Transfer of Learning Across Compositions of Sequential Tasks

Satinder P. Singh
Computer and Information Science Dept.
University of Massachusetts, Amherst,MA 01003
satinder@cs.umass.edu

Abstract

Most "weak" learning algorithms, including reinforcement learning methods, have been applied on tasks with single goals. The effort to build more sophisticated learning systems that operate in complex environments will require the ability to handle multiple goals. Methods that allow transfer of learning will play a crucial role in learning systems that support multiple goals. In this paper I describe a class of multiple tasks that represents a subset of routine animal activity. I present a new learning algorithm and an architecture that allows transfer of learning by the "sharing" of solutions to the common parts of multiple tasks. A proof of the algorithm is also provided.

1 INTRODUCTION

One of the pervasive problems facing machine learning researchers is the inability of most "weak" learning algorithms to scale well to complex tasks. Much of the research effort on the scaling problem has focussed on two alternatives: discovering new learning algorithms that show better scaling properties, and extending existing learning algorithms. However, while it is true that most animals learn multiple tasks and have multiple goals, most artificial learning systems focus on solving *single* complex tasks. Building complex learning agents that parallel some of the capabilities of real-life learning systems will necessitate handling multiple goals/tasks[1]. While discovering faster learning methods will help, methods that allow transfer of learning across tasks will play a crucial role in learning systems that support multiple goals.

The learning paradigm considered in this paper is that of reinforcement learning (RL) where at each time step an agent takes an action based on its current goal and the state of a dynamic external environment or world. The agent's action changes the state of the world and elicits a payoff that usually represents the immediate "value" of taking the action in that state. Tasks where the agent has to take a sequence of actions to accomplish its goal are called sequential decision tasks (SDTs). Typically, SDT's are posed as optimal control problems (Barto *et al.* [3],1990) where the objective of the agent is to maximize some functional of the payoffs received over a time horizon. An optimal control policy is a mapping from states to actions that achieves the agent's objective.

Control architectures based on reinforcement learning methods are increasingly being used for learning situation-action rules, or *reactions*, that can then be used for real-time decision making (Sutton [9], 1990; Whitehead and Ballard [11],1990). To date, most applications have dealt with learning to solve single tasks, i.e., learning the situation-action rules for satisfying a single goal. One way to solve multiple SDTs is to learn the optimal policy for each SDT independently. If the tasks are related, learning them independently will waste computational resources, both memory and time. Also, since RL methods do not scale efficiently to large tasks, the "learn each task independently" method will be feasible for simple tasks only.

In this paper I propose an "interesting" set of structured tasks, and then present a learning architecture that can learn multiple tasks with significant transfer of learning across tasks.

2 COMPOSITIONALLY STRUCTURED TASKS

A large part of everyday human activity involves complex sequential tasks that have compositional structure, i.e., complex tasks which are built up in a systematic way from simpler tasks. Consider, as an example, how many of our daily activities involve the subtasks: lift an object, open a door, sit down, walk, etc. To for-

[1] Throughout this paper I will use the words: tasks and goals, interchangeably.

mulate the problem abstractly, consider an agent that has to learn to solve many different simple and complex tasks. In general, there may be n *elemental* tasks labeled $T_1, T_2, \ldots, T_n$, i.e., tasks that cannot be decomposed into simpler tasks. *Composite* tasks can be produced by the temporal concatenation of a number of elemental tasks, for example, $[T_{i_1} T_{i_2} \cdots T_{i_{k-1}} T_{i_k}]$ is a composite task made up of k elemental subtasks. The solution for an elemental subtask is assumed to be context-free, i.e., independent of the other subtasks in the composite task. In this paper, I propose a learning method that discovers the decomposition of a composite task, and constructs solutions for composite tasks from the solutions for their elemental subtasks.

The sequential decision-making framework can incorporate multiple tasks simply by augmenting the state description to include task descriptors. The agent can then still have only one goal, that of maximizing its payoff functional, and yet perform different tasks. The particular task in which the agent is engaged will be contingent on the augmented state of the world, and clearly the agent would "see" different parts of the augmented state space for different tasks. An alternative viewpoint, and one that will be adopted in this paper, is to separate the task and state descriptions and imagine an external agency providing the goal or task command to the learning agent. Setting a different task will then amount to changing the payoff structure so that in maximizing the payoff functional the agent will accomplish the desired task.

