In Proc. 20th National Conference on Artificial Intelligence (AAAI-05), July 9–13, 2005, Pittsburgh, USA.

Improving Action Selection in MDP's via Knowledge Transfer

Alexander A. Sherstov and **Peter Stone**
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
{sherstov, pstone}@cs.utexas.edu

Abstract

Temporal-difference reinforcement learning (RL) has been successfully applied in several domains with large *state* sets. Large *action* sets, however, have received considerably less attention. This paper demonstrates the use of knowledge transfer between related tasks to accelerate learning with large action sets. We introduce *action transfer*, a technique that extracts the actions from the (near-)optimal solution to the first task and uses them in place of the full action set when learning any subsequent tasks. When optimal actions make up a small fraction of the domain's action set, action transfer can substantially reduce the number of actions and thus the complexity of the problem. However, action transfer between *dissimilar* tasks can be detrimental. To address this difficulty, we contribute *randomized task perturbation* (RTP), an enhancement to action transfer that makes it robust to unrepresentative source tasks. We motivate RTP action transfer with a detailed theoretical analysis featuring a formalism of related tasks and a bound on the suboptimality of action transfer. The empirical results in this paper show the potential of RTP action transfer to substantially expand the applicability of RL to problems with large action sets.

Introduction

Temporal-difference reinforcement learning (RL) (Sutton & Barto 1998) has proven to be an effective approach to sequential decision making. However, large state and action sets remain a stumbling block for RL. While large *state* sets have seen much work in recent research (Tesauro 1994; Crites & Barto 1996; Stone & Sutton 2001), large *action* sets have been explored to but a limited extent (Santamaria, Sutton, & Ram 1997; Gaskett, Wettergreen, & Zelinsky 1999).

Our work aims to leverage similarities between tasks to accelerate learning with large action sets. We consider cases in which a learner is presented with two or more *related* tasks with identical action sets, all of which must be learned; since real-world problems are rarely handled in isolation, this setting is quite common. This paper explores the idea of extracting the subset of actions that are used by the (near-)optimal solution to the first task and using them instead of the full action set to learn more efficiently in any subsequent tasks, a method we call *action transfer*. In many domains with large action sets, significant portions of the action set are irrelevant from the standpoint of optimal behavior. Consider, for example, a pastry chef experimenting with a new recipe. Several parameters, such as oven temperature and time to rise, need to be determined. But based on past experience, only a small range of values is likely to be worth testing. Similarly, when driving a car, the same safe-driving practices (gradual acceleration, minor adjustments to the wheel) apply regardless of the terrain or destination. Finally, a bidding agent in an auction can raise a winning bid by any amount. But past experience may suggest that only a small number of raises are worth considering. In all these settings, action transfer reduces the action set and thereby accelerates learning.

Action transfer relies on the similarity of the tasks involved; if the first task is not representative of the others, action transfer can handicap the learner. If *many* tasks are to be learned, a straightforward remedy would be to transfer actions from *multiple* tasks, learning each from scratch with the full action set. However, in some cases the learner may not have access to a representative sample of tasks in the domain. Furthermore, the cost of learning multiple tasks with the full action set could be prohibitive.

We therefore focus on the harder problem of identifying the domain's useful actions by learning as few as *one* task with the full action set, and tackling all subsequent tasks with the resulting reduced action set. We propose a novel algorithm, *action transfer with randomized task perturbation* (RTP), that performs well even when the first task is misleading. In addition to action transfer and RTP, this paper contributes: (i) a formalism of related tasks that augments the MDP definition and decomposes it into *task-specific* and *domain-wide* components; and (ii) a bound on the suboptimality of *regular* action transfer between related tasks, which motivates *RTP* action transfer theoretically. We present empirical results in several learning settings, showing the superiority of RTP action transfer to regular action transfer and to learning with the full action set.

Preliminaries

A *Markov decision process* (MDP), illustrated in Figure 1, is a quadruple $\langle \mathcal{S}, \mathcal{A}, \mathtt{t}, r \rangle$, where $\mathcal{S}$ is a set of *states*; $\mathcal{A}$ is a set of *actions*; $\mathtt{t} : \mathcal{S} \times \mathcal{A} \rightarrow \Pr(\mathcal{S})$ is a *transition function* indicating a probability distribution over the next states upon taking a given action in a given state; and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a *reward function* indicating the immediate payoff upon taking a given action in a given state. Given a sequence of rewards $r_0, r_1, \ldots, r_n$, the associated *return* is $\sum_{i=0}^{n} \gamma^i r_i$, where $0 \leq \gamma \leq 1$ is the *discount factor*. Given a *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ for acting, its associated *value function* $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ yields, for every state $s \in \mathcal{S}$, the expected return from starting in state s and following π. The objective is to find an *optimal policy* $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ whose value function dominates that of any other policy at every state.

The learner experiences the world as a sequence of states, actions, and rewards, with no prior knowledge of the functions $\mathbf{t}$ and r. A practical vehicle for learning in this setting is the *Q-value function* $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, defined as $Q^\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathbf{t}(s'|s,a) V^\pi(s')$. The widely used Q-learning algorithm (Watkins 1989) incrementally approximates the Q-value function of the optimal policy.

