Transfer Learning with Probabilistic Mapping Selection

Anestis Fachantidis
Department of Informatics
Aristotle University of Thessaloniki
afa@csd.auth.gr

Ioannis Partalas
University of Grenoble Alpes
LIG
F38000, Grenoble, France
ioannis.partalas@imag.fr

Matthew E. Taylor
School of Electrical Engineering and Computer Science
Washington State University
taylorm@eecs.wsu.edu

Ioannis Vlahavas
Department of Informatics
Aristotle University of Thessaloniki
vlahavas@csd.auth.gr

Abstract

When transferring knowledge between reinforcement learning agents with different state representations or actions, past knowledge must be efficiently mapped to novel tasks so that it aids learning. The majority of the existing approaches use pre-defined mappings provided by a domain expert. To overcome this limitation and enable autonomous transfer learning, this paper introduces a method for weighting and using multiple inter-task mappings based on a probabilistic framework. Experimental results show that the use of multiple inter-task mappings, accompanied with a probabilistic selection mechanism, can significantly boost the performance of transfer learning relative to 1) learning without transfer and 2) using a single hand-picked mapping. We especially introduce novel tasks for transfer learning in a realistic simulation of the iCub robot, demonstrating the ability of the method to select mappings in complex tasks where human intuition could not be applied to select them. The results verified the efficacy of the proposed approach in a real world and complex environment.

1 Introduction

Multiple *transfer learning* (TL) methods have recently been developed for *reinforcement learning* (RL) tasks (Fernández and Veloso, 2006; Lazaric et al., 2008; Taylor et al., 2008a,b; Ammar et al., 2012). Typically, when an RL agent leverages TL, it uses knowledge acquired in one or more (*source*) tasks to speed up its learning in a more complex (*target*) task.

Although the majority of such work presumes that the source task is connected in an obvious or natural way with the target task, this may not be the case in many real life applications. Indeed, many pairs of tasks will have different state and action spaces, or different reward and transition functions. One approach is to use functions that map the state and action variables of the source task to state and action variables of the target task, called *inter-task mappings* (Taylor and Stone, 2009).

While inter-task mappings have indeed been used successfully in several settings (Taylor and Stone, 2009; Torrey et al., 2005), this work addresses two important shortcomings. First, an agent typically uses a hand-coded mapping, requiring the knowledge of a domain expert. If human intuition can not be applied to the problem, selecting an inter-task mapping may require significant time and extensive experimentation, both of which may be in short supply in complex domains. Second, even if a correct mapping is used, it is fixed and applied to the entire state-action space, ignoring the possibility that different mappings may be better suited in different regions of the target task's state space. Furthermore, the performance of an agent using a single fixed mapping is bound by the quality of this mapping; if an expert selects the wrong mapping, learning performance can be significantly degraded.

This work alleviates these problems by contributing a method for automated on-line selection of the appropriate inter-task mappings without relying on human knowledge. An important insight of the method comes from the *multi-task* TL problem, where knowledge from multiple source tasks (and not mappings) can be transferred to a single target task.

The main contributions of this work are to:

1. Present a novel viewpoint in which every *multiple mapping* TL problem is equivalent to a *multi-task* TL problem, showing the relationship between two problem settings previously considered distinct.

2. Alleviate the problem of manually pre-defining a mapping between a source and target task in TL procedures by presenting a fully automated method for selecting both state and action inter-task mappings.

3. Propose the first multiple-mappings TL method, which means not only selecting a single best mapping but dynamically changing its selection based on data gathered in the target task and according to the agent's state in the target task.

4. Introduce and evaluate a novel algorithm, *COmpliance aware transfer for Model Based REinforcement Learning* (COMBREL), that implements the proposed method.

5. Provide empirical results both in a simple simulation of mountain car and in a new complex TL domain based on the simulator of the iCub robotic platform, showing that this TL method not only leads to to superior performance, compared to single mapping methods but it can also be applied to tasks more complex than simpler domains used by many TL papers in the literature.

Finally, the results presented in this article support its key insight on the *connection of multiple mappings and multi-task* TL problems.

The remainder of this article is organized as follows. Section 2 presents background information on reinforcement learning and transfer learning, and introduces some key concepts in multi-task transfer learning. Section 3 describes the proposed approach and Section 4 the domains of experimentation. Section 5 describes the experimental setup and presents results. Section 6 contrasts this method with related work. Section 7 concludes and gives a selection of future work.

2 Background

This section presents the necessary background to understand the algorithms developed in this article. Brief introductions are provided to reinforcement learning and transfer learning, which are then followed by a more detailed discussion on multi-task learning.

2.1 Reinforcement Learning

Reinforcement Learning addresses the problem of how an agent can learn a behavior through trial-and-error interactions with a dynamic environment (Sutton and Barto, 1998). In an RL task the agent, at each time step t, senses the environment's state, $s \in S$, where S is the finite set of possible states, and selects an action $a \in A(s)$ to execute, where $A(s)$ is the finite set of possible actions in state s. The agent then moves to a new state s' according to a transition function $\mathcal{P}_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ and receives a reward, $r \in \mathbb{R}$ generated

by a reward function $\mathcal{R}_a(s, s')$. The general goal of the agent is to maximize the expected return, where the return, R, is defined as some specific function of the reward sequence. The outcome will be an action-value function $Q^\pi(s, a)$ which expresses the expected return starting from s, taking action a, and following thereafter policy $\pi : S \rightarrow A$, which dictates how the agent acts in a certain situation in order to maximize the reward received over time.

This article focuses on model-based (also called *model learning*) RL. This class of algorithms, which includes algorithms such as Fitted R-max and PEGASUS, use experiences to learn a model of the transition and reward functions of a task. These models are used either to produce new mental experiences for direct learning (e.g., Q-Learning), or with dynamic programming methods and planning in order to construct a policy (Brafman and Tennenholtz, 2003; Jong and Stone, 2007; Ng et al., 2006). Fitted R-Max approximates an action-value function Q by constructing an MDP over a small and finite sample of states $\hat{S} \subset S$. First, it estimates the transition function $\mathcal{P}_{\widehat{S}}(s'|s, a)$ for a state-action pair $\langle \hat{s}, a \rangle$, where $\hat{s} \in \hat{S}$, using all the available data for action a and the states near $\hat{s}$. A model breadth parameter b controls the degree of generalization used to estimate $\mathcal{P}_{\widehat{S}}(s'|s, a)$. It then approximates the probability distribution over predicted successor states in S with a probability distribution over predicted successor states in $\hat{S}$. This results in a finite stochastic MDP for the state space $\hat{S}$ to which we can apply Dynamic Programming to acquire an approximation of the Q action-value function for S. In this work, Fitted R-Max is the algorithm used in all experiments, both in combination with transfer learning methods and without any transfer learning.

2.2 Single-Task Transfer Learning

The type of knowledge that can be transferred between tasks varies among different TL methods, including value functions, entire policies, or a set of samples from a source task used for a batch RL algorithm in a target task.

In order to enable transfer learning across tasks with different state variables (i.e., different state representation) or action sets, one must define how pairs of tasks are related. One way to represent this relation is to use a pair $\langle X_S, X_A \rangle$ of inter-task mappings (Taylor et al., 2007), where the function X_S maps a target task state variable to a source task state variable and X_A maps an action in the target task to an action in the source task.

TIMBREL (Taylor et al., 2008a) is an example of a transfer learning algorithm that uses a single inter-task mapping. TIMBREL enhances a model-based RL algorithm like Fitted R-Max by assisting it in model approximation. To do so, it transfers source task instances near the target task points that need to be approximated. These instances, in the form of tuples $\tau_i = \langle s_i, a_i, s'_i, r_i \rangle$, describe the

agent's experiences in the source task. These instances are *translated* to the target task using a *pre-defined* inter-task mapping. Each one of the translated instances is assigned a weight w_i depending on the Euclidean distance of the current state to s_i. Once an instance's weight fails to increase the cumulative weight by at least a *minimum fraction* parameter, the remaining instances contribution is considered negligible and these are not transferred. When the accumulated weight fails to reach a threshold equal to a *minimum weight* parameter w, TIMBREL does not transfer and continues with the exploration strategy of Fittted R-Max.

