A Dynamic Hierarchical Task Transfer in Multiple Robot Explorations

Mehran Asadi
Information Technology
The Lincoln University
Lincoln University, PA 19352, U.S.A
masadi@lincoln.edu

Manfred Huber
Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019, U.S.A
huber@cse.uta.edu

***Abstract*—To operate effectively in complex environments, learning agents have to selectively ignore irrelevant details by forming useful abstractions. These abstractions can be constructed using subtasks that are defined prior to the learning process. In this paper we extend our previous discoveries to a new multi-robot environment and we combine two recent methods in hierarchical reinforcement learning in order to introduce a novel mechanism that discovers the sub-policies in Markov Decision Process in a multi-agent system.**

I. Introduction

The work presented here focuses on the construction and transfer of control knowledge in the form of behavioral skill hierarchies and associated representational hierarchies in the context of a reinforcement learning agent. In particular, it facilitates the acquisition of increasingly complex behavioral skills and the construction of appropriate, increasingly abstract and compact state representations which accelerate learning performance while ensuring bounded optimality. Moreover, it forms a state hierarchy that encodes the functional properties of the skill hierarchy, providing a compact basis for learning that ensures bounded optimality.

II. Hierarchical Reinforcement Learning

To permit the construction of a hierarchical learning system, we model our learning problem as a Semi-Markov Decision Problem (SMDP) and use the options framework [1], [2] to define subgoals. An option is a temporally extended action which, when selected by the agent, executes until a termination condition is satisfied. While an option is executing, actions are chosen according to the option's own policy. An option is like a traditional macro except that instead of generating a fixed sequence of actions, it follows a closed-loop policy so that it can react to the environment. By augmenting the agent's set of primitive actions with a set of options, the agent's performance can be enhanced. More specifically, an option is a triple $o_i = (I_i, \pi_i, \beta_i)$, where I_i is the option's input set, i.e., the set of states in which the option can be initiated; π_i is the option's policy defined over all states in which the option can execute; and β_i is the termination condition, i.e., the option terminates with probability $\beta_i(s)$ for each state s. Each option that we use in this paper bases its policy on its own internal value function, which can be modified over time in response to the environment. The value of a state s under an SMDP policy π^o is defined as [3], [1], [4], [5]:

$$V^{\pi}(s) = E\left[R(s, o_i) + \sum_{s'} F(s'|s, o_i)V^{\pi}(s')\right]$$

where

$$F(s'|s, o_i) = \sum_{k=1}^{\infty} P(s_t = s'|s_t = s, o_i)\gamma^k$$

, where $\gamma \in [0, 1]$ is a discount-rate parameter.

III. Previous Work

In our previous work [6], [7] we constructed an appropriate BPMDP for a specific action set

$O_t = \{o_i\}$, and an initial model was constructed by concatenating all concepts associated with the options in O_t. Additional conditions are then derived to achieve the stability of partition and, once reward information is available, the partitions were further refined according to a defined criteria. This construction facilitates efficient adaptation to changing action repertoires.

To further utilize the power of abstract actions, a hierarchy of BPSMDP (Bounded Parameter SDMP) models was constructed where the decision-level model utilized the set of options considered necessary while the evaluation-level used all actions not considered redundant. In the our system, a simple heuristic was used where the decision-level set consisted only of the learned subgoal options while the evaluation-level set included all actions.

Let $P = \{B_1, \ldots, B_n\}$ be a partition for state space S derived by the action-dependent partitioning method, using subgoals $\{s_1, \ldots, s_k\}$ and options to these subgoals $\{o_1, \ldots, o_k\}$. If the goal state G belongs to the set of subgoals $\{s_1, \ldots, s_k\}$, then G is achievable by options $\{o_1, \ldots, o_k\}$ and the task is learnable. However, if $G \notin \{s_1, \ldots, s_k\}$ then the task may not be solvable using only the options that terminate at subgoals. The proposed approach solves this problem by maintaining a separate value function for the original state space while learning a new task on the partition space derived from only the subgoal options. During learning, the agent has access to the original actions as well as all options, but makes decisions only based on the abstract partition space information. While the agent tries to solve the task on the abstract partition space, it computes the difference in Q-values between the best actions in the current state in the abstract state space and in the original state space. If the difference is larger than a constant value , then there is a significant difference between different states underlying the particular block that was not captured by the subgoal options.

