formalism of *Semi-Markov Decision Processes* (SMDPs), which is an extension of the Markov Decision Process (MDP) formalism. SMDPs allow for temporally extended actions, i.e. actions that can take variable amounts of time as opposed to a fixed interval. Action selections are made at distinct epochs in time, and the state of the system may continue changing during the action. An SMDP is defined as a quintuple $M = \langle S, A, P, R, T \rangle$, where S is a finite set of states (the state space), A is a finite set of actions (the action space), P is the transition probability function $P: S \times A \times S \times \mathbb{N} \to [0,1]$, R is the reward function $R: S \times A \times S \to \mathbb{R}$, and T is the transition time function $T: S \times A \times S \to \mathbb{N}$. The transition probability function $P(s', N | s, a)$ is the probability of transitioning to state s' in N time steps, given that action a is taken in state s'. The reward function $R(s'|s,a)$ is the real-valued reward for taking action a in state s and reaching state s'. The transition time function $T(s'|s,a)$ is the completion time for taking action a in state s and reaching state s'. A policy π is a function $\pi: S \to A$, defining what action the agent takes in any given state.

One of the most commonly used approaches in hierarchical reinforcement learning is MAX-Q method. MAXQ approach works with decomposing the whole task into a set of subtasks, which are in turn decomposed into smaller subtasks. This structure forms a hierarchy whose leaves are primitive actions. This method is analogous to the introduction of subroutines in programming, but the order in which subtasks are executed is arbitrary. Once the programmer defines the hierarchy, this is the reinforcement learning system that will write the code for each subroutine. Each subtask has some termination conditions. These are the conditions that once fulfilled the control of program returns to the parent subtask. Termination conditions are not necessarily desirable conditions. For example, an inappropriate invoking of a subtask by the reinforcement learning system can also bring it to a termination condition. The desired subset of termination conditions, i.e. the conditions that show the successful invoking and performing of the subtask, are called goals.

A hierarchy can be represented by a task graph. The first component is the expected total reward received while executing action a in state s and following policy π denoted by $V^{\pi}(a,s)$, and the second component is the expected total reward of completing parent task M following policy π after a finished denoted by $C^{\pi}(M,s,a)$ and called completion function. Thus, we have:

$$Q^{\pi}(M,s,a) = V^{\pi}(a,s) + C^{\pi}(M,s,a) \tag{1}$$

where

$$C^{\pi}(M,s,a) = \sum_{s',N} P_M^{\pi}(s',N|s,a)\gamma^n Q^{\pi}(M,s',\pi(s')) \tag{2}$$

and

$$V^{\pi}(a,s) = \begin{cases} Q^{\pi}(a,s,\pi(s')) & if\ a\ is\ composite \\ P(s'|s,i)R(s'|s,i) & if\ a\ is\ primitive \end{cases} \tag{3}$$

Equation (1) shows the relation of action-value functions of a parent task to the action-value functions of its child tasks. Applied recursively, it shows how we can decompose the action-value function of the root task into summation of action-value functions of its descendant subtasks. In Theorem 2 of [4], it is shown that this decomposition can represent the value function of any hierarchical policy. The most popular learning algorithm for MAXQ decomposition is MAXQ-Q learning algorithm [3], [4], that we use the multiagent form of that described in [1] and [2].

B. *Hierarchical Multi-Agent Reinforcement Learning*

Makar et al. modified single agent framework of MAXQ approach for multiagent systems. In this respect, two learning approaches can be considered: selfish case and cooperative case. In selfish case, the learning agents learn with the given MAXQ structure but make no attempts to communicate with each other while in the cooperative case, the MAXQ structure is modified such that the Q nodes at the cooperation levels include the joint actions done by all the agents. The subtasks in cooperation level are called cooperative subtasks and the cooperation level is defined to be the level in which coordination among agents is of great importance. Cooperation levels are usually placed at the highest levels of the task graph.

