Reinforcement Learning Control With Knowledge Shaping

Xiang Gao, Jennie Si, *Fellow, IEEE*, and He Huang, *Fellow, IEEE*

***Abstract*— We aim at creating a transfer reinforcement learning framework that allows the design of learning controllers to leverage prior knowledge extracted from previously learned tasks and previous data to improve the learning performance of new tasks. Toward this goal, we formalize knowledge transfer by expressing knowledge in the value function in our problem construct, which is referred to as reinforcement learning with knowledge shaping (RL-KS). Unlike most transfer learning studies that are empirical in nature, our results include not only simulation verifications but also an analysis of algorithm convergence and solution optimality. Also different from the well-established potential-based reward shaping methods which are built on proofs of policy invariance, our RL-KS approach allows us to advance toward a new theoretical result on positive knowledge transfer. Furthermore, our contributions include two principled ways that cover a range of realization schemes to represent prior knowledge in RL-KS. We provide extensive and systematic evaluations of the proposed RL-KS method. The evaluation environments not only include classical RL benchmark problems but also include a challenging task of real-time control of a robotic lower limb with a human user in the loop.**

***Index Terms*— Reinforcement learning (RL), reward shaping, transfer learning, value function.**

I. Introduction

TRANSFER learning in the context of reinforcement learning is less studied than that in supervised or unsupervised learning [1], [2], [3], [4]. In this study, we investigate knowledge transfer in RL by virtual task. We consider a task to be described by the components of an RL problem, i.e., the state space, the action space, the system dynamics or the environment, and the rewards or costs. A successful knowledge transfer is expected to improve learning performance if knowledge from source tasks can be efficiently utilized in learning a new target task.

Prior knowledge can have a multitude of representations. A simple and intuitive idea is to directly reuse experience samples to reduce the number of samples needed to learn a new task [5]. Reusing previously learned value function [6], [7] and policy [8], [9] are among the most popular methods for knowledge representations in the context of a transfer. Cheng et al. [7] present a heuristic-guided RL (HuRL) that induces a much shorter-horizon subproblem that provably solves the original task. Nonetheless, the effectiveness of HuRL depends on the available heuristic and a bad heuristic will hinder learning. The method of successor features [8], [9] is promising as it can decouple environment dynamics from rewards, and further triggers policy transfer from a set of "base tasks" to a target task. However, these methods only work with certain forms of Markov decision processes (MDPs), and heuristics are still required to properly choose the "base tasks." Policy distillation [10], [11] transfers knowledge between tasks by summarizing multiple learned policies into a single one. As such, preparing multiple learned policies may take a long time and thus slow down the entire learning process.

Reward shaping is one of the more popular approaches for knowledge transfer. The original potential-based shaping and subsequent extensions of the idea [12], [13], [14], [15], [16] have been developed and mathematically shown to guarantee policy invariance under the shaped reward. This approach represents a large class of transfer methods that modulates the reward function. Later it was proven that potential-based shaping and Q-value initialization are equivalent [17]. In [18], a method that focuses on searching for policies that maximize the total reshaped reward over a finite planning horizon is proposed. Yet, it relies on a near-optimal cost-to-go oracle which is hard to obtain.

While different ideas for transfer in RL have been proposed and examined, almost all these results are empirical in nature with a handful of exceptions. An important theoretical result showed that the choice of knowledge representation, either by a reward or an initialization of the values, can significantly influence the learning performance in either direction, positively and negatively [19]. As such, how to choose a potential function is important to avoid an adverse transfer effect. These important issues have not been addressed from two important aspects. First is how to represent prior knowledge from source tasks and second, how to ensure a positive transfer.

This study addresses these important questions centered on a new, value function-based transfer scheme where prior knowledge can be represented in many ways centered around two principled ideas of either by a regression model via supervised learning, or directly in the form of a state-action

Manuscript received 11 February 2022; revised 3 September 2022 and 30 October 2022; accepted 2 February 2023. This work was supported by NSF under Grant 1563454, Grant 1563921, Grant 1808752, Grant 1808898, Grant 2211739, and Grant 2211740. *(Corresponding author: Jennie Si.)*

Xiang Gao and Jennie Si are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: si@asu.edu).

He Huang is with the Joint Department of Biomedical Engineering, North Carolina State University, Raleigh, NC 27695 USA, and also with the Joint Department of Biomedical Engineering, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599 USA (e-mail: hhuang11@ncsu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3243631.

Digital Object Identifier 10.1109/TNNLS.2023.3243631

value function. Such knowledge representation frameworks offer great flexibility. For example, learning from demonstration and imitation learning or specifically inverse RL both can be embedded into our proposed framework. The former can be a consequence of direct interaction with the source task with accessible reward signals, and the later can be a means of recovering the reward function from expert demonstrations in the source task. In this work, we also make a step forward by defining a new similarity measure between the source task and the target task. We show that the proposed reinforcement learning with knowledge shaping (RL-KS) will result in a positive transfer if both tasks are sufficiently similar. In summary, a key advantage of the proposed RL-KS includes the following. It allows the synthesis of an RL controller to leverage accumulated knowledge from previous experience of similar tasks in a variety of flexible ways. As such, it not only addresses the data shortage problem in some critical applications but also improves data and time efficiency in learning new tasks.

The main contributions of this article are summarized below.

1) We have not only formalized a value function-based transfer learning for RL control design framework, but also provided multiple and flexible ways to represent prior knowledge. Additionally, our approach ensures positive knowledge transfer. Strong results of this nature are not available in the current literature.
2) We have provided a theoretical analysis of RL-KS with performance guarantees such as convergence of learning, solution optimality, and positive transfer of prior knowledge.
3) We have performed extensive and systematic evaluation studies to demonstrate the effectiveness of RL-KS.

II. Background and Related Work

This study concerns how to represent prior knowledge and use it to augment reward/cost or value functions, and more importantly, how to ensure prior knowledge positively impacts on learning new tasks. As such, our RL-KS may be viewed as a reward-shaping technique. As previously proved that reward shaping is equivalent to value function initialization [17], our approach may also be considered as a technique of using prior knowledge to properly initialize the value function [20].

Reward shaping leverages prior knowledge from source tasks to shape the reward function in a target task aiming for improved learning. Potential-based reward shaping is a classical approach where the potential function R_f is used to provide auxiliary rewards to the target task reward R' based on prior knowledge from the source task reward R, i.e., $R' = R + R_f$. As will be shown, we introduce a new value-shaping framework and because of this, we will be able to provide performance guarantees and also, constructive approaches with the flexibility to represent prior knowledge.