IV. Autonomous Hierarchy Construction

In the multi-phase partitioning and hierarchical learning method discussed in the previous section, it has so far been assumed that either the correct set of actions for constructing an abstract state space is available or that, as a simple heuristic, all subgoal options are selected as the relevant action set. While the latter can lead to good results when used in conjunction with the learning method it might lead to an ever increasing action set if a large sequence of tasks is to be learned. In particular, this heuristic has the limitation that it can never remove an option from the action set used for multi-phase partitioning, even if it is not used for any of the tasks. To address this limitation, this section presents a method aimed at automatically constructing the abstract representation based on the information contained in the previously learned task policies.

In order to estimate the structure of the state space for learning future tasks, we construct the decision layer here based on an estimate of the expected time to learn a new task according to previously learned tasks. Let $\Pi = \{\pi_1, \ldots, \pi_n\}$ be the set of previously learned polices and $P_i = \{B_{i,1}, \ldots, B_{i,n}\}$ be the corresponding partitions. Also let the triple $T_i = (\pi_i, P_i, Q_i)$ be a task on partition $P_i = \{B_{i,1}, \ldots, B_{i,n}\}$ with policy π_i and the Q-function Q_i. The expected number of experiences required to learn a task, with high probability, on partition P with action set O using a DP-based version of Q-learning is [8]:

$$T_{conv}(P, O) = c|P|^2|O|$$

where c is a constant and it is assumed that the task is learnable on P with action set O.

The expected time required to learn task T_i on state representation P (including the refinement process) can be obtained by calculating the number of experiences that are needed for learning T_i on partition P plus the amount of time that is needed to refine a block of partition P, that is:

$$E[t_{T_i}|P] = t_{conv}(P, O) + \sum_{B_j \in P} P_{refine}(B_j|T_i) t_{conv}(\{B_{i,k}|B_{i,k} \cap B_j \neq \emptyset\}, O)$$

We compute the likelihood that a block B_j has to be refined during the exploration and learning of task T_i with the following equation:

$$P_{refine}(B_j|T_i) = \sum_{B_{i,k}: B_{i,k} \cap B_j \neq \emptyset} P_{refine}(B_j|B_{i,k}, T_i) P(B_{i,k}|T_i)$$

where

$$P_{refine}(B_j|B_{i,k},T_i) = \begin{cases} 1 & \text{if } A \\ 0 & \text{otherwise} \end{cases}$$

where $A = max_a(Q_i(B_{i,k},a) > max_{a\in O_{B_i,k}}(Q_i(B_{i,k},a))) + L$ and $L = 2(1 + (\frac{\gamma}{1-\gamma}))\max\{\epsilon,\delta\}$ and $P_{refine}(B_j|B_k^i,T_i)$ is the probability that block B_j has to be refined during the exploration and learning of T_i due to encountering block $B_{i,k}$ which is at least partially contained in B_j and for which an action a which is not contained in the currently considered action set $O_{B_i,k}$, with significantly higher value should then be included using the hierarchical learning scheme.
We compute the expected time required to learn a task randomly chosen from the distribution of previously learned tasks according to an importance distribution $U(T_i)$ which indicates the weight that should be put on each tasks by:

$$E[t_{learn}|P] = \sum_i \frac{U(T_i)}{\sum_i U(T_i)} E[t_{T_i}|P]$$

Algorithm 1 illustrates the process of autonomous hierarchy construction, in particular this is a greedy algorithm that finds action-dependent partitions that have the smallest expected learning time given previously learned tasks. The reason for the greedy approach is to reduce the complexity sufficiently to make it tractable. This approach is very similar to McCullum's U-tree algorithm [9], [10] except that splits are driven not by reward but by the expected learning time metric derived before. This procedure can be done either by splitting the blocks separately or by limiting the inclusion of actions across the state space. While the latter saves us more computational time, the former will give us more nuanced splits.

Algorithm 1 Autonomous Hierarchy Construction

Require: $O_0 = \emptyset$, $P_0 = \{s\}$
  $n = 0$
  repeat
    for all B_j in P_n and $o_i \in O - O_{n,B_j}$ **do**
      $P_{n+1,(i,j)} = P_n$ where B_j is refined with o_i
    end for
    $(k,l) = argmin_{(b,c)} E[t_{learn}|P_{n+1,(b,c)}]$
    $P_{n+1} = P_{n+1,(k,l)}$
    $B = B_l$
    for all $B_i \in P_n$ **do**
      for all $B_j \in P_{n+1}, B_j \subseteq B_i$ **do**
        if $B_i = B$ **then**
          $O_{n+1,B_j} = O_{n,B} \cup \{o_k\}$
        else
          $O_{n+1,B_j} = O_{n,B_i}$
        end if
      end for
    end for
    $n = n + 1$
  until $E[t_{learn}|P_n] \geq E[t_{learn}|P_{n-1}]$
  return P_{n-1}
  END