2.3 Multi-Task Transfer Learning

In a multi-task transfer learning problem (Wilson et al., 2007), an agent is transferring knowledge from more than one source tasks into a (typically) more complex target task that is related to the source tasks. To avoid *negative transfer*, where TL actually degrades learning performance, the agent must decide 1) what information to transfer, 2) from which source task, and 3) when to transfer this information (Fernández and Veloso, 2006; Lazaric et al., 2008).

Relevant to this setting, two probabilistic measures have been defined: *compliance* and *relevance* (Lazaric et al., 2008). These measures can assist an agent to determine the most similar source tasks to the target task (compliance) and also to select which instances to transfer from these source tasks (relevance).

Compliance is defined as follows. Let S be a source task and let $\tau_i = \langle s_i, a_i, s'_i, r_i \rangle$ be an experience tuple in a target task T, where s_i is a target task state, a_i an action applied there, s'_i the afterstate and r_i the reward received. Compliance λ_i of τ_i to S is the probability of this sample to be generated by S given a set of source samples $\widehat{S}$, that is:

$$\lambda_i = P(\tau_i|\widehat{S}) = \mathcal{P}_{\widehat{S}}(s'_i|s_i, a_i)\mathcal{R}_{\widehat{S}}(r_i|s_i, a_i) \tag{1}$$

where $\mathcal{P}_{\widehat{S}}$ and $\mathcal{R}_{\widehat{S}}$ represent an approximation of the transition and reward models respectively. To approximate these models, the authors in (Lazaric et al., 2008) also define the compliance of an experience sample of the target task to an *individual* source task sample.

Let $\tau_i = \langle s_i, a_i, s'_i, r_i \rangle$ and $\sigma_j = \langle s_j, a_j, s'_j, r_j \rangle$ be two experience tuples of a target task and a source task respectively. $\phi(x) = \exp(-x^2/\delta)$ is a Gaussian kernel function with bandwidth δ and will be applied to an Euclidean distance metric d. For the transition models of the two tasks, the compliance *of τ_i with respect to σ_j* is defined as:

$$\lambda^P_{ij} = w_{ij} \cdot \phi\left(\frac{d(s'_i, s_i + (s'_j - s_j))}{\delta_{s'}}\right)$$

where

$$w_{ij} = \frac{\phi(\frac{d(\langle s_i,a_i\rangle,\langle s_j,a_j\rangle)}{\delta_{sa}})}{\sum_{l=1}^{m}\phi(\frac{d(\langle s_i,a_i\rangle,\langle s_l,a_l\rangle)}{\delta_{sa}})} \tag{2}$$

The first term (w_{ij}) of λ_{ij}^{P} expresses the relative closeness of two samples' starting states and actions, while the second term expresses the similarity of the outcomes relative to the respective initial states. The reward function's compliance for τ_i, is defined (with respect to σ_j) as:

$$\lambda_{ij}^{R} = w_{ij} \cdot \phi\left(\frac{|r_i - r_j|}{\delta_r}\right)$$

The approximated transition and reward models in Equation 1 are the average of the compliance between τ_i and all the samples in $\hat{S}$ and so Equation 1 becomes:

$$\lambda_i = P(\tau_i|\widehat{S}) = \frac{1}{Z^P Z^R}(\sum_{j=1}^{m}\lambda_{ij}^{P})(\sum_{j=1}^{m}\lambda_{ij}^{R}) \tag{3}$$

where the first summation represents the approximation of the transition function, the second summation approximates the reward function and Z^P and Z^R are normalization terms such as $Z^P = \sum_{i=1}^{t}\sum_{j=1}^{m}\lambda_{ij}^{P}$, $Z^R = \sum_{i=1}^{t}\sum_{j=1}^{m}\lambda_{ij}^{R}$ and $t = |\hat{T}|$, the size of the target task sample set.

The compliance between the entire target task and the entire source task S is defined as: $\Lambda = \frac{1}{t}\sum_{i=1}^{t}\lambda_i P(s)$ where $t = |\hat{T}|$, the size of the target task sample set, λ_i is the compliance of each instance in the target task, defined as in Equation 3 and $P(s)$ is a prior over the source task instances.

Although compliance measures which source tasks are likely to be most relevant for transfer, it does not help decide which samples in $\hat{S}$ are best to transfer. For this, *relevance* was proposed as a quantity related to the compliance of a source task sample σ_j to the target task T, when the number of samples in $\hat{T}$ close to σ_j are sufficient. Where there are insufficient samples in the target task, relevance tends to have higher values based on the optimistic insight that when you do not have enough information about an approximated sample (i.e., there are no other samples close to it) it is better to transfer knowledge than to ignore it (Lazaric et al., 2008).

Let the samples τ_i in $\hat{T}$ be sorted in descending order according to their weight w_{ij} (see Equation 2) The average distance between σ_j and the target task samples is defined as:

$$d_j = \frac{1}{h_j}\sum_{i=1}^{h_j} d(\langle s_j, a_j\rangle, \langle s_i, a_i\rangle) \tag{4}$$

where h_j is such that $\sum_{i=1}^{h_j} w_{ji} < \mu$, where $\mu \in (0; 1]$ is the fraction of the total number of samples considered in the computation of the average distance. The relevance of a source task instance σ_j is defined as: $\rho_j = \rho(\lambda_j, d_j) = e^{-(\frac{\lambda_j - 1}{d_j})^2}$ where λ_j is defined in the same way as in Equation 3 but instead of using the source task S we now use the target task T and the source task transition σ_j.

3 Transferring with Multiple Inter-Task Mappings

The approach introduced in this article has three objectives. First, the TL method should enable transfer of knowledge between two related RL tasks while allowing for significant differences between them (e.g., different state and action variables). Second, the TL method should not rely on any *a priori* assumptions about how the state and action variables of the two tasks are connected. Third, the TL method should be flexible enough to allow different mappings in different areas of the state space.

To achieve these objectives, the algorithm described in Section 3.3 autonomously selects mappings based on an on-line probabilistic evaluation from the set of all possible inter-task mappings. The proposed method transfers knowledge from a source task in the form of instances $\langle s, a, r, s' \rangle$ (experience tuples). This choice is well suited for algorithms that implement mappings evaluation, since comparing instances is a straightforward way to compare task dynamics. Moreover, since instances (transitions) represent the lowest level of knowledge we can acquire from a task, they consequently contain all the required information to generate (local) models of an environment's dynamics from them. Another advantage of transferring instances, compared to directly transferring a value function, is that the source task agent's learning method and policy are irrelevant to the outcome of the TL process.

In our setting we assume that we can generate *or be provided with* a set $\mathcal{X}^S = \{X_i^S, i = 1 \ldots |\mathcal{X}^X|\}$ of state variable mappings and a set $\mathcal{X}^A = \{X_j^A, j = 1 \ldots |\mathcal{X}^A|\}$ of action mappings. Based on these sets we generate the set $\mathcal{X} = \{(X_i^S, X_i^A)\}_{k=1}^m$ which contains pairs of state and action mappings. For small state spaces the set $\mathcal{X}$ can consist of all the combinations of the available state and action mappings. In more complex environments with a large number of state variables and actions one could limit the set $\mathcal{X}$ to contain a manageable subset of pairs of mappings.

An outline for the proposed method continues as follows:

1. Record samples of an agent acting in a source task.

2. An agent begins acting in a target task while also recording its experiences.

3. While continuing to learn in the target task, source task and target task instances are used for a constant, on-line evaluation of each available pair of inter-task mappings in $\mathcal{X}$.

4. The estimated best mapping is selected and used to "translate" source task instances to compatible target task instances, where compatibility is defined in terms of state and action variables.

5. The translated instances are transferred to the target task, where an RL algorithm there is able to efficiently use them, for example, to approximate the dynamics of the target task and assist its learning.