In the original reward shaping formulation, the goal is to preserve the optimal policy with and without the auxiliary award, or in short, to achieve policy invariance. The potential-based shaping function can be a function of the state, or a state-action function for on-policy learning [13]. Another extension of the potential-based shaping [15] was to generalize the potential function so that the extra state-action value function can vary over time. This class of approaches has been shown to preserve policy, but it does not address the important question of if an auxiliary award will positively or negatively impact learning.

Another concern is that the main focus of the above approaches was on the shaping function $F(x, u)$ as an extra term in the reward function. Few efforts have directly addressed the important design question of how prior knowledge can be built into the shaping function, and furthermore, what learning tasks can benefit from such shaping, and how to ensure a positive knowledge transfer.

The problem of RL design for real-time control applications has been discussed in the field of adaptive dynamic programming (ADP). General results regarding convergence properties of value iteration under the Q-value formulation have been developed, first for exact value function approximation [21], [22] and later for considering value function approximation errors [23], [24]. Our work considers transfer learning within an RL context, and thus, addresses a broader class of application problems.

Consider a nonlinear discrete-time system

$$x_{k+1} = F(x_k, u_k), \quad k = 0, 1, 2, \ldots \tag{1}$$

where k is the time index of system dynamics, $x_k \in \mathbb{R}^n$ is the state and $u_k \in \mathbb{R}^m$ is the action. The mathematical system dynamic model $F(x_k, u_k)$ is unknown.

We need the following definition and assumption to develop our results.

Definition 1 (Stabilizable System): A nonlinear dynamical system is said to be stabilizable on a compact set $\Omega \in \mathbb{R}^n$, if for all initial states $x_0 \in \Omega$, there exists a control sequence $u_0, u_1, \ldots, u_k \in \mathbb{R}^m$, $k = 0, 1, \ldots$, such that the state $x_k \to 0$ as $k \to \infty$.

Assumption 1: System (1) is controllable and stabilizable, and the function $F(x_k, u_k)$ is Lipschitz continuous for $\forall x_k, u_k$. The system state $x_k = 0$ is an equilibrium state of system (1) under the control $u_k = 0$, i.e., $F(0, 0) = 0$. The feedback control sequence u_k is determined from control policy π which is the actor neural network, and in the most general case is bounded by actuator saturation. Let $u_0 = \pi(x_0) = 0$ for $x_k = 0$. The stage cost function $R(x_k, u_k)$ is finite, continuous in x_k and u_k, and positive definite with $R(0, 0) = 0$.

We consider the following infinite horizon discounted ($0 < \gamma < 1$) value or cost-to-go function for a control law $u_k = \pi(x_k)$:

$$J_\pi(x_k, u_k) = R(x_k, u_k) + \sum_{j=1}^{\infty} \gamma^j R\left(x_{k+j}, \pi\left(x_{k+j}\right)\right). \tag{2}$$

III. Actor-Critic Reinforcement Learning for Control

According to Bellman's optimality principle, the optimal cost-to-go $J^*(x_k, u_k)$ satisfies the action dependent discrete-time (DT) Hamilton Jacobi Bellman (HJB) equation

$$J^*(x_k, u_k) = R(x_k, u_k) + \gamma \min_{u_{k+1}} J^*(x_{k+1}, u_{k+1}) \tag{3}$$

and the optimal control law π^* can be expressed as

$$\pi^*(x_k) = \arg\min_{u_k} J^*(x_k, u_k). \tag{4}$$

By substituting (4) into (3), the DT-HJB equation becomes

$$J^*(x_k, u_k) = R(x_k, u_k) + \gamma J^*\big(x_{k+1}, \pi^*(x_{k+1})\big) \tag{5}$$

where $J^*(x_k, u_k)$ is the value function corresponding to the optimal control policy $\pi^*(x_k)$. This equation reduces to the Riccati equation in an LQR setting, which can be efficiently solved. In the general nonlinear case, the HJB cannot be solved exactly. In the following, we consider solving the value function $J^*(x_k, u_k)$ of the HJB equation (5) by actor-critic algorithms.

Actor-critic methods solve the DT-HJB by learning to approximate both policy and value functions where *actor* refers to the learned policy and *critic* refers to the learned value. An actor-critic algorithm starts with an initial value, e.g., $J_0(x, u) = 0$ and an initial arbitrary policy ξ_0. Then for $i = 0, 1, 2, \ldots$ it iterates between

$$\xi_i(x_k) = \arg\min_{u_k} J_i(x_k, u_k) \tag{6}$$

and

$$J_{i+1}(x_k, u_k) = R(x_k, u_k) + \gamma J_i(x_{k+1}, \xi_i(x_{k+1})). \tag{7}$$

The actor-critic scheme, therefore, is an incremental optimization procedure that iterates between a sequence of action policies π_i by greedy updates. Note that i is the actor-critic iteration steps, whereas k is the time index of system dynamics. Combining (6) and (7), we have

$$J_{i+1}(x_k, u_k) = R(x_k, u_k) + \gamma \min_{u_{k+1}} J_i(x_{k+1}, u_{k+1}). \tag{8}$$

Note that, the actor-critic algorithm does not require an initially stabilizing gain. The sequence $J_i(x_k, u_k)$ is not a sequence of Lyapunov functions for the corresponding policies $\xi_i(x_k)$ which are, in turn, not necessarily stabilizing.

IV. Knowledge Shaping in Reinforcement Learning for Control

Given a source task $\mathcal{T}_S$ and a target task $\mathcal{T}_T$, our proposed RL-KS algorithm aims to improve the learning of an optimal policy π^* for $\mathcal{T}_T$ using knowledge obtained from $\mathcal{T}_S$, where $\mathcal{T}_S \neq \mathcal{T}_T$. Later, we will address how transferable knowledge can be formulated as a knowledge representation function Q'. Here we first discuss how prior knowledge represented as Q' is integrated into an actor-critic RL solution.

A. RL With Knowledge Shaping

We begin by defining the following external bounded cost signal $R_f(x_k, u_k)$ that is of the form as in potential-based reward shaping

$$R_f(x_k, u_k) = Q'(x_k, u_k) - \gamma Q'(x_{k+1}, u_{k+1}) \tag{9}$$

where Q' is a representation of prior knowledge which will be discussed in Section V.