As a running example and experimental testbed, we introduce a novel grid world domain (Figure 2) featuring discrete states but continuous actions. Some cells are empty; others are occupied by a wall or a bed of quicksand. One cell is designated as a *goal*. The actions are of the form (d, p), where $d \in \{$NORTH, SOUTH, EAST, WEST$\}$ is an intended direction of travel and $p \in [0.5, 0.9]$ is a continuous parameter. The intuitive meaning of p is as follows. Small values of p are *safe* in that they minimize the probability of a move in an undesired direction, but result in *slow* progress (i.e., no change of cell is a likely outcome). By contrast, large values of p increase the likelihood of movement, albeit sometimes in the wrong direction. Formally, the move succeeds in the requested direction d with probability p; lateral movement (in one of the two randomly chosen directions) takes place with probability $(2p-1)/8$; and no change of cell results with probability $(9-10p)/8$. Note that $p = 0.5$ and $p = 0.9$ are the extreme cases: the former prevents lateral movement; the latter forces a change of cell. Moves into walls or off the grid-world edge cause no change of cell.

The reward dynamics are as follows. The discount rate is $\gamma = 0.95$. The goal and quicksand cells are absorbing states with reward 0.5 and -0.5, respectively. All other actions generate a reward of $-p^2$, making fast actions more expensive than the slow ones. The optimal policy is always to move toward the goal, taking slow inexpensive actions $(0.5 \leq p \leq 0.60)$ far from the goal or near quicksand, and faster expensive actions $(0.6 < p \leq 0.65)$ when close to the goal. The fastest 62% of the actions $(0.65 < p \leq 0.9)$ do not prove useful in this model. Thus, ignoring them cannot hurt the quality of the best attainable policy. In fact, eliminating them decreases the complexity of the problem and can speed up learning considerably, a key premise in our work.

The research pertains to large action sets but does not require that they be continuous. In all experiments, we discretize the p range at 0.01 increments, resulting in a full action set of size 164. Since nearby actions have similar effects, generalization in the action space remains useful. The above intuitive grid world domain serves to simplify the exposition and to enable a precise, focused empirical study of our methods. However, our work applies broadly to any domain in which the actions are not equally relevant.

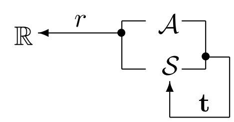

Figure 1: MDP formalism.

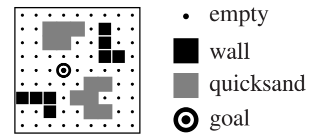

Figure 2: Grid world domain.

A Formalism for Related Tasks

The traditional MDP definition as a quadruple $\langle \mathcal{S}, \mathcal{A}, \mathbf{t}, r \rangle$ is adequate for solving problems *in isolation*. However, it is not expressive enough to capture similarities *across problems* and is thus poorly suited for analyzing knowledge transfer. As an example, consider two grid world maps. The abstract reward and transition dynamics are the same in both cases. However, the MDP definition postulates $\mathbf{t}$ and r as functions over $\mathcal{S} \times \mathcal{A}$. Since different maps give rise to different state sets, their functions $\mathbf{t}$ and r are formally distinct and largely incomparable, failing to capture the similarity of the reward and transition dynamics in both cases. Our new MDP formalism overcomes this difficulty by using *outcomes* and *classes* to remove the undesirable dependence of the model description ($\mathbf{t}$ and r) on the state set.

Outcomes Rather than specifying the effects of an action as a probability distribution $\Pr(\mathcal{S})$ over next *states*, we specify it as a probability distribution $\Pr(\mathcal{O})$ over *outcomes* $\mathcal{O}$ (Boutilier, Reiter, & Price 2001). $\mathcal{O}$ is the set of "nature's choices," or deterministic actions under *nature's* control. In our domain, these are: NORTH, SOUTH, EAST, WEST, STAY. Corresponding to every action $a \in \mathcal{A}$ available to the *learner* is a probability distribution (possibly different in different states) over $\mathcal{O}$. When a is taken, nature "chooses" an outcome for execution according to that probability distribution. In the new definition $\mathbf{t} : \mathcal{S} \times \mathcal{A} \rightarrow \Pr(\mathcal{O})$, the range $\Pr(\mathcal{O})$ is common to all tasks, unlike the original range $\Pr(\mathcal{S})$. The semantics of the outcome set is made rigorous in the definitions below.

Note that the qualitative effect of a given outcome differs from state to state. From many states, the outcome EAST corresponds to a transition to a cell just right of the current location. However, when standing to the left of a wall, the outcome EAST leads to a "transition" back to the current state. How an outcome in a state is mapped to the actual next state is map-specific and will be a part of a task description, rather than the domain definition.