Based on MAXQ function decomposition of the Q-functions with joint actions for cooperative subtasks, a learning algorithm is proposed in [1] called Cooperative HRL algorithm which is provided in Appendix A.

III. Problem Description

A. *Coverage Control Problem*

Consider a set of sensors $S = \{S_1, S_2, \dots, S_{N_s}\}$ located in an area in order to estimate a set of desired variables $V = \{V_1, V_2, \dots, V_{N_v}\}$ for a set of targets $T = \{T_1, T_2, \dots, T_{N_t}\}$. Each sensor S_i is specialized to estimate one variable in its finite range of detection b_i. A number of stations (agents) are distributed in the area to perform the task of switching the sensors on or off based on the request of a central unit, i.e., the center asks for sensing coverage of a specified variable in all the targets based on its requirements, and the agents make the sensors on to fulfill the task and then inform the center that the coverage is completed. Each agent has access to a limited subset of sensors in S and its sensors might have different characteristics like better estimation quality or larger range of detection that make the agents heterogeneous. In this section, we want to devise a learning algorithm to enhance the proficiency of autonomous agents for switching the sensors based on the requests. It is assumed that the agents receive a binary (to cover or not to cover) signal from the central unit for coverage of each variable. Moreover, they can sense the coverage status for each pair of target and variable. The agents are unaware of the relations between the signals and the variables, and they are to find out the relations through the learning process. Also, they should learn to switch the minimum number of sensors for covering all the targets as well as to inform accomplishment of the task to the center accurately and instantly. Since the sensing coverage is performed by multiple agents, the coordination between the agents is of great importance.

In the flat form, the state space for each agent consists of 2 values for center request of each variable (2^{N_v}), 2 values for coverage status of each target and variable pair ($2^{N_t N_v}$). Hence the total number of states is $2^{N_v} \times 2^{N_t N_v}$, and consequently there are $2^{N_v} \times 2^{N_t N_v} \times (N_s^i + N_v + 1)$ values for the Q-table of each agent (possible actions for each agent are switching the sensors on, informing the center about coverage accomplishment of each variable, and the remaining action is NOP (no operation)). Where N_s^i is the number of sensors for agent i. With the hierarchical framework we show that the number of learning values is considerably reduced.

B. *HARL Framework*

In this part a hierarchical reinforcement learning framework is presented for the proposed coverage control problem. The hierarchical model with subtasks and primitive actions are illustrated in Fig. 2. Generally, there are three composite subtasks consists of *Root*, *Perform* (V_i), and *Cover* (V_i). *Perform* (V_i) represents the subtask of covering the variable V_i in all targets and then informing the central unit. *Cover* (V_i) means to switch the sensors for variable V_i on so that the V_i is covered in all the targets. The primitive actions are the same as the flat case described above. *Root* is defined as a cooperative subtask, and the highest level of the hierarchy as the cooperative level. So, all the subtasks at the second level of hierarchy (*Perform* (V_i) and *NOP*) belong to set U_1(children of cooperative subtask) and coordination among the agents is fulfilled by using joint action-values at the *Root* level as explained in section II.

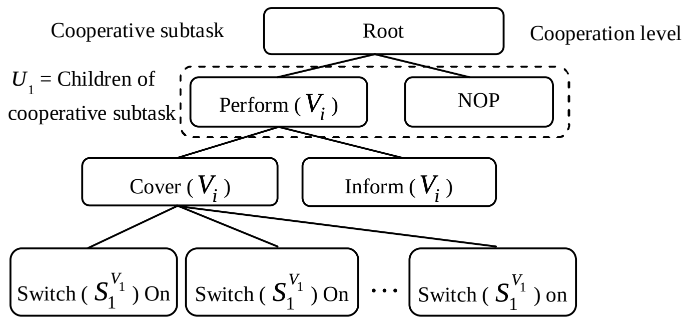
Cooperative subtask
Root
Cooperation level
U_1 = Children of cooperative subtask
Perform (V_i)
NOP
Cover (V_i)
Inform (V_i)
Switch ($S_1^{V_1}$) On
Switch ($S_1^{V_1}$) On
…
Switch ($S_1^{V_1}$) on