Let

$$R'(x_k, u_k) = R(x_k, u_k) + R_f(x_k, u_k) \tag{10}$$

then the knowledge-shaped value function $Q_\pi(x_k, u_k)$ for policy π is given as

$$Q_\pi(x_k, u_k) = R'(x_k, u_k) + \sum_{j=1}^{\infty} \gamma^j R'\big(x_{k+j}, \pi\big(x_{k+j}\big)\big). \tag{11}$$

From (11) we have

$$Q_\pi(x_{k+1}, \pi(x_{k+1})) = R'(x_{k+1}, \pi(x_{k+1})) + \sum_{j=1}^{\infty} \gamma^j R'\big(x_{k+1+j}, \pi\big(x_{k+1+j}\big)\big). \tag{12}$$

Given (11) and (12) we have the following HJB equation for $Q_\pi(x_k, u_k)$ given a policy π:

$$Q_\pi(x_k, u_k) = R'(x_k, u_k) + \gamma Q_\pi(x_{k+1}, \pi(x_{k+1})). \tag{13}$$

We formulate a new knowledge shaping RL iterative procedure, similar to (6) and (7), starting from $Q_0(x_k, u_k) = J_0(x_k, u_k)$ and $\pi_0 = \xi_0$. For $i = 0, 1, 2, \ldots$, RL-KS iterates between

$$\pi_i(x_k) = \arg\min_{u_k} Q_i(x_k, u_k) \tag{14}$$

and

$$Q_{i+1}(x_k, u_k) = R'(x_k, u_k) + \gamma Q_i(x_{k+1}, \pi_i(x_{k+1})). \tag{15}$$

Combining (14) and (15), we have

$$Q_{i+1}(x_k, u_k) = R'(x_k, u_k) + \gamma \min_{u_{k+1}} Q_i(x_{k+1}, u_{k+1}). \tag{16}$$

Note that the procedure depicted in (6)–(8) is a special case of that in (14)–(16) when there is no knowledge transfer, i.e., $R_f = 0$.

B. Obtaining Knowledge Shaping Function From Source Tasks

To represent prior knowledge Q'_i for RL-KS in iteration i, we introduce two principled approaches below. In both cases, the mathematical models (1) for both $\mathcal{T}_s$ and $\mathcal{T}_t$ are unknown to the RL agent. We begin by introducing a sequence of knowledge weighting factors $\{\alpha_i\}$ that satisfy $0 < \alpha_i < 1$, and $\alpha_i \to 0$ as $i \to \infty$. Intuitively, such weighting factors allow learning to immediately benefit from prior knowledge and then attend to the target task with the transferred knowledge.

1) Directly Learned Value From Source Tasks: From the previous section, we can directly transcribe previously learned value functions in a new task.

1) If the value function (Q-table) is available, then a direct transfer is applied

$$Q'_i(x_k, u_k) = \alpha_i Q_s(x_k, u_k) \tag{17}$$

where Q_s is the Q-table for the source task.

2) If the value function (Q-table) is unknown, then it can be estimated from expectations over training trajectories. For example, one can perform a Monte-Carlo simulation as follows: first, generate trajectories using a random

exploratory policy or a pretrained policy. Let this policy be μ, then for each state and action pair (x_k, u_k) in the Q-table, we use an N-step accumulated stage cost from the source task R_s to represent knowledge from the source task, that is,

$$Q'_i(x_k, u_k) = \alpha_i \left[R_s(x_k, u_k) + \sum_{j=1}^{N-1} \gamma^j R_s\left(x_{j+1}, \mu\left(x_{j+1}\right)\right) \right]. \tag{18}$$

We can also use Q-learning with learning rate a to update the Q-table for the source task. For iteration $j = 0, 1, \ldots, j_{\max}$

$$\begin{aligned} Q'_{(j+1)}(x_k, u_k) = {} & Q'_{(j)}(x_k, u_k) \\ & + a \Big[R_s(x_k, u_k) \\ & \quad + \gamma \min_{u_{k+1}} Q'_{(j)}(x_{k+1}, u_{k+1}) \\ & \quad - Q'_{(j)}(x_k, u_k) \Big]. \end{aligned} \tag{19}$$

Then

$$Q'_i(x_k, u_k) = \alpha_i Q'_{(j_{\max})}(x_k, u_k). \tag{20}$$

Additional approaches can also be used such as inverse RL, imitation learning, and other means. Once the Q-value from the source task is obtained, it can be applied as prior knowledge for transfer to target task. As such, this class of representation approaches actually include several different implementation procedures.

2) Indirectly Learned Value From Source Tasks: The indirect method formulates transferable knowledge based on a batch of samples of (x_k, u_k, x_{k+1}) from the source task, based on which we extract system dynamics in the form of $F_s(x_k, u_k)$ by supervised learning. For deterministic MDPs, F_s predicts the next state $\hat{x}_{k+1}$ given samples of (x_k, u_k, x_{k+1})

$$\hat{x}_{k+1} = F_s(x_k, u_k). \tag{21}$$

For stochastic MDPs, F_s predicts the transition probability. The system model F_s is learned using general function approximation models to minimize the following loss function:

$$L_{F_s} = \left(\hat{x}_{k+1} - x_{k+1}\right)^2. \tag{22}$$

Recently, several works reported that relaxing the Markov assumption and using a recurrent model can effectively retain necessary past information to prevent aliasing. This has also been empirically shown by A3C + LSTM [25]. Additional advantage of using dynamic model F_s is to allow offline off-policy training, which can be achieved by using samples generated from an exploratory policy (usually uniform random action selection). As such, we can bootstrap data collected from previous tasks, and having a batch of data from an exploratory policy generally leads to more stable learning, compared to on-policy training.

After a regression model F_s is obtained, we can now formulate a $Q'(x_k, u_k)$ value function based on N-step return by using $F_s(x_k, u_k)$: with an initialization of (x_j, u_j), we can generate a sequence of N state and action pairs (x_j, u_j), $(x_{j+1}, \pi_i(x_{j+1})), \ldots, (x_{j+N-1}, \pi_i(x_{j+N-1}))$ using (21) and policy π_i from (14). We then define prior knowledge $Q'_i(x_k, u_k)$ as

$$Q'_i\left(x_j, u_j\right) = \alpha_i \left[R_s\left(x_j, u_j\right) + \sum_{j=1}^{N-1} \gamma^j R_s\left(x_{j+1}, \pi_i\left(x_{j+1}\right)\right) \right]. \tag{23}$$

We now summarize the RL-KS algorithm in Algorithm 1.