V. Empirical Results

The experiment shows the result of the presented approach in a game domain that is more complex and more similar to real environments. While all these experiments use the heuristic of using all subgoals action to construct the abstract decision layer, the experiment in the same game domain investigates the autonomous hierarchy construction approach in order to illustrate the construction of an approximate partition using the information of the previously learned polices.
The actions are GoUp, GoDown, TurnLeft, TurnRight, PickUp and DropOff. The cost for each single step action is -1 and each action for navigation succeeds with probability 1. The reward in the goal state where the agent can pickup and drop off the object is 100. The state is here characterized by the agent's pose as well as by a set of local object percept, resulting in an effective state space with $20,000$ states. The agent is first presented with a reward function to learn to move to a specific location. Once this task is learned, subgoals are extracted by generating random sample trajectories.
In order to show the construction of the decision layer, a sequence of five different tasks is learned in the game environment. The first task is to navigate the environment, i.e., the agent learns how to move from one location to another location. The second task is to navigate the environment and pick up an object. The goal of third task is to navigate a different region of the environment, and in the fourth task the agent learns how to navigate, pick up an object and drop it off in another location. The fifth task is the combination of the first four tasks

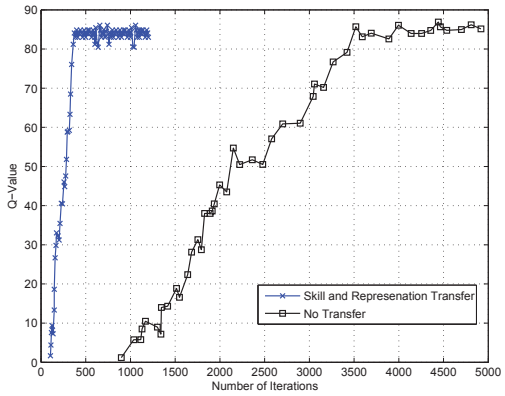

Fig. 1. Learning curves for the first navigation task. The agent learns to navigate the environment and the information acquired by learning this task will be used for constructing a partition for the next task ,i.e, the navigation and pickup tasks

by using the information acquired while learning the first four tasks, i.e., the agent learns to navigate the environment and to pick up an object and drop it off in another location.
Figures 1, 2, 3 and 4 show the learning curves for the first four tasks.

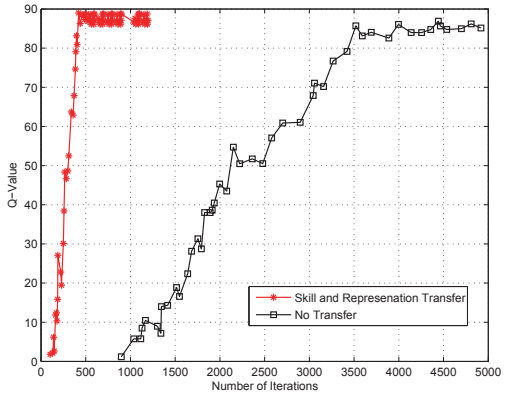

Fig. 2. Learning Curves for the second task, i.e., the navigation and pickup tasks. The information acquired by learning this task and the first task will be used for constructing a partition for the third task

At each step, a previously learned policy is added to the action set in order to construct a partition

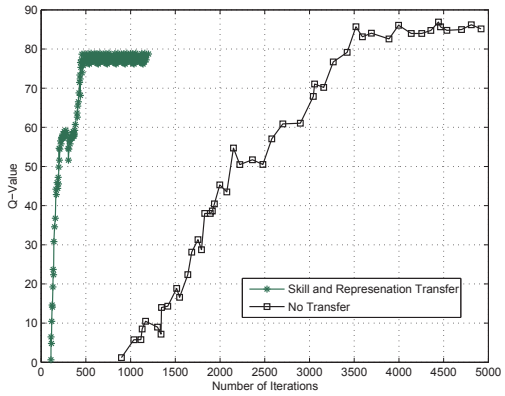

Fig. 3. Learning Curves for the third task, i.e., the second navigation task. The information acquired by learning this task and the previous two tasks will be used for constructing a partition for the fifth task