Fig. 1. Task graph for multi-agent dynamic coverage control problem

State abstraction can reduce the state space for this problem considerably, i.e., only the relevant states are used for each node of the task graph. We employed state abstraction as follows. For *Cover* (V_i) subtask only the coverage status of variable V_i in the targets are relevant. For the highest level subtask *Perform* (V_i) only the coverage status of variable V_i in the targets and the request signal from the center are relevant. Hence, the state abstraction can be employed to reduce the number of learning values in C and V functions and speed up the algorithm. Similar to the flat case, there is a default reward of -1 for every primitive action and a positive reward is given by the central unit to the agents who correctly inform coverage of a desired variable. In this study this reward is equal to $\frac{200}{N^V_{on\_sensors}}$ where $N^V_{on\_sensors}$ denotes the number of on-sensors of desired variable V in the area.

IV. Simulation Studies

A. Test Problem

In this part, we introduce a test problem for the coverage control problem described above. There are 5 targets and 2 variables, e.g., temperature (T) and pressure (P). Thirty two sensors are devoted to each agent, sixteen of that are temperature sensors and the remaining are pressure sensors. In this problem, it is assumed that the agents do not have common sensors. This configuration of sensors in the area is illustrated in Fig. 2, and the task graph for this test problem is presented in Fig. 3.

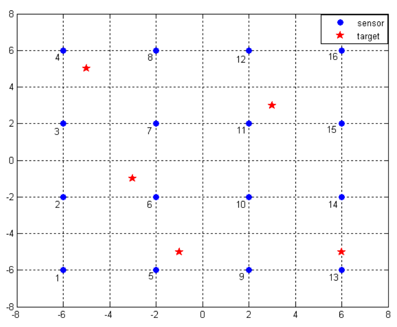

Fig. 2. Configuration of sensors and targets in x_y plane.

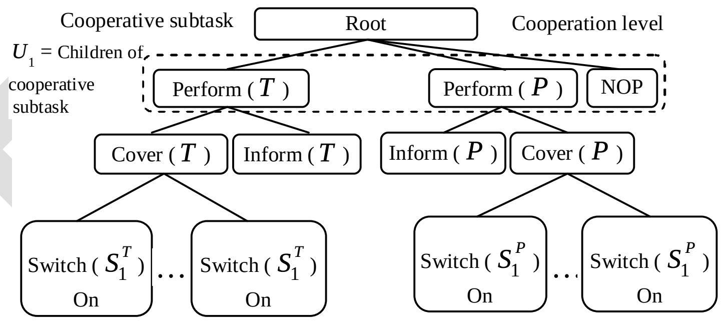

Fig. 3. Task graph for the test problem

Each node in the figure represents four sensors, two of them are T and P sensors for agent 1, and the others are T and P sensors for agent 2. In fact, in this problem the sensors are uniformly distributed in the area but this symmetry is just for simplicity of simulating the environment and is not required for the learning process we propose in the next part. In this test problem the metric that makes our agents heterogeneous is the finite range of detection for their sensors which is characterized in Table I. The sensing coverage is performed in finite rage of sensors which is formulated in (4).

$$\begin{cases} \boldsymbol{x} \text{ is covered by } S_j & \|\boldsymbol{p}_j - \boldsymbol{x}\| \le b_j \\ \boldsymbol{x} \text{ is not covered by } S_j & o.w. \end{cases} \tag{4}$$

where $\boldsymbol{x}$ can be any point in the pilot area and $\boldsymbol{p}_j$ is the location of the sensor S_j. One can see that agent 1 has more powerful sensors for sensing temperature than pressure while agent 2 is better provided by pressure sensors. It can be shown simply that agent 1 can cover the temperature variable for all the targets with three sensors (sensors $S^1_{T,3}$, $S^1_{T,10}$, $\{S^1_{T,11}$ or $S^1_{T,16}\}$) at the minimum while it needs five sensors for pressure coverage. Likewise, agent 2 can perform pressure coverage with at least three sensors ($S^2_{T,7}$, $S^1_{T,9}$, $\{S^1_{T,11}$ or $S^1_{T,15}\}$) while it needs five sensors for temperature coverage.