Algorithm 1 RL-KS

Input: Samples $\{(x_k, u_k, x_{k+1})\}$, which include those from source task $\mathcal{T}_s$.
Output: Optimal policy for target task $\mathcal{T}_t$.
Initialization: Set iteration index $i = 0, \ldots, i_{max}$, initial knowledge weighting factor $0 < \alpha_0 < 1$, number of steps N.
Hyperparameter: Knowledge weighting factor α_i decreases from α_0 to 0, e.g. $\alpha_i = \alpha_0^i$.

```
if T_s is a finite MDP then
    Obtain Q'_i from (17)-(20).
else
    Learn prior knowledge F_s in (21) using samples (x_k, u_k, x_{k+1}) from T_s.
    Get transfer knowledge Q'_i from (23).
    Get R' from (9) and (10).
end if
while i < i_max do
    Obtain policy π_i from (14).
    Collect sample (x_k, u_k, x_{k+1}) from task T_t.
    Update Q-value function Q_{i+1} from (15).
    i ← i + 1, k ← k + 1.
end while
return Optimal policy for target task T_t.
```

V. Analytical Performance Guarantee

Here we provide performance guarantees for the proposed RL-KS.

A. Properties of the Proposed Algorithm Without Approximation Error

The actor-critic update in (16) will generate a sequence of iterative Q-value functions $\{Q_i(x_k, u_k)\}$. In this section, the convergence of the RL-KS algorithm will be proved by showing that the sequence $\{Q_i(x_k, u_k)\}$ converges to the optimal value function $J^*(x_k, u_k)$ without considering approximation error.

Lemma 1: Let $\{\pi_i\}$ be the control laws as in (14). Define Q_i as in (15). Let $\{\mu_i\}$ be a sequence of control laws associated with Λ_i below

$$\Lambda_{i+1}(x_k, u_k) = R'(x_k, u_k) + \gamma \Lambda_i(x_{k+1}, \mu_i(x_{k+1})). \tag{24}$$

If $Q_0(\cdot) = \Lambda_0(\cdot) = 0$, then $Q_{i+1}(x_k, u_k) \leq \Lambda_{i+1}(x_k, u_k), \forall i$.

Proof: It can be derived by noticing that Q_{i+1} is the result of minimizing the right-hand side of (15) with respect

to the control input u_{k+1}, while Λ_{i+1} is a result of an arbitrary control input. □

Lemma 2: Let the sequence $\{Q_i\}$ be defined as in (15). If the system is controllable on Ω and $Q'(0,0) = 0$, then there is an upper bound Y such that $0 \le Q_i(x_k, u_k) \le Y$, for $i = 1, 2, \ldots$

Proof: Let $\eta(x_k)$ be a continuous mapping from x_k, and the resulting controls are upper bounded by saturation, and let $Z_0(\cdot) = 0$, and Z_i is updated by

$$Z_{i+1}(x_k, u_k) = R'(x_k, u_k) + \gamma Z_i(x_{k+1}, \eta(x_{k+1})) \tag{25}$$

$$Z_1(x_k, u_k) = R'(x_k, u_k). \tag{26}$$

Let $Z_i(x_k, \eta) = Z_i(x_k, \eta(x_k))$ be a shorthand notation. Noticing the difference

$$\begin{aligned} &Z_{i+1}(x_k, u_k) - Z_i(x_k, u_k) \\ &\quad = \gamma\big[Z_i(x_{k+1}, \eta) - Z_{i-1}(x_{k+1}, \eta)\big] \\ &\quad = \gamma^2(Z_{i-1}(x_{k+2}, \eta) - Z_{i-2}(x_{k+2}, \eta)) \\ &\quad \vdots \\ &\quad = \gamma^i(Z_1(x_{k+i}, \eta) - Z_0(x_{k+i}, \eta)) \\ &\quad = \gamma^i Z_1(x_{k+i}, \eta) \end{aligned} \tag{27}$$

we can obtain

$$\begin{aligned} &Z_{i+1}(x_k, u_k) \\ &\quad = \gamma^i Z_1(x_{k+i}, \eta) + Z_i(x_k, u_k) \\ &\quad = \gamma^i Z_1(x_{k+i}, \eta) + \gamma^{i-1} Z_1(x_{k+i-1},) + Z_{i-1}(x_k, u_k) \\ &\quad = \gamma^i Z_1(x_{k+i}, \eta) + \gamma^{i-1} Z_1(x_{k+i-1}, \eta) \\ &\qquad + \gamma^{i-2} Z_1(x_{k+i-2}, \eta) + Z_{i-2}(x_k, u_k) \\ &\quad = \gamma^i Z_1(x_{k+i}, \eta) + \gamma^{i-1} Z_1(x_{k+i-1}, \eta) \\ &\qquad + \gamma^{i-2} Z_1(x_{k+i-2}, \eta) + \cdots + \gamma Z_1(x_{k+1}, \eta) + Z_1(x_k, \eta) \\ &\quad = \sum_{j=0}^{i} \gamma^j Z_1\big(x_{k+j}, \eta\big). \end{aligned} \tag{28}$$

According to (26) and (28)

$$\begin{aligned} Z_{i+1}(x_k, u_k) &= \sum_{j=0}^{i} \gamma^j R'\big(x_{k+j}, u_{k+j}\big) \\ &\le \sum_{j=0}^{\infty} \gamma^j R'\big(x_{k+j}, u_{k+j}\big). \end{aligned} \tag{29}$$

Since $R'(x_{k+j}, u_{k+j})$ is bounded, there exists a finite Y such that

$$\sum_{j=0}^{\infty} \gamma^j R'\big(x_{k+j}, u_{k+j}\big) \le Y \quad \forall i. \tag{30}$$

By using Lemma 1, we get

$$Q_{i+1}(x_k, u_k) \le Z_{i+1}(x_k, u_k) \le Y \quad \forall i \tag{31}$$

so the proof is completed. □

Based on Lemmas 1 and 2, we now present the convergence proof of the Q-value function sequence.

Theorem 1: Define the sequence $\{Q_i\}$ as in (15) with $Q_0 = 0$, and the control law sequence $\{\pi_i\}$ as in (14). Let the source task knowledge $Q'(x_k, u_k) \ge 0$. Then, we can conclude that $\{Q_i\}$ is a nondecreasing sequence satisfying $Q_i \le Q_{i+1}, \forall i$.