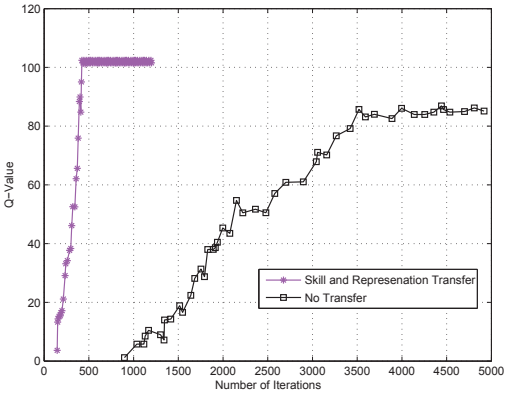

Fig. 4. Learning Curves for the fourth task, i.e., the navigation and dropoff tasks. A new partition will be constructed by using a history of previously learned tasks for future subsequent tasks

that is more relevant to the learning of a new task using Algorithm 1. The number of blocks for task 1 through 5 is illustrated in Figure 5. The number of blocks of this partition is illustrated in Figure 6. This experiment shows how a new partition can be constructed by using a history of previously learned task while it ensures that the new policy is within a fixed bound from the optimal policy. Figure 7 illustrates the learning curves on the compact state space, constructed by

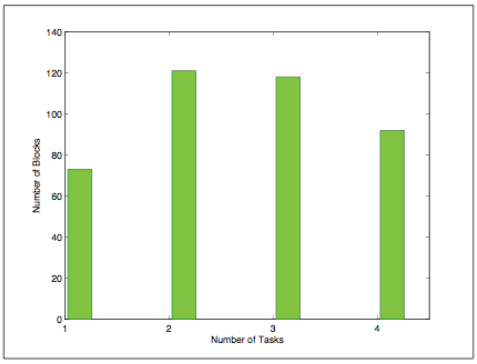

Fig. 5. Number of blocks constructed for learning task 1 through task 5

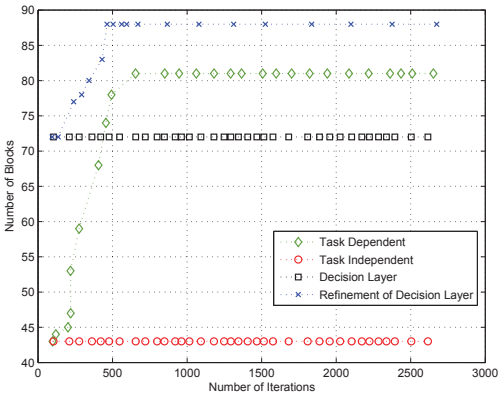

Fig. 6. Number of blocks for decision layer after refinement of task dependent partition. As a result of further refinement of the original blocks of partition the number of blocks increases, however this number becomes stables after finite and relatively small number of iterations.

using previously learned polices.

VI. Conclusion and Future Work

The results presented in this paper show a significant reduction in the number of states in the abstract state space, resulting in faster convergence of the value function. Furthermore, these experiments show a procedure to estimate the structure of the state space for learning future tasks and to construct the decision layer based on the expected time to learn a new task according to previously learned tasks.

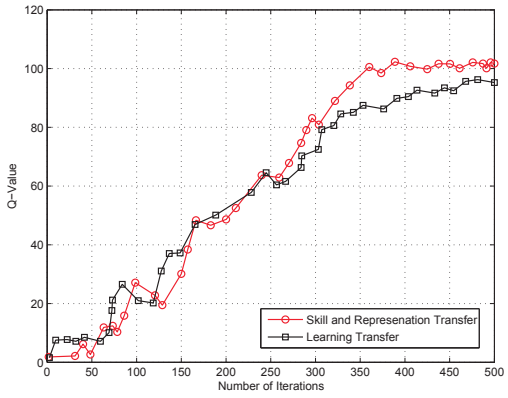

Fig. 7. Learning on a partition space obtained by Autonomous Hierarchy construction method by using the first four tasks. This experiment shows how a new partition can be constructed by using a history of previously learned tasks while it ensures that the new policy is within a fixed bound from the optimal policy

One of the future goals is to find even more efficient machine learning methods for control tasks. Algorithms can be developed for statistical generalization and reasoning about the algorithms that learn to incrementally scale up to analyze even more complex tasks. Discovering hierarchy in task structure and world structure is an important means in achieving this end. Algorithms need to be developed that learn to reason about their environment in a combinatorial way and learn to develop more cognitive internal representations that mimic relational structures. Integration of more powerful representations such as factorial HMMs and POMDPs are a potential follow-up to this work. Smarter hierarchical algorithms must be found to deal with larger tasks, and research must be directed at more intelligent representational design not only for incorporating hierarchy but also for sharing substructures.

References

[1] R. Sutton, D. Precup, and S. Singh, "Between MDPs and Semi-MDPs: Learning, Planning, and Representing Knowledge at Multiple Temporal Scales," *Artificial Intelligence*, vol. 112, pp. 181–211, 1999.