TABLE I
FINITE RANGE OF DETECTION FOR THE SENSORS OF AGENT1 AND AGENT2

Agent 1				Agent 2			
T sensors	range	*P* sensors	range	*T* sensors	range	*P* sensors	range
1	8	1	8	1	8	1	8
2	8	2	8	2	8	2	8
3	25	3	8	3	8	3	8
4	8	4	8	4	8	4	8
5	8	5	8	5	8	5	8
6	8	6	8	6	8	6	8
7	8	7	8	7	8	7	25
8	8	8	8	8	8	8	8
9	8	9	8	9	8	9	25
10	25	10	8	10	8	10	8
11	8	11	8	11	8	11	8
12	8	12	8	12	8	12	8
13	8	13	8	13	8	13	8
14	8	14	8	14	8	14	8
15	8	15	8	15	8	15	25
16	25	16	8	16	8	16	8

B. *Simulation Results*

In this part, detailed experimental results on the coverage control problem is presented, comparing several learning algorithms, including Cooperative HRL, selfish multi-agent HRL, decentralized Q-learning and centralized Q-leaning. The number of learning values for the flat case is 143,360 for decentralized Q-learning and 272,432 for centralized Q-learning. With state abstraction of the proposed hierarchical framework, the number of learning variables (in C and V functions) is reduced to 14,600 for cooperative HRL algorithm. The discount factor is set to 0.99, the learning rate α is 0.1 and the value of ε for ε-greedy policy of the learning algorithms is initialized at 0.1 and decreases by a factor of $\frac{1}{1+\varepsilon}$ every 100 steps. Note that we did not try to find the best parameters for each algorithm. In order to have better interpretation of the results, the first experiment is executed with the request signal generation as follows: the request signal for both T and P variables are set to 1 at the first of each learning episode and no new request is invoked until all the tasks are satisfied, after which the requests are refreshed to 1. In this experiment, the goal is to increase the throughput of the system which is measured by the number of successful accomplishment per 5000 steps. The results are averaged over five trials and demonstrated in Fig. 4. Also, the best trial among the five trials for Cooperative HRL agent and Selfish Multiagent HRL agent are selected and provided in Fig. 5.

Fig. 4 and Fig. 5 show that Cooperative HRL has a good performance and the algorithm has found the best order of setting the sensors and informing the center. Indeed, the throughput of system approximately converges to 1250 successful accomplishment in 5000 steps which is equivalent to switching on the minimum number of the sensors by each agent (as was discussed it can be achieved by three sensors) plus one action of informing the center, i.e. $\frac{5000}{4} = 1250$. Actually, we used the above-mentioned request signal generation mechanism to be able to calculate the throughput of the system. The abrupt raise or fall of learning performance in selfish multi-agent HRL might be because of lack of coordination that can happen when policy of one of the agents changes. The poor performance of centralized multi-agent Q-learning is because of the large number of learning values which decrease speed of learning. Fig. 6 shows that learning can be improved for centralized multi-agent Q-learning algorithm with larger values of ε, however, learning still is not mature with the specified number of steps.