Proof: Define a new sequence $\{\Phi_i\}$ as

$$\Phi_{i+1}(x_k, u_k) = R'(x_k, u_k) + \gamma \Phi_i(x_{k+1}, \pi_{i+1}(x_{k+1})) \tag{32}$$

where $\Phi_0 = Q_0 = 0$. Now we show that $\Phi_i(x_k, u_k) \le Q_{i+1}(x_k, u_k)$. Note that we use the shorthand notation in the following proof, e.g. $\Phi_i(x_{k+1}, \pi_{i+1}) = \Phi_i(x_{k+1}, \pi_{i+1}(x_{k+1}))$.

First, we prove that it holds for $i = 0$. Since

$$Q_1(x_k, u_k) = R'(x_k, u_k) \tag{33}$$

$$Q_1(x_k, u_k) - \Phi_0(x_k, u_k) = R'(x_k, u_k) \ge 0 \tag{34}$$

we have

$$\Phi_0(x_k, u_k) \le Q_1(x_k, u_k). \tag{35}$$

Second, we assume that it holds for $i-1$, i.e., $\Phi_{i-1}(x_k, u_k) \le Q_i(x_k, u_k), \forall x_k$. Then, for i, from (15) and (32), we get

$$\begin{aligned} &Q_{i+1}(x_k, u_k) - \Phi_i(x_k, u_k) \\ &\qquad = \gamma\big[Q_i(x_{k+1}, \pi_i) - \Phi_{i-1}(x_{k+1}, \pi_i)\big] \ge 0 \end{aligned} \tag{36}$$

that is,

$$\Phi_i(x_k, u_k) \le Q_{i+1}(x_k, u_k). \tag{37}$$

Thus, the above equation is true for any i by mathematical induction.

Furthermore, according to Lemma 1, we know that $Q_i(x_k, u_k) \le \Lambda_i(x_k, u_k)$. Combining with (37), we have

$$Q_i(x_k, u_k) \le \Phi_i(x_k, u_k) \le Q_{i+1}(x_k, u_k) \tag{38}$$

which completes the proof. □

According to Lemma 2 and Theorem 1, we can obtain that $\{Q_i\}$ is a monotonically nondecreasing sequence with an upper bound, and therefore, its limit exists. Here, we define the limit as $\lim_{i\to\infty} Q_i(x_k, u_k) = Q_\infty(x_k, u_k)$ and present the following theorem.

Theorem 2: Let the cost function sequence $\{Q_i\}$ be defined as in (15). Then, its limit satisfies

$$Q_\infty(x_k, u_k) = R'(x_k, u_k) + \gamma \min_{u_{k+1}} Q_\infty(x_{k+1}, u_{k+1}). \tag{39}$$

Proof: For any u_{k+1} and i, according to (15), we can derive

$$Q_i(x_k, u_k) \le R'(x_k, u_k) + \gamma Q_{i-1}(x_{k+1}, u_{k+1}). \tag{40}$$

Combining with

$$Q_{i-1}(x_{k+1}, u_{k+1}) \le Q_\infty(x_{k+1}, u_{k+1}) \tag{41}$$

which is obtained from (38), we have

$$Q_i(x_k, u_k) \le R'(x_k, u_k) + \gamma Q_\infty(x_{k+1}, u_{k+1}). \tag{42}$$

Let $i \to \infty$, we have

$$Q_\infty(x_k, u_k) \le R'(x_k, u_k) + \gamma Q_\infty(x_{k+1}, u_{k+1}). \tag{43}$$

Note that in the above equation, u_{k+1} is chosen arbitrarily, thus, it implies that

$$Q_\infty(x_k, u_k) \le R'(x_k, u_k) + \gamma \min_{u_{k+1}} Q_\infty(x_{k+1}, u_{k+1}). \tag{44}$$

On the other hand, since the cost function sequence satisfies

$$Q_{i+1}(x_k,u_k)=R'(x_k,u_k)+\gamma\min_{u_{k+1}}Q_i(x_{k+1},u_{k+1}) \tag{45}$$

applying inequality (41), and letting $i\to\infty$, we have

$$Q_\infty(x_k,u_k)\geq R'(x_k,u_k)+\gamma\min_{u_{k+1}}Q_\infty(x_{k+1},u_{k+1}). \tag{46}$$

Based on (44) and (46), we can conclude that (39) is true. □

Theorem 3: Let $Q_\infty(x_k,u_k)=\lim_{i\to\infty}Q_i(x_k,u_k)$ and $\pi_\infty(x_k)=\lim_{i\to\infty}\pi_i(x_k)$, then the Q-value sequence $Q_i(x_k,u_k)$ and the policy $\pi_i(x_k)$ converge to the optimal value J^* and optimal policy π^* as defined in (3) and (4), respectively,

$$Q_\infty(x_k,u_k)=J^*(x_k,u_k) \tag{47}$$

$$\pi_\infty(x_k)=\pi^*(x_k). \tag{48}$$

Proof: We need Theorem 2 to show this result. Notice that from the relationship between (14) and (15), we have

$$\begin{aligned}Q_\infty(x_k,u_k)&=R'(x_k,u_k)+\gamma\min_{u_{k+1}}Q_\infty(x_{k+1},u_{k+1})\\&=R'(x_k,u_k)+\gamma Q_\infty(x_{k+1},\pi_\infty(x_{k+1}))\end{aligned} \tag{49}$$

and

$$\pi_\infty(x_k)=\arg\min_{u_k}Q_\infty(x_k,u_k). \tag{50}$$

As $i\to\infty$, the knowledge weighting factor $\alpha\to 0$. Therefore, $R'(x_k,u_k)\to R(x_k,u_k)$. Thus (49) becomes

$$Q_\infty(x_k,u_k)=R(x_k,u_k)+\gamma Q_\infty(x_{k+1},\pi_\infty(x_{k+1})). \tag{51}$$

Comparing (3) and (4) with (50) and (51), we conclude that (47) and (48) are true. □

B. Properties of the Proposed Algorithm With Approximation Errors

Considering approximation errors in (14) and (15), starting from $\hat{Q}_0(x_k,u_k)=0$ with an initial $\hat{\pi}_0$, for $i=0,1,2,\ldots$, we have

$$\hat{\pi}_i(x_k)=\arg\min_{u_k}\hat{Q}_i(x_k,u_k)+\rho_i(x_k) \tag{52}$$

$$\begin{aligned}\hat{Q}_{i+1}(x_k,u_k)&=R'(x_k,u_k)+\gamma\hat{Q}_i(x_{k+1},\pi_i(x_{k+1}))\\&\quad+\varrho_i(x_k,u_k)\end{aligned} \tag{53}$$

where $\rho_i(x_k)$ and $\varrho_i(x_k,u_k)$ are finite approximation error functions of the iterative control and Q-value function, respectively. For convenience of analysis, for $\forall i=0,1,\ldots$, we assume that $\rho_i(x_k)=0$ and $\varrho_i(x_k,u_k)=0$ for $x_k=0$ and $u_k=0$.