While the first two steps of the method are implementation-level details, the third step of the method contains the key insight of this work on the evaluation and selection of inter-task mappings.

Specifically, each inter-task mapping function is considered to be a hypothesis, proposed to match the geometry and dynamics between a source task and a target task. Mapping states and actions from a target task to a source task not only transforms the way we view and use the source task, but also the way it behaves and responds to a fixed target task's state-action query (because the latter has to be mapped). Thus, every inter-task mapping X_i can be considered to be a constructor for a new *virtual source task*, S_{X_i}. This re-formulates the problem of **finding the best mapping as a problem of finding the most compliant virtual source task** (with compliance defined as in Section 2). Additionally, this re-formulation transforms the problem of finding which instances to transfer through a specific mapping to the problem of determining sample relevance (as defined in Section 2). These reformulations allow us to use multi-task TL algorithms as the base for a novel solution to the multiple-mappings TL problem.

Therefore the rest of this section extends the multi-task transfer method discussed in Section 2 to use multiple inter-task mappings, rather than multiple source tasks. The proposed method is designed to allow for the added flexibility of different state and action sets between the source and the target task. In contrast, multi-task transfer learning typically assumes that source tasks and the target task have the same state variables and actions (Taylor and Stone, 2009). In the following, Section 3.1 builds the foundations of the proposed method and Section 3.2 presents the novel algorithm COMBREL which takes advantage of the novel viewpoint proposed in this article.

3.1 Compliance and Relevance for Multiple Mappings

We first re-define compliance and relevance described earlier in this text, to incorporate the idea of virtual source tasks. The compliance λ_{X_k,τ_i} of a target task transition τ_i and a virtual source task S_{X_k} (constructed by mapping X_k), given a set of $|\hat{S}_{X_k}|$ translated source samples is:

$$\lambda_{X_k,\tau_i} = \frac{1}{Z^P Z^R}(\sum_{j=1}^{m} \lambda_{ij}^P)(\sum_{j=1}^{m} \lambda_{ij}^R), \tag{5}$$

where $m = |\hat{S}_{X_k}|$ and P and R refer respectively to the transition and reward functions of the target task and $\lambda_{ij}^P, \lambda_{ij}^R$ are the transition and reward compliances of a single target task tuple i to a single source task tuple j.

The compliance between the target task (all of its instances, not only one of them, as in Equation 5) and the virtual source task S_{X_k} generated by the mapping X_k is defined as:

$$\Lambda_{X_k} = \frac{1}{t}\sum_{i=1}^{t} \lambda_{X_k,\tau_i} P(S_X) \tag{6}$$

where $t = |\hat{T}|$, the size of the target task sample set and $P(S_X)$ is a prior over all the mappings (i.e., virtual source tasks), simply expressing the probability of selecting an instance from a specific virtual source task if you would randomly select from all the instances of all the available mappings. The relevance of a virtual source task instance σ_j is:

$$\rho_j = \rho(\lambda_{T,\sigma_j}, d_j) = e^{-(\frac{\lambda_{T,\sigma_j}-1}{d_j})^2} \tag{7}$$

where λ_{T,σ_j} is defined similarly to Equation 5 but instead of iterating over the instances of the virtual source task S_{X_k}, it now iterates over the target task instances in $\hat{T}$ and uses the source task sample σ_j. d_j is an average distance between σ_j and the target task samples (see Equation 4).

The above definitions of compliance and relevance are almost identical to those originally defined (see Section 2), the only modification being in the notation used, where the multiple source tasks S_k of the original setting, are now referred as virtual source tasks S_{X_k} each one corresponding to a single state-action mapping X_k. However the change in the ideas involved is significant. In our context, that of multiple mappings, compliance quantifies the similarity between target task samples and samples from a *virtual* source task S_{X_i}, generated from mapping X_i.

Relevance can now be interpreted as a measure of transfer eligibility for a given virtual source task instance. When the target task's sample set has instances that

are near a virtual source task instance and are also highly compliant with it, the relevance is high. Furthermore, relevance will also be high for a virtual source task instance when there are no close target task instances:

> "The assumption underlying the definition of relevance is that, whenever there is no evidence against the transfer of a sample, it is convenient to transfer it to the target task." (Lazaric et al., 2008)

Relevance is essentially a sorting criteria for the samples of the most compliant virtual source task, assigning a transfer priority to them. In contrast, other approaches such as that of TIMBREL (Taylor et al., 2008a), prioritize the source task samples according to their distances from a target task sample.

For a written example of using the equations presented in this section to the context of discovering an inter-task mapping, the reader is suggested to follow the link provided in this article. [1]

3.2 COMBREL

To use our transformed definitions of compliance and relevance in our proposed method, we design a novel multiple-mappings TL algorithm, *COmpliance aware transfer for Model Based REinforcement Learning* (COMBREL), which is able to select inter-task mappings based on their compliance and relevance, as defined above.

In order for the agent to select a mapping when in a state s, the compliance to each of the virtual source tasks (mappings) must be computed. However, this introduces a problem since when learning on-line the after-states of executing action a in state s are not known beforehand and by definition, compliance can only be calculated for *complete target task instances* (complete transitions with their after-states s'). Moreover, analytically calculating compliances in each state visited would significantly increase the computational complexity of the proposed method. In order to tackle this issue a key implementation idea behind COMBREL is to *pre-compute* the compliance for a limited set of recorded complete target task transitions in the first training episode and then use these pre-calculated compliances, to approximate the compliance of any other target task state, to each of the mappings. An agent in state s thus averages the pre-computed compliances of nearby states (nearest neighbors) in order to estimate its current state compliance to each of the mappings. For this online estimation of compliance COMBREL uses k-nearest neighbors regression (kNN).

[1] A written example of calculating the compliances of inter-task mappings is available at `http://lpis.csd.auth.gr/COMBREL/example.pdf`

Using the subset of the k-nearest target task instances to the current target task state provides also the flexibility of not having just a single best mapping for the entire target task, but possibly different ones for state space subsets being formed at each step. The algorithm thus dynamically adapts to target task anomalies by recalculating the most useful (compliant) mapping at each step. The described process is computationally feasible as these target task state space subsets near the current state include instances for which compliance, and thus the best mapping, has already been calculated from the algorithm. These pre-calculated compliances of the nearest target task instances to each of the mappings need only to be averaged using kNN regression to find the best mapping for any current state.

COMBREL, just like TIMBREL, can interface with any underlying instance-based RL algorithm, assisting its model approximation process by transferring source task instances that are near the target task points that need to be approximated. In the absence of such source task instances, no transfer takes place.

COMBREL is presented in Algorithm 1. Lines 3–5 of the algorithm record source task transitions and generate the set of possible inter-task mappings. Each mapping X_i is considered to be a virtual source task S_{X_i} and can be used interchangeably. In lines 6–14 a number of target task transitions (instances) are recorded in the first episode. For each of these instances, the algorithm computes its compliance to each of the virtual source tasks, S_{X_i} (lines 9–13). In lines 15–16, for all episodes, if the agent's model-based algorithm is unable to approximate a state with its current data (e.g., when initially there are not enough nearby target task instances), attempts to transfer source task data. In this case T and R represent the transition and the reward functions of the target task MDP, respectively.

To select the best mapping for the current agent's state, lines 17–19 form a subset of the k-nearest recorded target task instances (neighbors), with respect to the current state, and then compute the average compliance of that target task subset to each of the virtual source tasks (mappings). The virtual source task with the maximum average compliance defines the best mapping (line 20). In the rest of the algorithm, the agent computes the relevance of each sample in that (most compliant) virtual source task (line 21). The most relevant samples are transferred and added to the target task samples used for model approximation and the algorithm continues learning.

To better compare with previous work, we also consider ablating COMBREL so that it can also be used without relevance (only task compliance) so that the samples of the virtual source task are prioritized (ordered) according to their Euclidean distance to the target task state (just like TIMBREL) and not according to their relevance. Finally, after transfer, the algorithm will gather more target task data over time and rely less on data transferred from the source task.