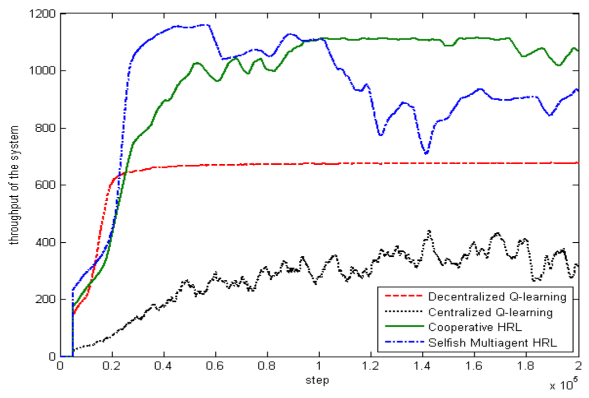

Fig. 4. Comparison between throughput of different algorithms

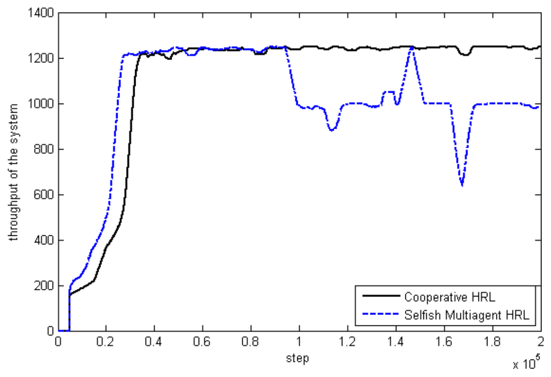

Fig. 5. Configuration of sensors and targets in x_y plane.

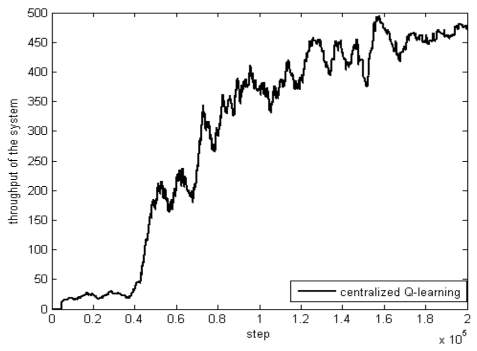

Fig. 6. Throughput of the system for centralized Q-learning with larger ε.

Finally, we performed an experiment with random request signal generator and the results for Cooperative HRL algorithm is depicted in Fig. 7.

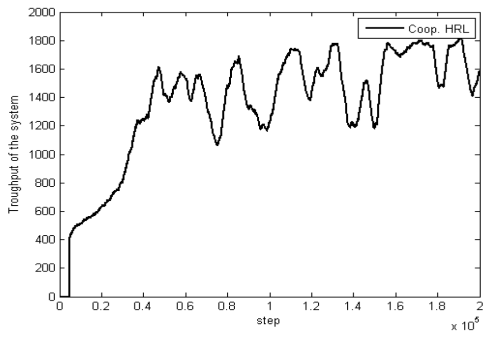

Fig. 7. Throughput of a Cooperative HRL agent in a random request signal generation system

V. Conclusion

In this study, a dynamic coverage control problem with autonomous agents and a set of sensors was described. A hierarchical framework for the process of learning in agents was presented, and two hierarchal multiagent reinforcement learning algorithms were employed to solve the problem. We compared the results of hierarchical algorithms (Cooperative HRL and Selfish Multiagent HRL) with multiagent classical RL algorithms for flat cases including centralized and decentralized Q-learning. The hierarchical algorithms were much superior to the flat algorithms in both learning speed and throughput of the system. Also, the hierarchical approach could help in significant decrease of state space and learning variables.

References

[1] M. Ghavamzadeh, S. Mahadevan, and R. Makar, "Hierarchical multi-agent reinforcement leaning," *Auton. Agents Multi-Agent Syst.*, vol. 13, no. 2, pp. 197–229, 2006.

[2] R. Makar, S. Mahadevan, and M. Ghavamzadeh, "Hierarchical multi-agent reinforcement leaning," in *Proc. Fifth International Conference on Autonomous Agents,* pages 246-253, 2001.

[3] T.G. Dietterich, "The MAXQ Method for Hierarchical Reinforcement Learning," in *Proc. Fifteenth International Conference on Machine Learning,* 1998, pp. 118-126.

[4] T.G. Dietterich, "The MAXQ Method for Hierarchical Reinforcement Learning," *Journal of Artificial Intelligence Research*, vol. 13, 2000, pp. 227-303.