Considering approximation errors, we generally have $\hat{\pi}_i(x_k)\neq\pi_i(x_k)$, $\hat{Q}_{i+1}(x_k,u_k)\neq Q_i(x_k,u_k)$, $i=0,1,\ldots$, and the convergence property in Theorems 1–3 for accurate case becomes invalid. Hence, in this section, new convergence criteria will be established considering approximation errors in each iteration, which makes the iterative Q-value function converge to a finite neighborhood of the optimal one.

Define the target Q-value function as

$$\begin{aligned}\Gamma_i'(x_k)=\min_{u_k}\big\{&R'(x_k,u_k)+\gamma\big[\hat{Q}_{i-1}(x_{k+1},u_{k+1})\\&+Q_{i-1}'(x_{k+1},u_{k+1})\big]\big\}\end{aligned} \tag{54}$$

and

$$\Gamma_i(x_k)=\min_{u_k}\big\{R'(x_k,u_k)+\gamma\hat{Q}_{i-1}(x_{k+1},u_{k+1})\big\} \tag{55}$$

where $\hat{Q}_0(x_k,u_k)=\Gamma_0(x_k,u_k)=0$.

For $i=0,1,2,\ldots$, there exist finite constants $\upsilon\geq 1$ and $\tau\geq 1$ that makes

$$\hat{Q}_i(x_k,u_k)\leq\upsilon\Gamma_i'(x_k,u_k)\leq\upsilon\tau\Gamma_i(x_k,u_k) \tag{56}$$

hold uniformly. Let $\sigma=\upsilon\tau$, it becomes

$$\hat{Q}_i(x_k,u_k)\leq\sigma\Gamma_i(x_k,u_k). \tag{57}$$

Hence, we can give the following theorem.

Theorem 4: Let $x_k\in\mathbb{R}^n$ be an arbitrary state and Assumption 1 hold. For $i=0,1,\ldots$, let $\Gamma_i(x_k)$ be expressed as (55) and $\hat{Q}_i(x_k,u_k)$ be expressed as (53). Let $0<\lambda<\infty$ and $1\leq\delta<\infty$ be constants that make

$$J^*(x_{k+1},u_{k+1})\leq\lambda R'(x_k,u_k) \tag{58}$$

hold uniformly. If there exists $1\leq\sigma<\infty$ that makes (56) hold uniformly, then we have

$$\hat{Q}_i(x_k,u_k)\leq\sigma\left(1+\sum_{j=1}^{i}\frac{\lambda^j\sigma^{j-1}(\sigma-1)}{(\lambda+1)^j}\right)J^*(x_k,u_k) \tag{59}$$

where $i=0,1,\ldots$

Proof: The theorem can be proved by mathematical induction. First, let $i=0$. Then, (59) becomes

$$\hat{Q}_0(x_k,u_k)\leq\sigma J^*(x_k,u_k). \tag{60}$$

This can be obtained as $\hat{Q}_0(x_k,u_k)\equiv 0\leq J^*(x_k,u_k)\leq\sigma J^*(x_k,u_k)$. Therefore, the conclusion holds for $i=0$.

Next, let $i=1$. According to (55), we have

$$\begin{aligned}&\Gamma_1(x_k,u_k)\\&=\min_{u_k}\big\{R'(x_k,u_k)+\gamma\hat{Q}_0(x_{k+1},u_{k+1})\big\}\\&\leq\min_{u_k}\big\{R'(x_k,u_k)+\gamma\sigma J^*(x_{k+1},u_{k+1})\big\}\\&\leq\min_{u_k}\left\{\left(1+\lambda\frac{\sigma-1}{\lambda+1}\right)R'(x_k,u_k)\right.\\&\qquad\left.+\gamma\left(\sigma-\frac{\sigma-1}{\lambda+1}\right)J^*(x_{k+1},u_{k+1})\right\}\\&=\left(1+\lambda\frac{\sigma-1}{\lambda+1}\right)\min_{u_k}\big\{R'(x_k,u_k)+\gamma J^*(x_{k+1},u_{k+1})\big\}\\&=\left(1+\lambda\frac{\sigma-1}{\lambda+1}\right)J^*(x_k,u_k).\end{aligned} \tag{61}$$

Note that in the derivation above, $\gamma J^*(x_{k+1},u_{k+1})\leq J^*(x_{k+1},u_{k+1})\leq\lambda R'(x_k,u_k)$ holds according to (58), we have

$$\hat{Q}_1(x_k,u_k)\leq\sigma\left(1+\lambda\frac{\sigma-1}{\lambda+1}\right)J^*(x_k,u_k) \tag{62}$$

which shows that (59) holds for $i=1$.

Assume that (59) holds for $i=l-1$, where $l=1,2,\ldots$ Then, for $i=l$, let $c=\sum_{j=1}^{l-1}((\lambda^{j-1}\sigma^{j-1}$

$(\sigma-1))/((\lambda+1)^j))$, $d = ((\lambda^{l-1}\sigma^{l-1}(\sigma-1))/((\lambda+1)^l))$, we have

$$\begin{aligned}
&\Gamma_l(x_k,u_k)\\
&\quad=\min_{u_k}\{R'(x_k,u_k)+\gamma\hat{Q}_{l-1}(x_{k+1},u_{k+1})\}\\
&\quad\le\min_{u_k}\{R'(x_k,u_k)+\gamma\sigma(1+\lambda c)J^*(x_{k+1},u_{k+1})\}\\
&\quad\le\min_{u_k}\{R'(x_k,u_k)+\gamma\sigma(1+\lambda c)J^*(x_{k+1},u_{k+1})\}\\
&\quad\le\min_{u_k}\{(1+\lambda c+\lambda d)R'(x_k,u_k)\\
&\qquad\qquad+\gamma[\sigma(1+\lambda c)-c-d]J^*(x_{k+1},u_{k+1})\}\\
&\quad=\left(1+\lambda\sum_{j=1}^{l-1}\frac{\lambda^{j-1}\sigma^{j-1}(\sigma-1)}{(\lambda+1)^j}+\lambda\frac{\lambda^{l-1}\sigma^{l-1}(\sigma-1)}{(\lambda+1)^l}\right)\\
&\qquad\times\min_{u_k}\{R'(x_k,u_k)+\gamma J^*(x_{k+1},u_{k+1})\}\\
&\quad=\left(1+\sum_{j=1}^{l}\frac{\lambda^{j}\sigma^{j-1}(\sigma-1)}{(\lambda+1)^j}\right)J^*(x_k,u_k). \qquad (63)
\end{aligned}$$