Algorithm 1 COMBREL

$m \leftarrow$ number of source task transitions to record
$k \leftarrow$ number of nearest target task instances to be used to estimate compliance
Learn in the source task, recording m (s, a, r, s') transitions.
Generate $|X|$ pairs (X^S, X^A) of inter-task mappings
Each mapping pair X_i, is considered a virtual source task S_i
while training in the target task **do**
    while in the first training episode **do**
        Record target task transitions τ_i
        for (each virtual source task S_i) **do**
            Compute transition compliance λ_{ij}^P of τ_i for each $\sigma_j \in S_i$
            Compute reward compliance λ_{ij}^R of τ_i for each $\sigma_j \in S_i$
            Compute compliance λ_i of τ_i to the virtual source task S_i
        end for
    end while
    for (each episode) **do**
        if the model-based RL algorithm is unable to accurately estimate $T(x_T, a_T)$ or $R(x_T, a_T)$ **then**
            Find the k-nearest recorded target task instances to $\langle x_T, a_T \rangle$
            while $T(x_T, a_T)$ or $R(x_T, a_T)$ does not have sufficient data **do**
                Calculate the average compliance Λ_i of the k-nearest target task instances to each virtual source task S_i
                Find the best mapping where $X^{best} \leftarrow X_{\arg\max(\Lambda_i)}$ ▷ The most compliant
                Compute relevance ρ_j of each source task instance $\sigma_j \in X^{best}$ to $\langle x_T, a_T \rangle$
                Add the most relevant samples σ_j to the current model for $\langle x_T, a_T \rangle$.
            end while
        end if
        The flow continues by the model-based RL algorithm of choice
    end for
end while

4 Experimental Domains

This section presents the two domains used for the experimental evaluation of COMBREL. Both domains have continuous state spaces and discrete action spaces.

4.1 Mountain Car

The Mountain Car (Singh and Sutton, 1996) is a standard 2D task (MC2D) that requires an under-powered car to drive up a hill. The state is described by two continuous variables, the position $x \in [-1.2, 0.6]$, and velocity $v_x \in [-0.07, 0.07]$. The actions are {Neutral, Left, Right}, and the goal state is $x >= 0.5$, with a reward of -1 on each time step.

The 4D mountain car task (MC4D) extends the 2D task by adding an extra spatial dimension (Taylor et al., 2008a). The state is composed by four continuous variables. The coordinates x and y are in $[-1.2, 0.6]$, and the velocities v_x and v_y are in $[-0.07, 0.07]$. The actions are {Neutral, West, East, South, North}, the goal is $x, y >= 0.5$ and the reward is -1 on each time step.

The MC software[2] used in this work is based on version 3.0 of the RL-Glue library[3] (Tanner and White, 2010).

4.2 iCub Humanoid Robot and the Ball Striking Task

The iCub is an open source humanoid robot platform (see Figure 1) designed for research on developmental robotics (Cangelosi and Schlesinger, 2012; Sandini et al., 2007). This work uses a computer simulation model of the iCub robot [4] (Tikhanoff et al., 2008), RL-Glue for the RL controllers and a novel implementation of a YARP wrapper [5], which allows a standard RL-Glue agent to interact with iCub.

The "ball striking task" is a novel transfer learning task for RL in which iCub must learn to strike a ball located in a fixed starting position on a table by controlling different parts of its body. The iCub is provided with a reward of -1 on every time step and with +100 for every successful hit of the ball. Any time the iCub successfully moves the ball, even slightly, the agent is rewarded and the episode ends. Also, the robot is considered "blind" since no optical input is used and the robot must search for the ball in the space surrounding it. An episode ends when the ball is moved (after a successful strike) or after 400 steps and for the next episode the positions of the robot and the ball are reset to the same start positions as before.

[2] Available at `http://library.rl-community.org/`

[3] Available at `http://glue.rl-community.org/`

[4] Available at `http://eris.liralab.it/wiki/Main_Page`

[5] Available at `http://mlkd.csd.auth.gr/COMBREL.html`

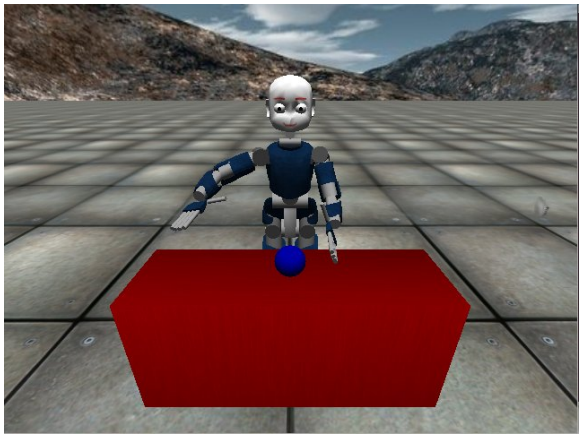

Figure 1. The ball striking task.

In the source task, the robot must learn a policy for controlling two Degrees of Freedom (DoF) of the robot's torso, to strike the ball, with the minimum number of joint movements (see Figure 1). All other iCub joints are fixed and the robot is expected to strike the ball with the tip of its right hand fingers (the most extended part of is body). The state space for this task is defined by two state variables, the positions of two of the torso joints as measured by the robot's encoders. The action space is a 2-dimensional vector of the form, $\langle$*joint,direction*$\rangle$ representing a command to move *joint* $i, i \in \{1, 2\}$ in a positive or negative *direction*, $d \in \{-1, 1\}$. The extent of each movement is fixed and equals the velocity of the corresponding joint. In this case this unitless amount of velocity is fixed to 10.

In the target task, the robot must now learn a policy for controlling four DoF of the robot's right arm instead of controlling its torso. The state space S is comprised of four state variables, which correspond to the positions of four right arm joints and the action space is once again a 2-dimensional vector of the form, $\langle$*joint,direction*$\rangle$ representing a command to move *joint* i where $i \in \{1, 2, 3, 4\}$, to a positive or negative *direction*, $d \in \{-1, 1\}$.

Finally, we note that transfer between the two tasks is non-trivial because each task uses different joint groups (the shoulder and the torso, respectively) to hit the ball. These joint groups have intrinsically different dynamics and characteristics.

The knowledge we expect to be transferred has to do with the orientation of the movements — the ball is in the same location in front of the robot in both tasks and the robot must lean or lower some body parts and then turn right to hit it.

Table 1. The parameters used in the MC experiments.

Parameter		Notation	Value
Fitted R-max			
	model breadth	b	0.4
	minimum fraction	-	0.1
	minimum weight	w_{min}	0.01
COMBREL			
	source task instances	$\|S\|$	1000
	# of nearest target task instances used	k	3

5 Experiments and Results

This section presents positive transfer learning results using COMBREL in two domains.

5.1 Mountain Car

To test the efficiency of our transfer method in MC we first use Q-learning to learn the Mountain Car 2D task while recording instances from the agent's experiences. These instances, recorded as tuples: $\langle s, a, r, s' \rangle$, formed a source task dataset containing 1000 instances which adequately covers the state-space.

In the target task, Fitted R-Max was used as the underlying model-based algorithm for COMBREL (see Table 1 for a summary of the parameters used in MC). The experiments compare several algorithms: Fitted R-Max (labelled No Transfer), TIMBREL, which uses a single-mapping, COMBREL without relevance, and COMBREL with relevance. COMBREL (both with and without relevance sorting) was provided an (exhaustive) set of 1960 inter-task mappings [6].

The first experiment (Figure 2) tests how well COMBREL can handle a large number of mappings. This is a particularly useful case when there is no expert or domain intuition to select the correct inter-task mapping. The experiment also measures whether there is a benefit to not using a fixed mapping for the entire state space.