Classes *Classes* $\mathcal{C}$, common to all tasks, generalize the remaining occurrences of $\mathcal{S}$ in $\mathbf{t}$ and r. Each state in a task is labeled with a class from among $\mathcal{C}$. An action's reward and transition dynamics are identical in all states of the same class. Formally, for all $a \in \mathcal{A}$ and $s_1, s_2 \in \mathcal{S}$,
$\kappa(s_1) = \kappa(s_2) \implies r(s_1, a) = r(s_2, a), \mathbf{t}(s_1, a) = \mathbf{t}(s_2, a)$, where $\kappa(\cdot)$ denotes the class of a state. Classes allow the definition of $\mathbf{t}$ and r as functions over $\mathcal{C} \times \mathcal{A}$, a set common to all tasks, rather than the task-specific set $\mathcal{S} \times \mathcal{A}$. Combining classes with outcomes enables a task-independent description of the transition and reward dynamics: $\mathbf{t} : \mathcal{C} \times \mathcal{A} \rightarrow \Pr(\mathcal{O})$ and $r : \mathcal{C} \times \mathcal{A} \rightarrow \mathbb{R}$.

To illustrate the finalized descriptions of $\mathbf{t}$ and r, consider the grid world domain. It features three classes, corresponding to the empty, goal, and quicksand cells. The reward and transition dynamics are the same in each class. Namely, the reward for action (d, p) is $-p^2$ in cells of the "empty" class, 0.5 in cells of the "goal" class, and -0.5 in cells of the "quicksand" class. Likewise, an action (NORTH, p) has the same distribution over the outcome set $\{\text{NORTH, SOUTH, EAST, WEST, STAY}\}$ within each class: it is $[0\ \ 0\ \ 0\ \ 0\ \ 1]^{\mathrm{T}}$ for all s in the "goal" and "quicksand" classes, and $[p\ \ 0\ \ (p-0.5)/8\ \ (p-0.5)/8\ \ (9-10p)/8]^{\mathrm{T}}$ for states in class "empty"; similarly for (SOUTH, p), etc.

Complete Formalism The above discussion casts the transition and reward dynamics of a *domain* abstractly in terms of outcomes and classes. A *task* within a domain is fully specified by its state set $\mathcal{S}$, a mapping $\kappa : \mathcal{S} \to \mathcal{C}$ from its states to the classes, and a specification $\eta : \mathcal{S} \times \mathcal{O} \to \mathcal{S}$ of the next state given the current state and an outcome. Thus, the defining feature of a task is its state set $\mathcal{S}$, which the functions κ and η interface to the abstract domain model.

Figure 3 illustrates the complete formalism, emphasizing the separation of what is common to all tasks in the domain from the specifics of individual tasks. Note the contrast with the original MDP formalism in Figure 1. Formally, domains and tasks are defined as follows:

Definition 1 *A* domain *is a quintuple $\langle \mathcal{A}, \mathcal{C}, \mathcal{O}, \mathbf{t}, r\rangle$, where $\mathcal{A}$ is a set of actions; $\mathcal{C}$ is a set of state classes; $\mathcal{O}$ is a set of action outcomes; $\mathbf{t} : \mathcal{C} \times \mathcal{A} \to \Pr(\mathcal{O})$ is a transition function; and $r : \mathcal{C} \times \mathcal{A} \to \mathbb{R}$ is a reward function.*

Definition 2 *A* task *within the domain $\langle \mathcal{A}, \mathcal{C}, \mathcal{O}, \mathbf{t}, r\rangle$ is a triple $\langle \mathcal{S}, \kappa, \eta\rangle$, where $\mathcal{S}$ is a set of states; $\kappa : \mathcal{S} \to \mathcal{C}$ is a state classification function; and $\eta : \mathcal{S} \times \mathcal{O} \to \mathcal{S}$ is a next-state function.*

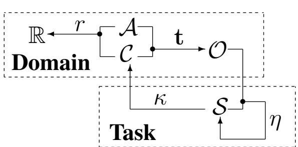

Figure 3: The formalism of related tasks in a domain.

Action Transfer: A Suboptimality Bound

Let $\tilde{\mathcal{A}}^* = \{a \in \mathcal{A} : \pi^*(s) = a \text{ for some } s \in \mathcal{S}\}$ be the *optimal action set* of an *auxiliary* task, and let $\mathcal{A}^*$ be the true optimal action set of the *primary* task. In action transfer, the primary task is learned using the transferred action set $\tilde{\mathcal{A}}^*$, in the hope that $\tilde{\mathcal{A}}^*$ is "similar" to $\mathcal{A}^*$. If $\mathcal{A}^* \not\subseteq \tilde{\mathcal{A}}^*$, the best policy $\tilde{\pi}^*$ achievable with the transferred action set in the primary task may be suboptimal. This section bounds the decrease in the highest attainable value of a state of the primary task due to the replacement of the full action set $\mathcal{A}$ with $\tilde{\mathcal{A}}^*$. The bound will suggest a principled way to cope with unrepresentative auxiliary experience.