Formulations of the payoff structure for single SDTs usually associate payoffs only with the state-action pairs. For multiple tasks, multiple payoff values can be defined for each state-action pair, one for each task. However, for many sets of multiple tasks it is possible to define a single payoff value for each state-action pair, if the different tasks can associate payoffs with the states themselves. Intuitively, one can think of associating "costs" with the state-actions and "rewards" with the states. Consider, as an example, that the "cost" of opening a door is usually independent of the final destination. In such cases, changing the payoff structure for a new task will only involve changing the state payoffs and not the state-action pair payoffs. I will adopt the latter, more economical, formulation of multiple SDTs.

3 COMPOSITIONAL Q-LEARNING

If an exact model of the dynamics of the world and the payoff structure is available, dynamic programming based computational procedures can be used to solve for the optimal policy (see Barto and Singh [1], 1990). In the absence of a world-model and the payoff structure, reinforcement learning methods that approximate dynamic programming techniques, such as temporal difference (TD) procedures (Sutton [8], 1988; Barto *et al.* [2], 1983) and Q-learning methods (Watkins [10], 1989; Barto and Singh [4], 1990), can be used to directly estimate an optimal policy without building an explicit model of the world-dynamics.

For a deterministic SDT with state set S and action set A, Q-learning maintains a Q-value for each state-action pair, Q_{xa}, which is the discounted sum of future payoffs on taking action a in state x and following the optimal policy thereafter. The Q-values satisfy the following: $Q_{xa} = r(x,a) + \gamma \max_{a' \in A} Q_{ya'}$, where $r(x,a)$ is the payoff on taking action a in state x, and y is the resulting state. The optimal action for state x, $a^* = \arg\max_a Q_{xa}$, maximizes the discounted sum of all future payoff. The Q-values should satisfy the following equation:

$$Q_{x_0 a_0^*} = \sum_{i=0}^{\infty} \gamma^i r(x_i, a_i^*), \qquad (1)$$

where x_i is the state at the i^{th} time step, and a_i^* denotes the optimal action from state x_i. The update rule to learn the right Q-values is given by: $Q_{xa} = Q_{xa} + \alpha((r(x,a) + \gamma \max_{a' \in A} Q_{ya'}) - Q_{xa})$, where α is a learning rate parameter.

Compositional Q-learning (CQ-learning) is a computationally inexpensive way to construct the Q-value functions for composite tasks from the Q-value functions for their elemental tasks. Formally, let $Q_{T_i}(x,a)$ be the Q-value for state-action pair (x,a) when the agent is doing only the elemental task T_i. For composite task $C = [T_{j_1} T_{j_2} \cdots T_i \cdots T_{j_k}]$, let $Q^C_{T_i}(x,a)$ represent the Q-value for state-action pair (x,a) when the agent is performing subtask T_i in composite task C. Then for deterministic SDTs with Markovian state descriptions (see Barto *et al.* [3],1990), with the following assumptions:

(A1) Each task has a single goal state.
(A2) The optimal path length for all elemental tasks is bounded.
(A3) $\gamma = 1.0$.
(A4) The payoff associated with the state-action pairs is independent of the goal being accomplished.
(A5) For an elemental task, any task-dependent payoff is associated only with the goal state. For composite tasks, task-dependent payoff can be associated with the goal states of some or all elemental subtasks.

it is proved in the Appendix A, that for all states x and actions a:

$$Q^C_{T_i}(x,a) = Q_{T_i}(x,a) + K^C_{T_i}, \qquad (2)$$

where $K^C_{T_i}$ is a constant whose value depends on the subtask T_i and the composite task C, and is *independent* of the state and action.

Note that CQ-learning does not figure out the decomposition of a composite task.

4 SCHEDULING ARCHITECTURE

The scheduling architecture is adapted from the modular gating architecture developed by Jacobs [5] (1990), which has only been used to learn multiple non-sequential tasks within the supervised learning paradigm. The scheduling architecture combines CQ-learning with Nowlan's [6] (1990) competitive experts algorithm in order to learn both the decomposition for a composite task and its Q-value function. A very brief description of the scheduling architecture is presented here (See Nowlan [6], (1990); Singh [7], (1991) for details.).