Then, according to (56), we can obtain (59), which proves the conclusion for $i=0,1,\ldots$ □

Theorem 5: Let $x_k\in\mathbb{R}^n$ be an arbitrary state and Assumption 1 hold. Suppose Theorem 4 holds for $\forall x_k\in\mathbb{R}^n$. If for $0<\lambda<\infty$ the inequality

$$1\le\sigma\le\frac{\lambda+1}{\lambda} \qquad (64)$$

holds, then as $i\to\infty$, approximated value function $\hat{Q}_i(x_k,u_k)$ in (53) is uniformly convergent to a bounded neighborhood of the optimal value function $J^*(x_k,u_k)$, that is,

$$\lim_{i\to\infty}\hat{Q}_i(x_k,u_k)\le\sigma\left(1+\frac{\lambda(\sigma-1)}{1-\lambda(\sigma-1)}\right)J^*(x_k,u_k). \qquad (65)$$

Proof: According to (63) in Theorem 4, we can see that for $j=1,2,\ldots$ the sequence $\{((\lambda^j\sigma^{j-1}(\sigma-1))/((\lambda+1)^j))\}$ is a geometric series. Then, (63) can be written as

$$\Gamma_i(x_k,u_k)=\left(1+\frac{\frac{\lambda(\sigma-1)}{(\lambda+1)}\left(1-\left(\frac{\lambda\sigma}{\lambda+1}\right)^i\right)}{1-\frac{\lambda\sigma}{\lambda+1}}\right)J^*(x_k,u_k). \qquad (66)$$

As $i\to\infty$, if $1\le\sigma\le((\lambda+1)/\lambda)$, then (66) becomes

$$\begin{aligned}
\lim_{i\to\infty}\Gamma_i(x_k,u_k)&=\Gamma_\infty(x_k,u_k)\\
&\le\left(1+\frac{\lambda(\sigma-1)}{1-\lambda(\sigma-1)}\right)J^*(x_k,u_k). \qquad (67)
\end{aligned}$$

According to (56), let $i\to\infty$, we have

$$\hat{Q}_\infty(x_k,u_k)\le\sigma\Gamma_\infty(x_k,u_k). \qquad (68)$$

From (67) and (68), we can obtain (65). □

C. Positive Knowledge Transfer

We consider a knowledge transfer by first defining a task similarity measure between $\mathcal{T}_s$ and $\mathcal{T}_t$.

Definition 2: Consider the iterative cost-to-go $J_{i+1}(x_k,u_k)$ in (6) and (7), and the cost-to-go with knowledge transfer $Q_{i+1}(x_k,u_k)$ in (14) and (15), respectively. A source task $\mathcal{T}_s$ and a target task $\mathcal{T}_t$ are similar if $\mathcal{D}_{\mathcal{T}_s,\mathcal{T}_t}(x_k,u_k)=J_{i+1}(x_k,u_k)-Q_{i+1}(x_k,u_k)<\tau$ for all $i=0,1,\ldots$, where τ is a positive threshold value.

Definition 3: The N-step return for the iterative cost-to-go $J_{i+1}(x_k,u_k)$ in (6) and (7) is defined as $J_i^N(x_k,u_k)=\sum_{j=1}^{N-1}\gamma^jR(x_{k+j},\xi_i(x_{k+j}))$, and the N-step return for the cost-to-go with knowledge transfer $Q_{i+1}(x_k,u_k)$ in (14) and (15) is defined as $Q_i^N(x_k,u_k)=\sum_{j=1}^{N-1}\gamma^jR'(x_{k+j},\pi_i(x_{k+j}))$.

We would like to show how learning performance can be improved with knowledge transfer by a transfer measure. Let $C(\mathcal{T}_s,\mathcal{T}_t)\in\mathbb{R}$ below denote the benefit of transferring learned knowledge in source task $\mathcal{T}_s$ to the target task $\mathcal{T}_t$. Then $C(\mathcal{T}_s,\mathcal{T}_t)>0$ indicates positive transfer, while $C(\mathcal{T}_s,\mathcal{T}_t)<0$ indicates negative transfer. We denote such transfer benefit by using a jump-start measure

$$C(\mathcal{T}_s,\mathcal{T}_t)=J_i^N(x_k,u_k)-Q_i^N(x_k,u_k). \qquad (69)$$

Our result for the benefit of a transfer is given below.

Theorem 6: Let there exist $\tau>0$, and assume that the source task $\mathcal{T}_s$ and the target task $\mathcal{T}_t$ are sufficiently similar accordingly to Definition 2. Let $R_f(x_k,u_k)\ge\tau$, where R_f is as in (9). Then the benefit of transfer $C(\mathcal{T}_s,\mathcal{T}_t)>0$ as $N\to\infty$.

Proof: Equations (7) and (15) can be written as

$$\begin{aligned}
&J_{i+1}(x_k,u_k)\\
&\quad=R(x_k,u_k)+J_i^N(x_k,u_k)+\sum_{j=N}^{\infty}\gamma^jR\big(x_{k+j},\xi_i\big(x_{k+j}\big)\big) \qquad (70)
\end{aligned}$$

and

$$\begin{aligned}
Q_{i+1}(x_k,u_k)&=R'(x_k,u_k)+Q_i^N(x_k,u_k)\\
&\quad+\sum_{j=N}^{\infty}\gamma^jR'\big(x_{k+j},\pi_i\big(x_{k+j}\big)\big). \qquad (71)
\end{aligned}$$

As $N\to\infty$, $\sum_{j=N}^{\infty}\gamma^jR(x_{j+1},\xi_i(x_{j+1}))\to0$ and $\sum_{j=N}^{\infty}\gamma^jR'(x_{j+1},\pi_i(x_{j+1}))\to0$ hold. Then (70) and (71) become

$$J_{i+1}(x_k,u_k)=R(x_k,u_k)+J_i^N(x_k,u_k) \qquad (72)$$

and

$$Q_{i+1}(x_k,u_k)=R'(x_k,u_k)+Q_i^N(x_k,u_k). \qquad (73)$$

Next, we are going to prove $J_i^N(x_k,u_k)>Q_i^N(x_k,u_k)$ holds for the following situations.