[2] R. Parr, "Hierarchical Control and Learning for Markov Decision Processes," Ph.D. dissertation, University of California, Berkeley, CA, 1998.

[3] C. Boutilier, T. Dean, and S. Hanks, "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," *Journal of Artificial Intelligence Research*, vol. 11, pp. 1–94, 1999.

[4] K. Kim and T. Dean, "Solving Factored MDPs using Non-Homogeneous Partitions," *Artificial Intelligence*, vol. 147, pp. 225–251, 2003.

[5] T. G. Dietterich, "An Overview of MAXQ Hierarchical Reinforcement Learning," *Lecture Notes in Computer Science*, vol. 1864, 2000.

[6] M. Asadi and M. Huber, "State Space Reduction For Hierarchical Reinforcement Learning," in *Proceedings of the 17th International FLAIRS Conference*. AAAI, 2004, pp. 509–514.

[7] ——, "Effective control knowledge transfer through learning skill and representation hierarchies," in *IJCAI, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, 2007, pp. 2054–2059.

[8] A. Barto and S. Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning," *Discrete Event Dynamic Systems*, vol. 13, pp. 341–379, 2003.

[9] A. McGovern and A. Barto, "Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density," in *Proceedings of the 18th International Conference on Machine Learning*, 2001, pp. 361–368.

[10] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[11] H. Janzadeh and M. Huber, "Learning policies in partially observable mdps with abstract actions using value iteration," in *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference.*, 2013.

[12] H. Rahmanian and M. Huber, "Data modeling using channel-remapped generalized features," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 864–869.

[13] M. Aurangzeb, F. L. Lewis, and M. Huber, "Efficient, swarm-based path finding in unknown graphs using reinforcement learning," *Control and Intelligent Systems*, vol. 42, no. 3, 2014.

[14] D. M. Clement and M. Huber, "Using multi-agent options to reduce learning time in reinforcement learning," in *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference*, 2015, pp. 26–31.

[15] C. Boutilier, R. Dearden, and M. Goldszmidt, "Exploiting Structure in Policy Construction," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995, pp. 1104–1111.

[16] A. McCallum, "Overcoming Incomplete Perception with Utile Distinction Memory," in *Proceedings of the Tenth International Machine Learning Conference*, 1993.

[17] T. Dean, R. Givan, and M. Greig, "Equivalence Notions and Model Minimization in Markov Decision Processes," in *Special issue on planning with uncertainty and incomplete information*, 2003, pp. 163–223.

[18] T. Dean, L. Kaelbling, J. Kirman, and A. Nicholson, "Planning Under Time Constraints in Stochastic Domains," *Artificial Intelligence*, vol. 76, no. 1-2, pp. 35–74, 1995.

[19] B. Digney, "Emergent hierarchical control structures: Learning reactive / hierarchical relationships in reinforcement environments," in *Proceedings of the Fourth Conference on the Simulation of Adaptive Behavior*, 1996.

[20] C. Drummond, "Using a Case Base of Surfaces to Speed-Up Reinforcement Learning," in *Proceedings of the Second International Conference on International Conference on Case-Based Reasoning*, 1997, pp. 435–444.

[21] T. Dean, R. Givan, and S. Leach, "Model Reduction Techniques for Computing Approximately Optimal Solutions for Markov Decision Processes," in *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*. San Francisco, CA: Morgan Kaufmann Publishers, 1997, pp. 124–131.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[23] R. H. Crites and A. G. Barto, "Elevator group control using multiple reinforcement learning agents," *Machine Learning*, vol. 33, pp. 235–262, 1998.

[24] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines. in," in *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1998.

[25] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Artificial Intelligence Research*, vol. 13, pp. 227–303, 2000.

[26] M. L. Puterman, *Markov Decision Problems*. New York: Wiley, 1994.

[27] G. J. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, pp. 257–277, 1992.

[28] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, pp. 674–690, 1997.

[29] S. Mahadevan, N. Marchalleck, T. Das, and A. Gosavi, "Self-improving factory simulation using continuous-time average-reward reinforcement learning," in *Proceedings of the 14th International Conference on Machine Learning*, Nashville, TN, 1997.

[30] S. Singh and D. Bertsekas, "Reinforcement learning for dynamic channel allocation in cellular telephone systems," in *Proceedings of the 1996 Conference on Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1997.

[31] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, 1989.

[32] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems*. Cambridge, MA: The MIT Press, 1996, pp. 1038–1044.