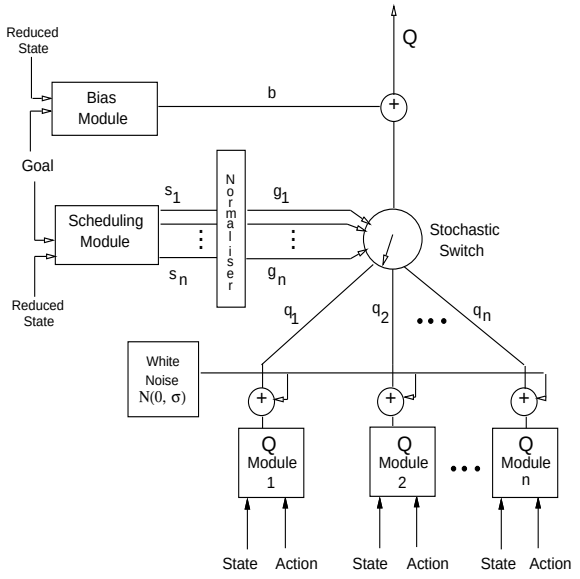

Figure 1: Scheduling Architecture (adapted from Nowlan [6] (1990)). See text for details.

The scheduling architecture (see Figure 1) consists of a number of Q-modules that compete to learn the Q-value functions for the elemental tasks. The scheduling module has two functions: it learns to assign the different elemental tasks to the different Q-modules, and it forms temporal compositions of the Q-modules to represent the Q-value functions of composite tasks. For a composite task, the Q-module that best represents the Q-value function of the particular elemental task being performed at a given time wins the competition, and therefore learns the Q-value function of that elemental task even better. Over time, the scheduling module adjusts its outputs to turn on the winning Q-modules only. This process leads to the discovery of the temporal decomposition of a composite task.

The action at time t was selected from the winning Q-module via the Boltzman distribution, i.e. for action $a \in A$ and $x \in S$, $P(a/x) = e^{\beta Q_{xa}} / \sum_{a' \in A} e^{\beta Q_{xa'}}$, where the parameter β tended to infinity over time via an annealing process.

Table 1: Tasks T_1, T_2, and T_3 are elemental tasks. Tasks C_1, C_2, and C_3 are composite tasks. The last column describes the compositional structure of the tasks.

Label	*Repr.*	*Description*	*Decomp.*
T_1	000001	visit A	T_1
T_2	000010	visit B	T_2
T_3	000100	visit C	T_3
C_1	001000	visit A and then C	T_1T_3
C_2	010000	visit B and then C	T_2T_3
C_3	100000	visit A, then B, then C	$T_1T_2T_3$

5 TASK DESCRIPTION

In this paper, I ignore the representation and the function approximation issues, and focus on transfer of learning by the "sharing" of solutions to common elements of multiple composite tasks. I use a deterministic task (see Figure 2) where a robot has to navigate in a 8×8 grid room with 3 goal locations designated A, B and C. In each state the robot has 4 actions: UP, DOWN, LEFT and RIGHT. Any action that would take the robot into an obstacle or boundary wall does not change the robot's location. The three elemental tasks: T_1, T_2 and T_3, and the three composite tasks: C_1, C_2, and C_3, are described in Table 1.

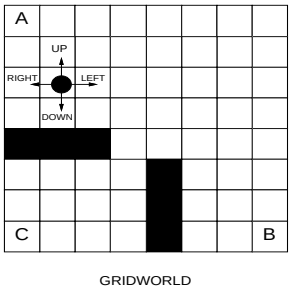

Figure 2: An 8×8 grid room with 3 goal locations designated as A, B and C. The filled squares are obstacles that the robot must navigate around.

Lookup tables were used to implement all the modules of the scheduling architecture and unit-basis representations (see Table 1) were chosen for task commands. To simplify matters further, it was assumed that the state′ input to the bias and the scheduling modules (see Figure 1) has 3 binary bits, one for each elemental task. For a composite task, the bit corresponding to each elemental subtask was set when that subtask was accomplished and remained set for the duration of the composite task. The state′ bits were reset at the beginning of each task.

6 EXPERIMENTAL RESULTS

The[2] scheduling architecture containing 3 Q-modules was compared to a one-for-one architecture containing 6 modules. Both architectures were separately trained to do six tasks: T_1, T_2, T_3, C_1, C_2, and C_3. The one-for-one architecture learned each task in a separate pre-assigned module. For each trial, the task and the starting state of the robot were chosen randomly. The trial ended when the robot reached the goal state. Figure 3A shows the number of actions, averaged over 50 trials, taken by the robot to get to the goal. The one-for-one architecture performs better initially because it can learn the three elemental tasks quickly, but learning the composite tasks takes much longer due to the long action sequences required to accomplish the composite tasks. The scheduling architecture performs worse initially, until the outputs of the scheduling module become approximately correct, at which point all six tasks are learned rapidly.