Figure 2 shows results from the first experiment, where COMBREL uses an exhaustive set of possible inter-task mappings. The results show the significant benefit of using a multiple-mappings method which is also accompanied with a selection mechanism and also demonstrates the performance boost of the proposed algorithm over the other approaches. The performance difference of COMBREL

[6] There are $4!$ different possible state mappings and 90 action mappings, assuming that every state variable and every action is mapped to at least one state variable and action in the source task.

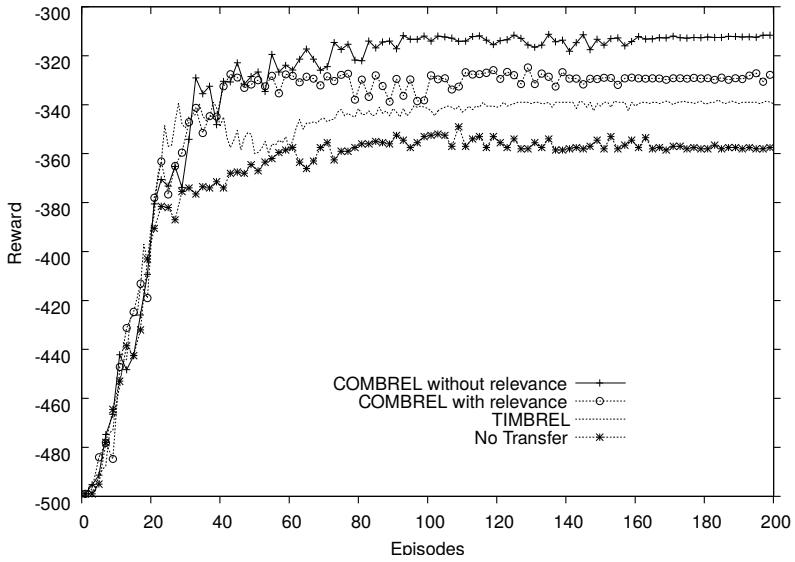

Figure 2. Learning with Transfer in 4D Mountain Car using 1000 source task instances. The curves are averaged over 15 runs.

compared to the other algorithms was statistically significant at a 95% level. COMBREL performs significantly better than TIMBREL, showing that a single and fixed mapping methodology performs worse than a multiple mappings one, even if that single mapping is the intuitive one. Interestingly, COMBREL with Relevance demonstrated lower performance, as discussed later in this section.

The second experiment (see Figure 3) tests the performance of using relevance in a single-mapping TL algorithm (TIMBREL) without an explicit mappings selection mechanism, i.e., using relevance without compliance or any compliance-based mapping selection process. Without a mappings selection process one can generate a single artificial pool of source task instances consisting from m translations of each original source task instance, where m is the number of available mappings. In this case the agent does not select a virtual source task (mapping) to transfer from, but transfers from a common pool of source task instances generated from all the mappings which can also be seen as a single source task dataset. Figure 3 shows the results of using TIMBREL with relevance and without any explicit mappings selection mechanism, compared to COMBREL with and without relevance (using COMBREL curves from Figure 2). The results show poor performance compared to COMBREL with or without relevance, showing that a compliance-based task selection mechanism does indeed improve performance.

Using relevance as the transfer criteria in our experimental settings was harmful

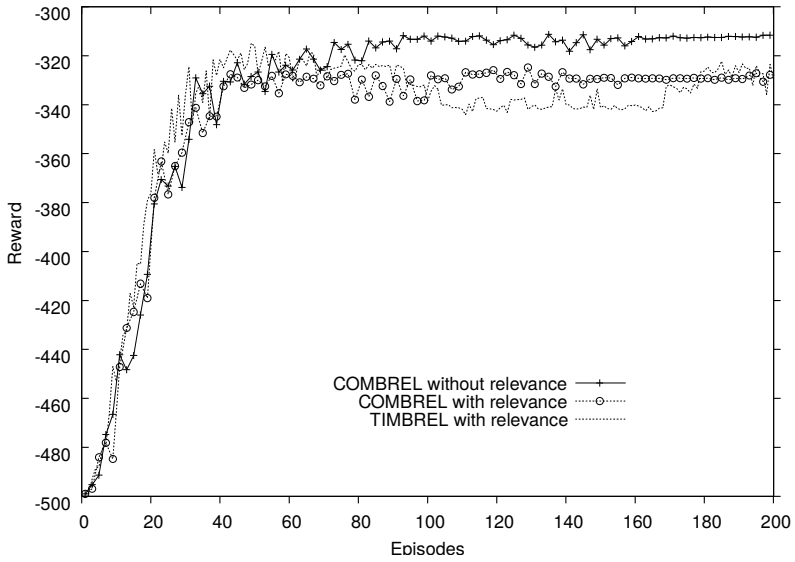

Figure 3. Using the single-mapping TIMBREL with relevance and an exhaustive set of mappings in 4D Mountain Car. All the algorithms use the same source task dataset of 1000 instances.

to the performance of the agent. In both experiments, the COMBREL agent that used relevance was frequently selecting mappings that were substantially different from each other (most of their state/action variables were mapped differently). Relevance was originally defined within a multi-task single-mapping setting where state and action spaces are the same in both the source and target tasks. As mentioned earlier (Section 3.1), relevance implements optimism in the face of uncertainty. However, this is a poor heuristic when transferring from virtual source tasks (mappings) that are substantially different from the target task. Particularly when using a large set of mappings, using COMBREL without relevance is better than using COMBREL with relevance.

An analysis of the mappings chosen by COMBREL (without relevance) brought to light a number of interesting characteristics. The COMBREL agent used on average, 4 state-action mappings per episode while in a total trial of 1000 episodes it used $\sim$ 20 state-action mappings from the 1960 available. In the total number of runs, COMBREL constantly finds the correct (e.g., intuitive) state-mapping but frequently changes the action-mapping based on its position and velocity (see Table 2 for the most frequent selections of action mappings). The reason for this difference is that interchanging state variables in a state mapping (e.g., incorrectly mapping a source task position variable to a target task velocity variable) results

Table 2. The 9 most frequently selected action mappings over all states, from a pool of 90 different action-mappings. The mappings show which source task action (left column, with letters representing actions **L**eft, **N**eutral, **R**ight) is mapped to which MC4D action (the columns representing MC4D actions: Neutral, West, East, South, North). The three most common mappings per MC4D action (columns) are shown in bold. The most intuitive mapping is indicated with an arrow.

		MC4D Action					
		Nt	W	E	S	No	All
Mapping	L L R N R	**43.7**	**39.9**	**40.6**	**38**	**21.7**	36.8
	L N R N R	**13.6**	**16.5**	**13.9**	**22.7**	15.5	16.3
	R N L R L	1.3	8.3	**10.4**	**14.3**	**15.7**	10.1
	R L L R N	**25.1**	3.4	8.5	4	1	8.3
	L N L R R	1.5	3.9	2.2	3.3	**22.1**	6.6
$\rightarrow$	N L R L R	2.6	3.2	9.3	1.5	0	3.7
	R N R L L	4.4	4.4	1.4	0.4	4.8	3
	L N R R N	0.4	0	2.2	9.3	0	2.4
	L N R R L	0	**10.3**	0.2	0.9	0	2
	Others	7.4	10.1	11.3	5.6	19.2	10.8
	Total	100	100	100	100	100	100

in large errors and consequently to mappings with very low compliance. However, the action mappings selection process handles much smaller compliance value differences than those of the state mappings, since different actions can have very similar effects when executed in different state space regions.

Using a Kruskal-Wallis test (non-parametric ANOVA) we found a statistically significant relation (at $p < 0.05$) of the position and speed state variables to the mappings selected, showing the strong dependence of the chosen mappings to the agent's state. Also, a chi-square test revealed a statistically significant relation (at $p < 0.05$) between actions and the mappings selected by COMBREL. This shows us that in a specific state, the agent uses different mappings when it considers a different target task action a. This happens because COMBREL computes the compliance of each virtual source task only to the target task instances that apply action a, thus resulting to different compliance values of each mapping for each target task action.