In the related-task formalism above, a given state s can be succeeded by at most $|\mathcal{O}|$ states $s_1, s_2, \ldots, s_{|\mathcal{O}|}$ (not necessarily distinct), where s_i denotes the state that results if the ith outcome occurs. Suppose an oracle were to reveal the optimal values of these successor states; given a task, these values are well-defined. We refer to the resulting vector $\mathbf{v} = [V^*(s_1)\quad V^*(s_2)\quad \ldots\quad V^*(s_{|\mathcal{O}|})]^{\mathrm{T}}$ as the *outcome value vector* (OVV) of state s. OVV's are intimately linked to optimal actions: $\mathbf{v}$ immediately identifies the optimal action at s, $\pi^*(s) = \arg\max_{a \in \mathcal{A}}\{r(c, a) + \gamma \mathbf{t}(c, a) \cdot \mathbf{v}\}$, where $c = \kappa(s)$ is the class of s. Consider now the set of *all* OVV's of a task, grouped by the classes of their corresponding states: $U = \langle U_{c_1}, U_{c_2}, \ldots, U_{c_{|\mathcal{C}|}}\rangle$. Here U_{c_i} denotes the set of OVV's of states of class c_i. Together, the OVV's determine the task's optimal action set in its entirety.

Definition 3 *Let $U = \langle U_{c_1}, U_{c_2}, \ldots, U_{c_{|\mathcal{C}|}}\rangle$ and $\tilde{U} = \langle \tilde{U}_{c_1}, \tilde{U}_{c_2}, \ldots, \tilde{U}_{c_{|\mathcal{C}|}}\rangle$ be the OVV sets of the primary and auxiliary tasks, respectively. The* dissimilarity *of the primary and auxiliary tasks, denoted $\Delta(U, \tilde{U})$, is:*

$$\Delta(U, \tilde{U}) \stackrel{\text{def}}{=} \max_{c \in \mathcal{C}} \max_{\mathbf{u} \in U_c} \left\{\min_{\tilde{\mathbf{u}} \in \tilde{U}_c} ||\mathbf{u} - \tilde{\mathbf{u}}||_2\right\}.$$

Intuitively, dissimilarity $\Delta(U, \tilde{U})$ is the worst-case distance between an OVV in the primary task and the nearest OVV of the same class in the auxiliary task. The notion of dissimilarity allows us to establish the desired suboptimality bound (see Appendix for a proof):

Theorem 1 *Let $\tilde{\mathcal{A}}^*$ be the optimal action set of the auxiliary task. Replacing the full action set $\mathcal{A}$ with $\tilde{\mathcal{A}}^*$ reduces the highest attainable value of a state in the primary task by at most $\Delta(U, \tilde{U}) \cdot \sqrt{2}\gamma/(1-\gamma)$, where U and $\tilde{U}$ are the OVV sets of the primary and auxiliary tasks, respectively.*

Randomized Task Perturbation

Theorem 1 implies that learning with the transferred actions is safe if every OVV in the primary task has in its vicinity an OVV of the same class in the auxiliary task. We confirm this expectation below with action transfer across *similar* tasks. However, two *dissimilar* tasks can have very different OVV makeups and thus possibly different action sets. This section studies a detrimental instance of action transfer in light of Theorem 1 and proposes a more sophisticated approach that is robust to misleading auxiliary tasks.

Detrimental Action Transfer Consider the auxiliary and primary tasks in Figure 4. In one case, the goal is in the southeast corner; in the other, it is moved to a northwesterly location. The optimal policy for the auxiliary task, shown in Figure 4, includes only SOUTH and EAST actions. The primary task features all four directions of travel in its optimal policy. Learning the primary task with actions transferred from the auxiliary task is thus a largely doomed endeavor: the goal will be practically unreachable from most cells.

RTP action transfer To do well with unrepresentative auxiliary experience, the learner must sample the domain's OVV space not reflected in the auxiliary task. *Randomized task perturbation* (RTP) allows for a more thorough exposure to the domain's OVV space while learning in the same auxiliary task. The method works by internally distorting the optimal value function of the auxiliary task, thereby inducing an artificial *new* task while operating in the *same* en-

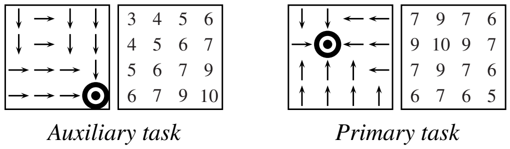

Figure 4: A pair of auxiliary and primary tasks, along with their optimal policies and value functions (rounded to integers).

vironment. RTP action transfer learns the optimal policy and optimal actions in the artificial and original tasks.

Figure 5 illustrates the workings of RTP action transfer. RTP distorts the optimal value function of the original task (Figure 5*a*) by randomly selecting a small fraction ϕ of the states and labeling them with randomly chosen values, drawn uniformly from $[v_{\min}, v_{\max}]$. Here $v_{\min} = r_{\min}/(1-\gamma)$ and $v_{\max} = r_{\max}/(1-\gamma)$ are the smallest and largest state values in the domain. The smallest and largest one-step rewards $r_{\min}$ and $r_{\max}$ are estimated or learned.