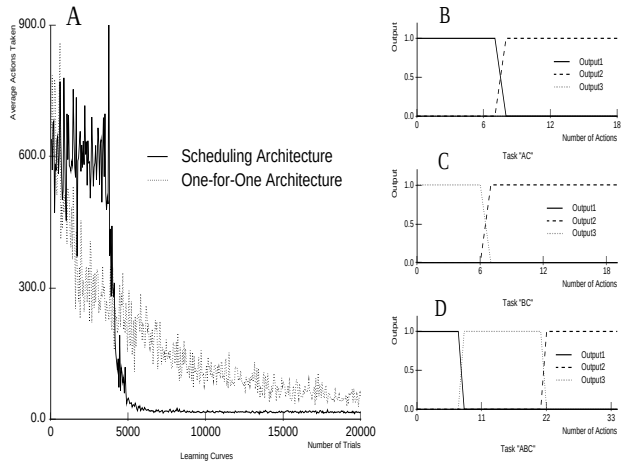

Figure 3: **A:** Typical results for learning both elemental and composite tasks. See text for details.

Figures 3B,3C, and 3D show the 3 normalized outputs of the scheduling module for one trial each of tasks C_1, C_2, and C_3 respectively. The trials shown were chosen after the robot had learned to do the tasks, i.e., after 10,000 learning trials. The elemental tasks T_1, T_2, and T_3 are learned by the Q-modules 1,3 and 2 respectively. Figures 3B,3C, and 3D show that for each composite task the scheduling module switches on the appropriate Q-modules in the right temporal sequence.

[2]Appendix B lists the parameter values used for the experiment reported in this section. Please note that due to lack of space, details of the scheduling architecture have not been provided. See Singh [7], (1991) for details.

7 DISCUSSION

This paper presented an economical formulation for multiple, compositionally-structured tasks within the sequential decision-making framework. A new algorithm, CQ-learning, was presented and it was shown that under assumptions A1–A5 (see Section 3), there is an efficient way of constructing solutions for composite tasks from the solutions of their elemental tasks. The results in Section 6 show that the scheduling architecture represents a general mechanism whereby a "vocabulary" of elemental tasks can be learned in separate modules, and arbitrary[3] temporal syntactic compositions of elemental tasks can be learned with the help of the bias and scheduling modules.

Animals rarely learn a new task in isolation, they bring to bear on a new task knowledge gained in numerous prior experiences with many different tasks. The traditional response within the machine learning community has been to reject the tabula rasa approach and to capture that experience by building prior knowledge into the data structures used by the learning system. However, it is my conjecture that much will be gained by building learning systems that learn to perform multiple tasks in a complex environment.

Future work will investigate the use of CQ-learning and the scheduling architecture on more realistic tasks and more complex state and task command representations. Also, the context-free nature of the solutions to elemental tasks is a strong constraint, and means of extending the above method to more general multiple task formulations are being considered.

8 APPENDIX

A Proof of Equation 2

Consider an elemental deterministic sequential decision task (SDT) T_i and its goal state $x_g^{T_i}$. For any arbitrary start state x_0, assume w.l.o.g. that the optimal path from x_0 to $x_g^{T_i}$ goes through $n+2$ states; $x_0, x_1, \ldots, x_{n-1}, x_n, x_g^{T_i}$. The action sequence ends once the agent is in the goal state. Define $U_{T_i}(x) = \max_a Q_{T_i}(x,a) \doteq Q_{T_i}(x,a^*)$. Let $r(x_i, a_i^*) = c_i$ for $0 \le i \le (n-1)$ and $r(x_n, a_n^*) = c_n + E^{T_i}$, where E^{T_i} is the goal-dependent payoff associated with goal state $x_g^{T_i}$. Then using Equation 1, $\forall j; 0 \le j \le n$:

$$U_{T_i}(x_j) \quad = \quad c_j + c_{j+1} + \ldots + c_{n-1} + (c_n + E^{T_i}) \qquad (3)$$

For any arbitrary action, a, and $\forall j; 0 \le j \le (n-1)$:

$$Q_{T_i}(x_j, a) \quad = \quad r(x_j, a) + U_{T_i}(x_{j+1}). \qquad (4)$$

[3]This assumes that the state′ representation is rich enough to distinguish repeated performances of the same elemental task.