Table 2 shows the most frequent *action-mapping* choices made, per target task action. In the first column we can see the nine most frequently used action-mappings. As an example, the mapping described as "L L R N R" is a mapping that maps the action "Left" of MC2D to the action "Neutral" in MC4D, the action "Left" in MC2D to the action West in MC4D, the action "Right" in MC2D to the

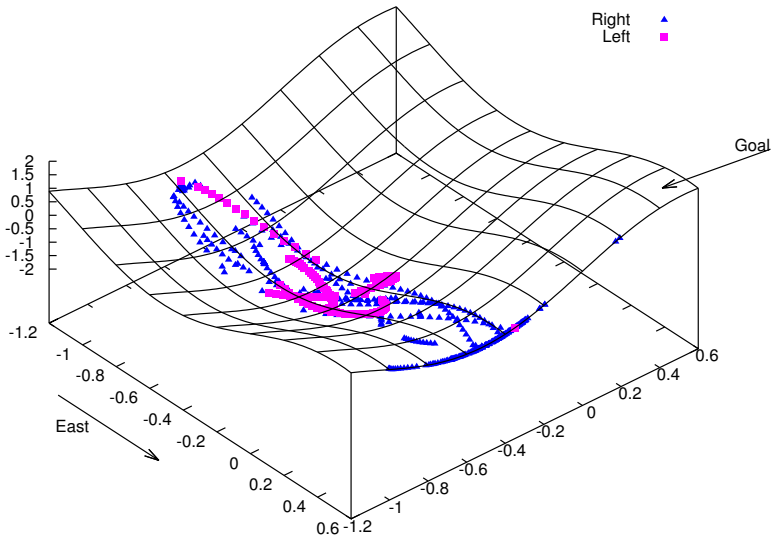

Figure 4. Action mappings selection for the target action East. Mappings that map action East to the same source action are represented with the same pattern.

action "East, the action "Neutral" in MC2D to the action "South" in MC4D and finally the action "Right" in MC2D to the action "North" in MC4D. Different target task actions cause different action mappings to be used, although the first mapping is most frequently used. The mapping considered as the most intuitive earlier in (Taylor et al., 2008a) is indicated with an arrow. This action mapping is considered to be the most intuitive one since it maps the actions in MC4D that move the car uphill and towards the goal state to the action "Right" in the MC2D, which also moves the car uphill and towards the goal state. Consequently this action mapping maps the "Left" action to the actions "South" and "West" which are all actions that move the car away from its goal state.

In order to further understand the meaning of these mapping selections, consider the area of the MC state space $(0.00827 \pm 0.0199, 0.0198 \pm 0.0279)$. This area, which is on the slope leading to the goal state, uses exclusively the first action mapping from Table 2 (which is also the most common choice). This mapping is different from the intuitive one in that it maps the "Left" (and not the "Neutral") action of MC2D to the "Neutral" action of the MC4D and maps the "Neutral" action (instead of "Left") to the "South" action of MC4D. Indeed, the effect of the MC2D action "Left" (away from the goal) is very similar to the "Neutral" action in MC4D when on the slope. Similarly for the effect of the MC2D action "Neutral", which in

Table 3. The parameters used in the iCub experiments.

Parameter		Notation	Value
Fitted R-max			
	model breadth	b	0.1
	minimum fraction	-	0.01
	minimum weight	w_{min}	0.01
COMBREL			
	source task instances	$\|S\|$	1000
	# of nearest target task instances used	k	3

this particular area, resembles the effect of the "South" MC4D action (away from the goal).

In another example, Figure 4 shows the final mapping effect for action East in several state space points. For clarity, all the action mappings that result in mapping a specific 2D action to the action East are represented with the same pattern. We can see that the mappings that translate action East to action Left are mostly used in the valley and the west portion of the state space. At these points COMBREL finds that the final transition effect of executing action East in that specific state is more similar to executing the action left at an analogous point in Mountain Car 2D. This can be correct when the car has gained enough speed towards the west direction and its inertia causes an opposite transition effect to that expected when we have zero speed. The above examples shows that since COMBREL is an instance-based method, it handles mappings based on the actual similarity of the transition effects and not on any expected effect.

5.2 Ball Striking

In the ball striking task we tested COMBREL without relevance against two versions of TIMBREL, one using a random mapping and one using the the single most frequently selected mapping by COMBREL. Moreover, we tested a No-Transfer case using a standard Fitted R-Max agent. The first version of TIMBREL was set to randomly select a mapping *in every* transfer attempt. This baseline case let us evaluate the importance of actually finding a correct mapping using a method such as COMBREL. The second version of TIMBREL uses COMBREL's single most frequently selected mapping in order to evaluate the importance of not just using a single mapping, even if this single mapping is considered as the most appropriate for most of the target task's state space (as previously calculated by COMBREL). Finally, we ran a standard Fited R-max agent for the no-transfer case (see Table 3 for a summary of the parameters used by all the algorithms in iCub). .

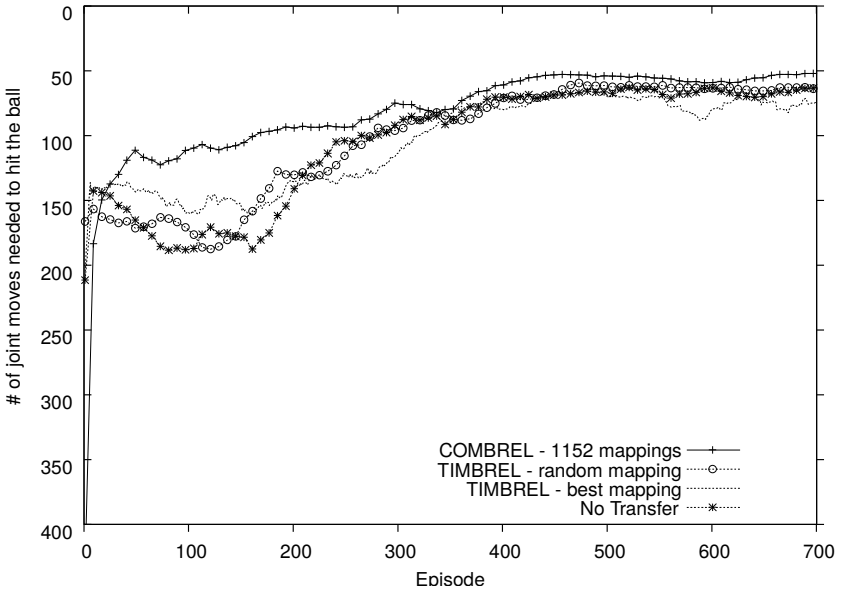

Figure 5. Learning with Transfer in the iCub ball striking task using 1000 source task instances. The curves are averaged over 10 runs.

To learn the source task, we ran for 1000 episodes a standard Fitted R-max agent to control iCub and learn a policy for hitting the ball using only the two torso joints[7]. While learning we recorded transitions (experiences). A dataset of 1000 transitions was formed, containing non-terminal and terminal transitions. Fitted R-Max converged to a policy of approximately 25 joint commands to hit the ball in 1000 episodes.