The selected states form a set $\mathcal{F}$ of *fixed-valued states*. Figure 5*b* shows these states and their assigned values on a sample run with $\phi = 0.2$. RTP action transfer learns the value function of the artificial task *by treating the values of the states in $\mathcal{F}$ as constant*, and by iteratively refining the other states' values via Q-learning. Figure 5*c* illustrates the resulting optimal values. Note that the fixed-valued states have retained their assigned values, and the other states' values have been computed with regard to these fixed values.

RTP created an artificial task quite different from the original. The optimal policy in Figure 5*d* features *all four directions of travel*, despite the goal's southeast location. We ignore the action choices in $\mathcal{F}$ since those states are semantically absorbing. The p components (not shown in the figure) of the resulting actions are in the useful range $[0.5, 0.65]$—a marked improvement over the full action set, in which 62% of the actions are in the useless range $(0.65, 0.9]$.

In terms of the formal analysis above, the combined (original + artificial) OVV set in RTP action transfer is closer to, or at least no farther from, the primary task's OVV set than is the OVV set of the original auxiliary task alone. The algorithm thereby reduces the dissimilarity of the two tasks and improves the suboptimality guarantees of Theorem 1. Figure 6 specifies RTP transfer embedded in Q-learning.

Notes on RTP action transfer RTP action transfer is easy to use. The algorithm's only parameter, ϕ, offers a trade-off: $\phi \approx 0$ results in an artificial task almost identical to the original; $\phi \approx 1$ induces an OVV space that ignores the domain's transition and reward dynamics and is thus not representative of tasks in the domain. Importantly, RTP action transfer requires no environmental interaction of its own—it reuses the $\langle s, a, r', s' \rangle$ quadruples generated while learning the unmodified auxiliary task. It may be useful to run RTP action transfer several times, using the combined action set over all runs. A data-economical implementation learns all artificial Q-value functions Q_1^+, Q_2^+, etc., within the same algorithm. The data requirement is thus the same as in traditional Q-learning. The space and running time requirements are a modest multiple k of those in Q-learning, where k is the number of artificial tasks learned.

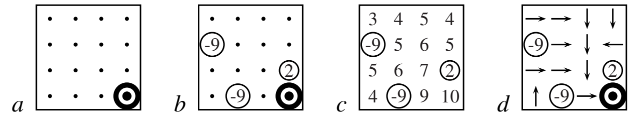

Figure 5: RTP action transfer at work: original auxiliary task (*a*); random choice of fixed-valued states and their values (*b*); new optimal value function (*c*, rounded to integers) and policy (*d*).

Add each $s \in \mathcal{S}$ to $\mathcal{F}$ with probability ϕ
foreach $s \in \mathcal{F}$
  do $\textit{random-value} \leftarrow \mathrm{rand}(v_{\min}, v_{\max})$
    $Q^+(s,a) \leftarrow \textit{random-value}$ for all $a \in \mathcal{A}$
repeat $s \leftarrow$ current state, $a \leftarrow \pi(s)$
    Take action a, observe reward r, new state s'
    $Q(s,a) \overset{\alpha}{\leftarrow} r + \gamma \max_{a' \in \mathcal{A}} Q(s', a')$
    if $s \in \mathcal{S} \setminus \mathcal{F}$
      then $Q^+(s,a) \overset{\alpha}{\leftarrow} r + \gamma \max_{a' \in \mathcal{A}} Q^+(s', a')$
  until converged
$\mathcal{A}^* = \cup_{s \in \mathcal{S}} \{\arg\max_{a \in \mathcal{A}} Q(s,a)\}$
$\mathcal{A}^+ = \cup_{s \in \mathcal{S} \setminus \mathcal{F}} \{\arg\max_{a \in \mathcal{A}} Q^+(s,a)\}$
return $\mathcal{A}^* \cup \mathcal{A}^+$

Figure 6: RTP action transfer in pseudocode. The left arrow indicates regular assignment; $x \overset{\alpha}{\leftarrow} y$ denotes $x \leftarrow (1-\alpha)x + \alpha y$.

While RTP action transfer is a product of the related-task formalism and suboptimality analysis above, it *does not rely on knowledge of the classes, outcomes, and state classification and next-state functions.* As such, it is applicable to any two MDP's with a shared action set. In the case of tasks that do obey the proposed formalism, the number of outcomes is the dimension of the domain's OVV space, and the number of classes is a measure of the heterogeneity of the domain's dynamics (few classes means large regions of the state space with uniform dynamics). RTP action transfer thrives in the presence of few outcomes and few classes. RTP action transfer will also work well if the same action is optimal for many OVV's, increasing the odds of its discovery and inclusion in the transferred action set.

Extensions to Continuous Domains RTP transfer readily extends to continuous state spaces. In this case, the set $\mathcal{F}$ cannot be formed from individual states; instead, $\mathcal{F}$ should encompass *regions* of the state space, each with a fixed value, whose aggregate area is a fraction ϕ of the state space. A practical implementation of RTP can use, e.g., *tile coding* (Sutton & Barto 1998), a popular function-approximation technique that discretizes the state space into regions and generalizes updates in each region to nearby regions. The method can be readily adapted to ensure that fixed-valued regions retain their values (e.g., by resetting them after every update).