[5] A.G. Barto, and S. Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning," In Discrete Event Dynamic Systems: Theory and Applications, vol. 13, 2003, pp. 41-77.

I. Appendices

A. The Cooperative HRL Algorithm [1]

```
1: Function Cooperative-HRL(Agent $j$, Task $M_i$ at the $l$th level of the hierarchy, State $s$)
2: let $Seq = \{\}$ be the sequence of (state-visited, actions in $\bigcup_{k=1}^{L} U_k$ being performed by the other agents) while executing $M_i$    /* $L$ is the number of levels in the hierarchy */
3: if $M_i$ is a primitive action then
4:   execute action $M_i$ in state $s$, receive reward $r(s'|s,i)$ and observe state $s'$
5:   $V_{t+1}^j(i,s) \longleftarrow [1-\alpha_t^j(i)]V_t^j(i,s) + \alpha_t^j(i)r(s'|s,i)$
6:   push (state $s$, actions in $\{U_l | l$ is a cooperation level$\}$ being performed by the other agents) onto the front of $Seq$
7: else    /* $M_i$ is a non-primitive subtask */
8:   while $M_i$ has not terminated do
9:     if $M_i$ is a cooperative subtask then
10:      choose subtask $a^j$ according to the current exploration policy $\pi_i^j(s,a^1,\dots,a^{j-1},a^{j+1},\dots,a^n)$
11:      let ChildSeq = Cooperative-HRL($j$, $a^j$, $s$), where ChildSeq is the sequence of (state-visited, actions in $\bigcup_{k=1}^{L} U_k$ being performed by the other agents) while executing subtask $a^j$
12:      observe result state $s'$ and $\hat{a}^1,\dots,\hat{a}^{j-1},\hat{a}^{j+1},\dots,\hat{a}^n$ actions in $U_l$ being performed by the other agents
13:      let $a^* = \arg\max_{a' \in A_i}[C_t^j(i,s',\hat{a}^1,\dots,\hat{a}^{j-1},\hat{a}^{j+1},\dots,\hat{a}^n,a') + V_t^j(a',s')]$
14:      let $N = 0$
15:      for each $(s,\ a^1,\dots,a^{j-1},a^{j+1},\dots,a^n)$ in ChildSeq from the beginning do
16:        $N = N + 1$
17:        $C_{t+1}^j(i,s,a^1,\dots,a^{j-1},a^{j+1},\dots,a^n,a^j) \longleftarrow$
           $[1-\alpha_t^j(i)]C_t^j(i,s,a^1,\dots,a^{j-1},a^{j+1},\dots,a^n,a^j) +$
           $\alpha_t^j(i)\gamma^N[C_t^j(i,s',\hat{a}^1,\dots,\hat{a}^{j-1},\hat{a}^{j+1},\dots,\hat{a}^n,a^*) + V_t^j(a^*,s')]$
18:      end for
19:    else    /* $M_i$ is not a cooperative subtask */
20:      choose subtask $a^j$ according to the current exploration policy $\pi_i^j(s)$
21:      let ChildSeq = Cooperative-HRL($j$, $a^j$, $s$), where ChildSeq is the sequence of (state-visited, actions in $\bigcup_{k=1}^{L} U_k$ being performed by the other agents) while executing subtask $a^j$
22:      observe result state $s'$
23:      let $a^* = \arg\max_{a' \in A_i}[C_t^j(i,s',a') + V_t^j(a',s')]$
24:      let $N = 0$
25:      for each state $s$ in ChildSeq from the beginning do
26:        $N = N + 1$
27:        $C_{t+1}^j(i,s,a^j) \longleftarrow [1-\alpha_t^j(i)]C_t^j(i,s,a^j) + \alpha_t^j(i)\gamma^N[C_t^j(i,s',a^*) + V_t^j(a^*,s')]$
28:      end for
29:    end if
30:    append ChildSeq onto the front of Seq
31:    $s = s'$
32:  end while
33: end if
34: return Seq
35: end Cooperative-HRL
```