In the target task, COMBREL used a non-exhaustive set of 1152 mappings generated by 24 state mappings and 48 action mappings. This non-exhaustive set of mappings was chosen in order to include only the mappings that map every state and action variable of the source task to at least one state and action variable of the target task. An exhaustive set of mappings would also include mappings that ignore one of the two joint groups of the source task. Next, TIMBREL, as a single mapping transfer algorithm, had to be given a predefined mapping. We ran two different versions of TIMBREL — one used a random mapping and one used the mapping most frequently selected by COMBREL. The last case is useful for these comparisons since the experiment's goal is to illustrate not only the importance of automatically discovering the single most appropriate mapping but also the im-

[7] A video for solving the source task is available in: `http://mlkd.csd.auth.gr/COMBREL.html`

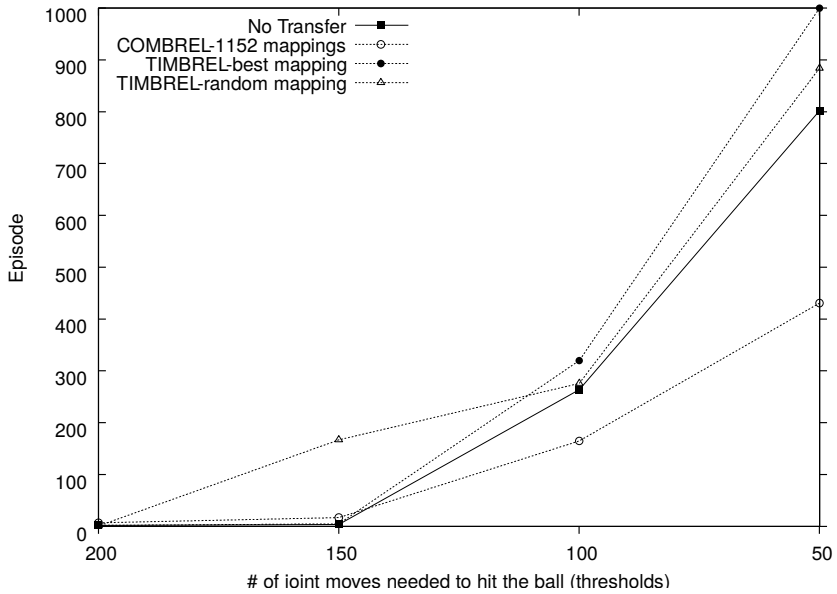

Figure 6. Time to threshold graph for the iCub ball striking task.

portance of changing mappings in complex domains and not using a single one only. Both COMBREL and TIMBREL used the same dataset of 1000 experiences recorded in the source task and Fitted R-Max is the base RL algorithm for both of them.

Figure 5 shows the results from using COMBREL on the ball striking target task. In the highly stochastic and complex environment of iCub, COMBREL assists learning by successfully selecting the appropriate mappings and transferring instances from the ball striking source task, demonstrating significantly better performance than the no-transfer case[8]. Moreover, in episodes 50 - 300, COMBREL significantly outperforms (at a 95% level) both versions of TIMBREL, by using multiple mappings and automatically selecting the most appropriate one depending on the state.

Figure 6 present the results using the time-to-threshold TL metric. The x-axis shows various thresholds representing the steps needed by the agent to find the goal (successful hit of the ball) and the y-axis represents the corresponding training time needed to achieve these goals. Each curve again averages the results from 10 runs. In this graph we can clearly see how COMBREL speeds up learning compared

[8]A video for solving the target task is available in: `http://mlkd.csd.auth.gr/COMBREL.html`

to the other methodologies since it is able to strike the ball with 50 and 100 joint moves in half the training time needed by the other algorithms to achieve these two thresholds.

Although TIMBREL is also tested using COMBREL's most frequently selected mapping (i.e., the best mapping), its performance is poor and after 600 episodes it is worse than the no-transfer case. This negative transfer happens because it uses a single-mapping for transferring between two complex tasks with non-trivial and non-uniform similarities. Interestingly, when TIMBREL uses random mappings it performs better in the long run than using the best mapping. In episodes 0 to 200, TIMBREL uses random mappings and transfers mostly irrelevant source task instances because there are as yet no close target task instances to the approximated states, resulting in a very poor performance. However, after 200 episodes its performance gradually increases because sufficient target task instances have been collected and the source task instances are no longer considered close enough to be transferred. Consequently, after 200 episodes no more transfer occurs in this TIMBREL experiment, resulting in similar performance to the no transfer case.

Using TIMBREL with the best mapping in this domain performs poorly. A better state mapping usually allows transferring samples close to the target task instance, but the single action mapping is a poor choice in some areas of the state space. This discussion leads us to one important difference between COMBREL and TIMBREL: COMBREL transfers instances based not only on the similarity of the states but also on the similarity of the action outcomes whereas TIMBREL "trusts" its single action mapping and transfers instances solely based on the state similarity (closeness), not evaluating the actual similarity of the actions outcome.

The most frequent mapping selected by COMBREL is the combination of the state mapping #8 and the action mapping #21 (see Table 4 and Figure 7). This combination of mappings, #8 and #21 (a state-action mapping) maps 1) the torso yaw joint to the shoulder yaw, 2) the shoulder roll joint to the elbow pitch joint, and 3) the torso pitch joint to the shoulder pitch joint. This state-action mapping reflects the authors' intuition of the orientations so that in both tasks a successful policy is a series of downward pitch movements and left yaw movements. Finally, we should note that just like in the mountain car experiments presented earlier in this text, COMBREL constantly selects only one state mapping (#8) whereas depending on the state, it changes action mappings with the most frequent choice being the mapping #21.

Figure 7. iCub joints controlled in the source task (green arrows) and the target task (red arrows), where each joint corresponds to a state variable. The arrow head shows the positive direction of the joint (+) and also represents the positive action on that joint (e.g., S1+.)

	Target task action								
Action Mapping	T1-	T1+	T2-	T2+	T3-	T3+	T4-	T4+	Percentage
21	S1+	S1-	S2+	S2-	S2-	S2+	S1+	S1-	79.5%
23	S1+	S1-	S2+	S2-	S2+	S2-	S1+	S1-	11.1%
20	S1+	S1-	S2+	S2-	S1-	S1+	S2+	S2-	6.8%
2	S1+	S1-	S1+	S1-	S2-	S2+	S2+	S2-	2.6%

Table 4. Action mappings selected by COMBREL in the iCub ball striking task and percentage of times each one was selected. See Figure 7 for the meaning of the action's encoding.

Table 5. Average running times of the compared algorithms in MC 3D for 200 episodes.

Algorithm	**Avg. running time (m)**
Fitted R-max (no transfer)	24.0
TIMBREL (1 predefined mapping)	28.1
COMBREL (1960 mappings)	40.9

5.3 Computational Complexity

When considering n virtual source tasks with m samples each, and t samples are collected from the target task, the time complexity of COMBREL in the first episode, where compliances are calculated analytically, is $O(nmt)$. In the rest of the episodes the time complexity is $O(n|S|t)$ where n is the number of virtual source tasks, $|S|$ is the number of state variables and t the number of samples collected from the target task (kNN complexity multiplied by the number of virtual source tasks) .

Table 5 reports the recorded running times for 200 episodes of our implementation of COMBREL in Mountain Car. The experiment ran on an Intel Core 2 Duo 2.8 GHz PC with 16GB RAM. COMBREL needs around 40.9 minutes to complete 200 episodes. 12.1 minutes are used just for the first episode because compliances are analytically calculated. The remaining 199 episodes used only 28.8 minutes. We can conclude that COMBREL using an exhaustive set of mappings (1195 mappings) needs significantly more time in the first episode, where compliances are analytically calculated but for the rest of the episodes where COMBREL uses kNN regression to estimate compliances, the execution of the algorithm is not significantly slower compared to the other two algorithms.

6 Related Work

In the multi-task transfer learning setting an algorithm that implements the notions of compliance and relevance has been proposed (Lazaric et al., 2008). A detailed description of this work can be found in Section 2. Our proposed method differs significantly since it is used in a distinct, different setting, that of a multiple mappings single-task transfer learning problem and proposes also a novel algorithm suitable for that setting. Moreover, our method allows for different state and action variables between tasks.

TIMBREL, an algorithm that implements single-mapping transfer learning for model-based RL agents has been proposed (Taylor et al., 2008a). A description of this work can be found in Section 2 of this paper. Although it demonstrates

promising results and a significant performance gain, it uses only one hand-coded mapping between the source and the target task. Our proposed model-based multiple mappings method is based on TIMBREL but significantly extends it with the use of a multiple mappings mechanism. The mechanism is able to autonomously select a mapping while learning and moreover, with the adoption of compliance and relevance, it uses a sophisticated instance sorting criteria instead of the simpler distance criteria.