Empirical Results

This section puts RTP action transfer to the test in several learning contexts, confirming its effectiveness.

Relevance-weighted action selection A valuable vehicle for exploiting action transfer is *action relevance*, which we define to be the fraction of states at which an action is optimal: $\text{RELEVANCE}(a) = |\{s \in \mathcal{S} : \pi^*(s) = a\}|/|\mathcal{S}|$. (In case of continuous-state domains, the policy π^* and the relevance computation are over a suitable discretization of the state space.) The ϵ-greedy action selection creates a substantial opportunity for exploiting the actions' relevances: *exploratory* action choices should select an action with probability equal to its relevance (estimated from the optimal solution to the auxiliary task and to its perturbed versions), rather than uniformly. The intuition here is that the likelihood of a given action a being optimal in state s is $\text{RELEVANCE}(a)$, and it is to the learner's advantage to explore its action options in s in proportion to their optimality potential in s.

We have empirically verified the benefits of relevance-weighted action selection and used it in all experiments below. This technique allows action transfer to accelerate learning even if it does not reduce the *number* of actions. In this case, information about the actions' *relevances* alone gives the learner an appreciable advantage over the default (learning with the full action set and uniform relevances).

Methodology and Parameter Choices We used Q-learning with $\epsilon = 0.1$, $\alpha = 0.1$, and optimistic initialization (to 10, the largest value in the domain) to compare the performance of the optimal, transferred, and full action sets in the primary task shown in Figure 2. The optimal action set was the actual set of optimal actions on the primary task, in the given discretization of the action space. The transferred action sets were obtained from the auxiliary tasks of Figure 7 by regular transfer in one case and by RTP transfer in the other ($\phi = 0.1$ and 10 trials, picked heuristically and not optimized). Regular and RTP action transfer required 1 million episodes and an appropriate annealing régime to solve the auxiliary tasks optimally. That many episodes would be needed in any event to solve the auxiliary tasks, so the knowledge transfer generated no overhead.

The experiments used relevance-weighted ϵ-greedy action selection. All the 164 actions in the full set were assigned the default relevance of $1/164$. In the transferred action sets, the relevance of an action was computed by definition from the optimal policy of the auxiliary task; in the case of RTP transfer, the relevances were averaged over all trials.

For function approximation in the p dimension, we used tile coding (Sutton & Barto 1998). Grid world episodes started in a random cell and ran for 100 time steps, to avoid spinning indefinitely in absorbing goal/quicksand states. The performance criterion was the *highest* average state value under any policy discovered, vs. the number of episodes completed. This performance metric was computed from the learner's policies using an external policy evaluator (value iteration) and was unrelated to the learner's own imperfect value estimates.

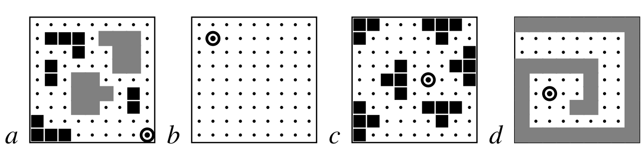

Figure 7: Auxiliary maps used in the experiments.

Results Figure 8 plots the performance of the four action sets with different auxiliary tasks. The top of the graph (average state value ≈ 4.28) corresponds to optimal behavior. The optimal and full action-set curves are repeated in all graphs because they do not depend on the auxiliary task (however, note the different y-scale in Figure 8*a*).

The optimal action set is a consistent leader. The performance of regular transfer strongly depends on the auxiliary map. The first map's optimal action set features only EAST and SOUTH actions, leaving the learner unprepared for the test task and resulting in worse performance than with the full action set. Performance with the second auxiliary map is not as abysmal but is far from optimal. This is because map b does not feature slow EAST and SOUTH actions, which are common on the test map. The other two auxiliary tasks' optimal action sets resemble the test task's, allowing regular action transfer to tie with the optimal set.

RTP transfer, by contrast, consistently rivals the optimal action set. The effect of the auxiliary task on RTP transfer is minor, resulting in performance superior to the full action set *even with misleading auxiliary experience*. These results show the effectiveness of RTP transfer and the comparative undesirability of learning with the full and transferred action sets. We have verified that RTP transfer substantially improves on *random* selection of actions for the partial set. In fact, such randomly-constructed action sets perform more poorly than even the full set, past an initial transient.