Let the symbol C stand for a deterministic composite task $[T_{j_1} T_{j_2} \cdots T_i \cdots T_{j_k}]$ which contains subtask T_i. Then $U^C_{T_i}(x) = \max_a Q^C_{T_i}(x,a) \doteq Q^C_{T_i}(x,a^*)$. The optimal path for the composite task C has to pass through the goal states of all the elementary subtasks that form the composite task. Furthermore, each elemental subtask terminates in its own goal state. Consider the start state x_0 chosen in the previous paragraph. Then with the assumptions cited in Section 3 the optimal path from x_0 to $x_g^{T_i}$ remains the same as when the agent was performing only the elemental task T_i. Define $V^C_{T_i} = U^C_{T_{i+1}}(x_g^{T_i})$, where T_{i+1} is the elemental subtask that starts when subtask T_i finishes. Let the goal-dependent payoff for reaching state $x_g^{T_i}$ in task C be $\acute{E}^C_{T_i}$. Then $\forall j; 0 \leq j \leq n$:

$$U^C_{T_i}(x_j) \quad = \quad c_j + \ldots + c_{n-1} + (c_n + \acute{E}^C_{T_i}) + V^C_{T_i}, \quad (5)$$

and for any arbitrary action, a, and $\forall j; 0 \leq j \leq (n-1)$:

$$Q^C_{T_i}(x_j, a) \quad = \quad r(x_j, a) + U^C_{T_i}(x_{j+1}), \quad (6)$$

Then for any arbitrary elemental task T_i, and any composite task C such that T_i is contained in C, and for all states x, and for all actions a: Equations 3, 4, 5,and 6 show that:

$$U^C_{T_i}(x) \quad = \quad U_{T_i}(x) + (V^C_{T_i} + \acute{E}^C_{T_i} - E^{T_i}).$$

Define $K^C_{T_i} = V^C_{T_i} + \acute{E}^C_{T_i} - E^{T_i}$, then:

$$Q^C_{T_i}(x,a) \quad = \quad Q_{T_i}(x,a) + K^C_{T_i},$$

Q.E.D.

The above proof can be extended to stochastic sequential decision problems that satisfy the assumptions stated in Section 3 by noting that the transition probabilities for an elemental task are independent of the position of that elemental task within a composite task.

B Parameter Values

The initial values for the lookup tables implementing the Q-modules were random values in the range 0.0–0.5, and the initial values for the scheduling module lookup table were random values in the range 0.0–0.4. The policy selection parameter β, started at 0.1 and was incremented by 1.0 after every 100 trials.

Acknowledgements

I would like to thank Andrew Barto and the UMass Adaptive Networks Group for their help on this project. This work was supported by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526 and by the National Science Foundation under Grant ECS-8912623.

References

[1] A. G. Barto and S. P. Singh. Reinforcement learning and dynamic programming. In *Proc. of the Sixth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, Aug 1990.

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE SMC*, 13:835–846, 1983.

[3] A. G. Barto, R. S. Sutton, and C. Watkins. Sequential decision problems and neural networks. In D. S. Touretzky, editor, *NIPS 2*, pages 686–693, San Mateo, CA, 1990. Morgan Kaufmann.

[4] A.G. Barto and S.P. Singh. On the computational economics of reinforcement learning. In *Proc. of the 1990 Connectionist Models Summer School*, San Mateo, CA, Nov. 1990. Morgan Kaufmann.

[5] R. A. Jacobs. *Task decomposition through competition in a modular connectionist architecture.* PhD thesis, COINS dept Univ. of Massachusetts, Amherst, Mass. U.S.A., 1990.

[6] S. J. Nowlan. Competing experts: An experimental investigation of associative mixture models. Technical Report CRG-TR-90-5, Department of Computer Sc., Univ. of Toronto, Toronto, Canada, 1990.

[7] S. P. Singh. Transfer of learning by composing elemental sequential tasks, 1991. submitted to Machine Learning – Special Issue on Reinforcement Learning.

[8] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[9] R. S. Sutton. Integrating architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proc. of the Seventh International Conf. on Machine Learning*, pages 216–224, San Mateo, CA, 1990. Morgan Kaufmann.

[10] C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge Univ., Cambridge, England, 1989.

[11] S. D. Whitehead and D. H. Ballard. Active perception and reinforcement learning. In *Proc. of the Seventh International Conf. on Machine Learning*, Austin, TX, June 1990.