In other work, (Taylor et al., 2008b) propose an autonomous mapping selection method (MASTER) which is able to select a mapping based on the similarity between transitions in the source and target task. MASTER learns a model of the action effects in the source task. In the target task, it first selects actions randomly, sampling target task transitions. It then compares this transitions with queries on the source task model and calculates the error (difference). It selects the best mapping to minimize this error, but suffers from an exponential increase in complexity in the number of state variables and actions. Our model-based method is also based on the similarity of the action effects between two tasks but it does an off-line calculation of the error thus not spending the agent's time with random environment interaction. Moreover, COMBREL uses compliance and relevance to compute task similarity. Lastly, our proposed method requires no explicit model learning in the source task, as it only transfers instances from it.

A method considering each possible inter-state mapping as an expert has been proposed (Talvitie and Singh, 2007). With the use of a multi-armed bandit algorithm the agent decides which mapping to use. This method shows significant improvement but its performance is decreased in the long run as it continues to explore always taking "advice" from low return experts.

Another promising method that has been proposed (Sorg and Singh, 2009) is capable of learning a continuously parametrized function representing a stochastic mapping between the state spaces of two MDPs using "soft homomorphisms. However, this work does not handle action mappings and requires the explicit learning of a model in the source task and an on-line learning process of the mapping function, in the target task. Our proposed method requires only a set of source task instances and implements an off-line computation of task similarity in the target task. The reader is directed to read more about the various transfer learning methodologies in more comprehensive treatments (Taylor and Stone, 2009).

There are other TL methods which are focused on *learning* a mapping and not on computationally discovering it, like COMBREL does. COMBREL is not inferring a mapping that is not explicitly present in the data (learning) but instead searches for it by evaluating pre-defined mappings from a possibly exhaustive set of mappings. In an example of a work that aims to learn a mapping between arbitrary tasks (Ammar et al., 2012), the authors use a data-driven method to discover

Table 6. Summary of the results obtained from this work.

Result	**Experiment**	**Stat. sig.**
COMBREL performs better than No Transfer	Fig. 3, 6 & 7	Yes
Multiple Mappings (COMBREL) performs better than single mapping (TIMBREL)	Fig. 3, 6 & 7	Yes
COMBREL w/o relevance performs better than COMBREL with relevance	Fig. 3, 4	Yes
COMBREL with relevance performs better than TIMBREL with relevance	Fig. 4	Yes (episodes 100-170)
When a single mapping is not consistently the best, using a single mapping performs poorly even if it is the most intuitive one (a random one can perform better by actually disabling transfer)	Fig. 6 & 7	Yes

a mapping between two tasks in a new dimensional space using sparse coding. However, the authors do not consider allowing multiple mappings, nor do they consider what information to transfer from the source task. Moreover, this work can not be applied between domains with different reward functions since it does not implement a reward similarity measure such as that of reward compliance in COMBREL.

Learning complete inter-task mappings with reward similarity, forms a learning problem of high complexity which could be avoided provided with measures such as that of compliance and relevance that take into account the full similarity of the MDP's and explicitly calculate it.

Finally, (Ammar et al., 2013) introduce a method for automatically determining an inter-task mapping for pairs of tasks using a restricted Boltzmann Machine. However, their method does not consider how related the tasks are — once a mapping is learned, all source task data is used, regardless of the usefulness of the data in the target task. Furthermore, this mapping is static and cannot be adjusted to different parts of the state space.

7 Conclusions and Future Work

This article examined the benefits of selecting and using multiple mappings in transfer learning by extending the ideas of compliance and relevance so rather than being used to select instances from multiple source tasks, they can be used to select instances from multiple "virtual" source tasks (mappings).

Both MC and iCub experiments showed that COMBREL is able to achieve

better performance than a single-mapping methodology even if that simple mapping methodology uses the best or the most frequently selected by COMBREL mapping (see Table 6 for a summary of the results obtained in this article). This shows the importance of not applying a single mapping to an entire TL task — dynamically changing the selected mapping according to the agent's state improves performance. In both domains, the selection of the best action mappings varied across the state space, but our algorithm converged to a single state mapping.

Moreover, by experimenting on the iCub domain this work demonstrated the usefulness of multiple mappings when transferring in domains where there is no intuitive mapping or one that is consistently the best, removing the requirement for a human to select an inter-task mapping.

Finally, a key qualitative result of this work is that transforming a multiple task, single mapping problem to a single task, multiple mapping one is feasible, allowing the exchange of previously unrelated results, between the two contexts.

A limitation of the proposed approach is that it applies to the specific TL setting (that of transferring source task instances) but future extensions could apply the ideas of compliance and relevance to other transferable forms of knowledge such as task models, value functions, etcetera. In tasks where the the first episode may not be representative, a more computationally efficient version of our algorithms should be able to use more target task transitions, collected through more episodes and not only those from the first episode(s). Future work also includes the development of a generalized form of the relevance metric which will not favor distant transitions in order to correctly evaluate transfer eligibility from very dissimilar virtual source tasks (mappings). Moreover, future work should also experiment on incorporating multiple source tasks together with multiple virtual source tasks (i.e., mappings). Lastly, we would like to test our methods on the physical iCub platform.

Acknowledgements

This work was supported in part by NSF IIS-1149917. We would also like to deeply thank Prof. Angelo Cangelosi and post-doctorate researcher Alessandro Di Nuovo for providing us all the necessary resources for experimenting on the iCub robotic platform.

References

Haitham Bou Ammar, Karl Tuyls, Matthew E. Taylor, Kurt Driessen, and Gerhard Weiss. Reinforcement learning transfer via sparse coding. In *International Con-*

ference on Autonomous Agents and Multiagent Systems (AAMAS), June 2012.

Haitham Bou Ammar, Decebal Constantin Mocanu, Matthew E. Taylor, Kurt Driessens, Karl Tuyls, and Gerhard Weiss. Automatically mapped transfer between reinforcement learning tasks via three-way restricted boltzmann machines. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, September 2013. 25

Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3: 213–231, March 2003. ISSN 1532-4435. doi: 10.1162/153244303765208377. URL `http://dx.doi.org/10.1162/153244303765208377`.

Angelo Cangelosi and Matthew Schlesinger, editors. *Developmental robotics: From babies to robots*. MIT Press, 2012.

F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *5th international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006.

N. Jong and P. Stone. Model-based exploration in continuous state spaces. *Abstraction, Reformulation, and Approximation*, pages 258–272, 2007.

A. Lazaric, M. Restelli, and A. Bonarini. Transfer of samples in batch reinforcement learning. In *Proc. of the 25th ICML*, pages 544–551, 2008.

A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX*, pages 363–372, 2006.

Giulio Sandini, Giorgio Metta, and David Vernon. The icub cognitive humanoid robot: an open-system research platform for enactive cognition. In Max Lungarella, Rolf Pfeifer, Fumiya Iida, and Josh Bongard, editors, *50 years of artificial intelligence*, pages 358–369. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3-540-77295-2, 978-3-540-77295-8.

Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1-3):123–158, 1996.

Jonathan Sorg and Satinder Singh. Transfer via soft homomorphisms. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 741–748, 2009. ISBN 978-0-9817381-7-8.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, An Introduction*. MIT Press, 1998.

Erik Talvitie and Satinder Singh. An experts algorithm for transfer learning. In *20th IJCAI*, pages 1065–1070, 2007.

Brian Tanner and Adam White. Rl-glue: Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research*, 10:2133–2136, 2010.

M. E. Taylor, N. K. Jong, and P. Stone. Transferring instances for model-based reinforcement learning. In *Proc. of ECML*, pages 488–505, 2008a.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8:2125–2167, 2007.

Matthew E. Taylor, Gregory Kuhlmann, and Peter Stone. Autonomous transfer for reinforcement learning. In *7th AAMAS*, pages 283–290, 2008b.

V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, and F. Nori. An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator. In *PerMIS '08: Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 57–61, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-293-1.

L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *16th European Conference on Machine Learning*, 2005.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 1015–1022, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273624. URL `http://doi.acm.org/10.1145/1273496.1273624`.