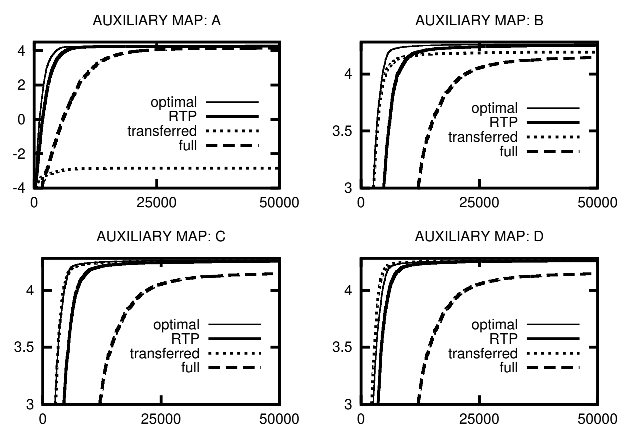

Figure 8: Comparative performance. Each curve is a point-wise average over 100 runs. At a 0.01 significance level, the ordering of the curves is: T<F< {RTP, O} (map *a*, starting at 5000); F<T< {RTP, O} (map *b*, starting at 17000). F< {T, RTP, O} (maps *c*–*d*, starting at 100).

Related Work

Knowledge transfer has been applied to hierarchical (Hauskrecht *et al.* 1998; Dietterich 2000), first-order (Boutilier, Reiter, & Price 2001), and factored (Guestrin *et al.* 2003) MDP's. A limitation of this related research is the reliance on a human designer for an explicit description of the regularities in the domain's dynamics, be it in the form of matching state regions in two problems, a hierarchical policy graph, relational structure, or situation-calculus fluents and operators. RTP action transfer, while inspired by an analysis using outcomes, classes, and state classification and next-state functions, requires none of this information. It discovers and exploits the domain's regularities to the extent that they are present and requires no human guidance along the way. Furthermore, our method is robust to unrepresentative auxiliary experience.

In addition, the longstanding tradition in RL has been to attack problem complexity on the *state* side. For example, the above methods identify regions of the state space with similar optimal behavior. By contrast, our method simplifies the problem by identifying useful *actions*. A promising approach would be to combine these two lines of work.

Conclusion

This paper presents *action transfer*, a novel approach to knowledge transfer across tasks in domains with large action sets. The algorithm rests on the idea that actions relevant to an optimal policy in one task are likely to be relevant in other tasks. The contributions of this paper are: (i) a formalism isolating the commonalities and differences among tasks within a domain, (ii) a formal bound on the suboptimality of action transfer, and (iii) action transfer with *randomized task perturbation* (RTP), a more sophisticated and empirically successful knowledge-transfer approach inspired by the analysis of regular transfer. We demonstrate the effectiveness of RTP empirically in several learning settings. We intend to exploit RTP's potential to handle truly continuous action spaces, rather than merely large, discretized ones.

Acknowledgments

The authors are thankful to Raymond Mooney, Lilyana Mihalkova, and Yaxin Liu for their feedback on earlier versions of this manuscript. This research was supported in part by NSF CAREER award IIS-0237699, DARPA award HR0011-04-1-0035, and an MCD fellowship.

References

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for first-order MDPs. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 690–697.

Crites, R. H., and Barto, A. G. 1996. Improving elevator performance using reinforcement learning. In Touretzky, D. S.; Mozer, M. C.; and Hasselmo, M. E., eds., *Advances in Neural Information Processing Systems 8*. Cambridge, MA: MIT Press.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research* 13:227–303.

Gaskett, C.; Wettergreen, D.; and Zelinsky, A. 1999. Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, 417–428.

Guestrin, C.; Koller, D.; Gearhart, C.; and Kanodia, N. 2003. Generalizing plans to new environments in relational MDPs. In *Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*.

Hauskrecht, M.; Meuleau, N.; Kaelbling, L. P.; Dean, T.; and Boutilier, C. 1998. Hierarchical solution of Markov decision processes using macro-actions. In *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 220–229.

Santamaria, J. C.; Sutton, R. S.; and Ram, A. 1997. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior* 6(2):163–217.

Stone, P., and Sutton, R. S. 2001. Scaling reinforcement learning toward RoboCup soccer. In *Proc. 18th International Conference on Machine Learning (ICML-01)*, 537–544. Morgan Kaufmann, San Francisco, CA.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tesauro, G. 1994. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation* 6(2):215–219.

Watkins, C. J. C. H. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, Cambridge University.

Proof of Theorem 1

Lemma 1 *Let* $\tilde{U} = \langle \tilde{U}_{c_1}, \tilde{U}_{c_2}, \ldots, \tilde{U}_{c_{|\mathcal{C}|}} \rangle$ *be the auxiliary task's OVV set, and let* $\tilde{\mathcal{A}}^*$ *be the corresponding action set. Then* $\max_{a\in\mathcal{A}}\{r(c,a) + \gamma\mathbf{t}(c,a)\mathbf{v}\} - \max_{a\in\tilde{\mathcal{A}}^*}\{r(c,a) + \gamma\mathbf{t}(c,a)\mathbf{v}\} \le \sqrt{2}\gamma \min_{\mathbf{u}\in\tilde{U}_c}\{||\mathbf{v}-\mathbf{u}||_2\}$ *for all* $\mathbf{v} \in \mathbb{R}^{|\mathcal{O}|}$ *and* $c \in \mathcal{C}$.

Proof: Let $a_\mathbf{v} = \arg\max_{a\in\mathcal{A}}\{r(c,a) + \gamma\mathbf{t}(c,a)\mathbf{v}\}$. Let $a_\mathbf{u} = \arg\max_{a\in\mathcal{A}}\{r(c,a) + \gamma\mathbf{t}(c,a)\mathbf{u}\}$ for an arbitrary $\mathbf{u} \in \tilde{U}_c$, so that $a_\mathbf{u} \in \tilde{\mathcal{A}}^*$. We immediately have: $r(c,a_\mathbf{v}) + \gamma\mathbf{t}(c,a_\mathbf{v})\mathbf{u} \le r(c,a_\mathbf{u}) + \gamma\mathbf{t}(c,a_\mathbf{u})\mathbf{u}$. Therefore,

$$
\begin{aligned}
&\max_{a\in\mathcal{A}}\{r(c,a) + \gamma\mathbf{t}(c,a)\mathbf{v}\} - \max_{a\in\tilde{\mathcal{A}}^*}\{r(c,a) + \gamma\mathbf{t}(c,a)\mathbf{v}\}\\
&\quad\le [r(c,a_\mathbf{v}) + \gamma\mathbf{t}(c,a_\mathbf{v})\mathbf{v}] - [r(c,a_\mathbf{u}) + \gamma\mathbf{t}(c,a_\mathbf{u})\mathbf{v}]\\
&\quad= [r(c,a_\mathbf{v}) - r(c,a_\mathbf{u})] - [\gamma\mathbf{t}(c,a_\mathbf{u})\mathbf{v} - \gamma\mathbf{t}(c,a_\mathbf{v})\mathbf{v}]\\
&\quad\le [\gamma\mathbf{t}(c,a_\mathbf{u})\mathbf{u} - \gamma\mathbf{t}(c,a_\mathbf{v})\mathbf{u}] - [\gamma\mathbf{t}(c,a_\mathbf{u})\mathbf{v} - \gamma\mathbf{t}(c,a_\mathbf{v})\mathbf{v}]\\
&\quad= \gamma[\mathbf{t}(c,a_\mathbf{u}) - \mathbf{t}(c,a_\mathbf{v})]\cdot[\mathbf{u}-\mathbf{v}]\\
&\quad\le \gamma||\mathbf{t}(c,a_\mathbf{u}) - \mathbf{t}(c,a_\mathbf{v})||_2 \cdot ||\mathbf{u}-\mathbf{v}||_2 \le \sqrt{2}\gamma\cdot||\mathbf{u}-\mathbf{v}||_2.
\end{aligned}
$$

Since the choice of $\mathbf{u} \in \tilde{U}_c$ was arbitrary and any other member of $\tilde{U}_c$ could have been chosen in its place, the lemma holds. □

Let V^* and $\tilde{V}^*$ be the optimal value functions for the primary task $\langle\mathcal{S},\kappa,\eta\rangle$ using $\mathcal{A}$ and $\tilde{\mathcal{A}}^*$, respectively. Let $\delta = \max_{s\in S}\{V^*(s) - \tilde{V}^*(s)\}$. Then for all $s \in \mathcal{S}$,

$$
\begin{aligned}
\tilde{V}^*(s) &= \max_{a\in\tilde{\mathcal{A}}^*}\left\{r(\kappa(s),a) + \gamma\textstyle\sum_{o\in\mathcal{O}} t(\kappa(s),a,o)\tilde{V}^*(\eta(s,o))\right\}\\
&\ge \max_{a\in\tilde{\mathcal{A}}}\left\{r(\kappa(s),a) + \gamma\textstyle\sum_{o\in\mathcal{O}} t(\kappa(s),a,o)V^*(\eta(s,o))\right\} - \gamma\delta.
\end{aligned}
$$

Applying Lemma 1 and denoting by $\mathbf{v}$ the OVV corresponding to s in U, we obtain:

$$
\begin{aligned}
\tilde{V}^*(s) &\ge V^*(s) - \sqrt{2}\gamma \min_{\tilde{\mathbf{u}}\in\tilde{U}_{\kappa(s)}} ||\mathbf{v}-\tilde{\mathbf{u}}||_2 - \gamma\delta\\
&\ge V^*(s) - \sqrt{2}\gamma \max_{c\in\mathcal{C}}\max_{\mathbf{u}\in U_c}\{\min_{\tilde{\mathbf{u}}\in U_c}||\mathbf{u}-\tilde{\mathbf{u}}||\} - \gamma\delta\\
&= V^*(s) - \sqrt{2}\gamma\Delta(U,\tilde{U}) - \gamma\delta.
\end{aligned}
$$

Hence, $V^*(s) - \tilde{V}^*(s) \le \delta \le \sqrt{2}\gamma\Delta(U,\tilde{U}) + \gamma\delta$, and $V^*(s) - \tilde{V}^*(s) \le \Delta(U,\tilde{U})\cdot\sqrt{2}\gamma/(1-\gamma)$ for all $s \in \mathcal{S}$. □