1) If $Q_{i+1}(x_k,u_k)\le J_{i+1}(x_k,u_k)$, then we can get $J_i^N(x_k,u_k)>Q_i^N(x_k,u_k)$ from (72) and (73) because $R'(x_k,u_k)=R(x_k,u_k)+R_f(x_k,u_k)>R(x_k,u_k)$.
2) If $Q_{i+1}(x_k,u_k)>J_{i+1}(x_k,u_k)$, according to Definition 2 we have $Q_{i+1}(x_k,u_k)-J_{i+1}(x_k,u_k)<\tau\le R_f(x_k,u_k)$. Then from (72) and (73)

$$\begin{aligned}
&J_i^N(x_k,u_k)-Q_i^N(x_k,u_k)\\
&\quad=\big[J_{i+1}(x_k,u_k)-R(x_k,u_k)\big]\\
&\qquad-\big[Q_{i+1}(x_k,u_k)-R'(x_k,u_k)\big]\\
&\quad=J_{i+1}(x_k,u_k)-Q_{i+1}(x_k,u_k)+R_f(x_k,u_k)\\
&\quad>0. \qquad (74)
\end{aligned}$$

Hence, both cases result in a positive transfer for $i = 0, 1, \ldots$ □

Theorem 6 is formulated and proved relying on the task similarity measure within a similarity threshold τ, namely that smaller τ means more similarity between the tasks, and that prior knowledge gained from the source task will benefit the learning of the target task. In other words, the transfer is a positive transfer. However, currently how to properly select τ remains an open question and heuristics are still needed. It is a trade-off between making transfers between a wider range of tasks and ensuring positive transfer. In Section VII, we provide a few examples to illustrate this task similarity measure and positive learning transfer.

VI. Implementation of RL-KS

Here we provide implementation procedures of the proposed RL-KS algorithm and different knowledge representations including Q-Table and actor-critic.

A. Using Q-Table

For simple cases such as finite state MDPs and the state and action spaces are small enough, the value functions can be represented as tables. In this tabular case, exact solutions can often be found by updating the values in the Q-table. The proposed algorithm first obtain knowledge shaping function Q' from the source task as in (17)–(20), then follows (10) and (16) for updating the Q-values.

B. Actor-Critic Neural Networks

For the case of large or continuous state and control space problems, the proposed RL-KS method is implemented using an actor-critic structure. Here we provide the implementation details used in our experiments.

1) Critic Network: The critic network consists of three layers of neurons, namely the input layer, the hidden layer and the output layer. It takes the state x_k and the action u_k as inputs. We chose sigmoid as the activation function in our implementation. The prediction error $e_{c,k} \in \mathbb{R}$ of the critic network can be written as

$$e_{c,k}^{(i+1)} = R'(x_k, u_k) + \gamma \hat{Q}_i\left(x_{k+1}, \pi^{(i)}(x_{k+1})\right) - \hat{Q}_{i+1}(x_k, u_k) \tag{75}$$

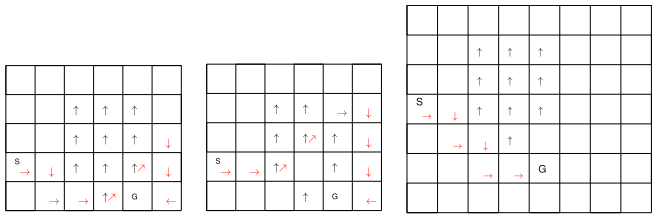
where $\hat{Q}_i$ is the approximated value function.

To correct the prediction error, the weight update objective is to minimize the squared prediction error $E_{c,k}$, denoted as

$$E_{c,k}^{(i+1)} = \frac{1}{2}\left(e_{c,k}^{(i+1)}\right)^2. \tag{76}$$

2) Action Network: The action network also has three layers of neurons with the output layer neuron having a hyperbolic tangent activation function. The input to the action network is x_k and the out is u_k. Under our problem formulation, the objective of adapting the action network is to minimize the following performance error based on the approximated Q-value:

$$E_{a,k}^{(i+1)} = \frac{1}{2}\hat{Q}_{i+1}^2(x_k, u_k). \tag{77}$$



(a) (b) (c)

Fig. 1. Windy gridworld problem. The black arrows indicate wind directions, and the red arrows indicate an optimal path. (a) Original gridworld and an optimal path. (b) Changed wind condition. (c) Changed grid size and goal location.

VII. Experimental Results

In this section, we evaluate RL-KS performance based on three experiments, including a finite state MDP problem "windy gridworld," and two continuous control problems namely the cart-pole balancing problem and the real-time control of a robotic knee with a human user in the loop. In several evaluations, we compare RL-KS with two baselines: 1) a naive actor-critic and 2) a pretrained actor-critic. The former is based on [26] and trained from scratch for new tasks, while the latter is obtained by training the naive actor-critic for the source task. All hyperparameters are given below in respective tasks.

A. Windy Gridworld

Fig. 1(a) shows a rectangular "windy gridworld" representation of a simple finite MDP, which is adopted from [27] and [28]. In this problem, the agent's actions are affected by the wind directions. The left panel of Fig. 1 shows an optimal policy (path) for the original gridworld setup. Four actions are possible at each cell: north, south, east, and west, which deterministically cause the agent to move one cell in the indicated direction. If the agent takes an action that would take it off the grid, its location will remain unchanged. Notice that the "windy" states are indicated by arrows, where the agent experiences an extra "push" action that makes it move one step toward the indicated wind direction. For example, if the agent is in a windy state with the north wind (pointing upward) and executes an action to the right, then the agent will end up moving right one cell but also another upward move. As a result, the agent moves diagonally upward to the right.

Each episode includes a maximum of 30 time steps. At the beginning of each episode, the agent starts from the "S" state and moves according to some policy until it reaches a maximum of 30 steps or until it reaches the "G" state. The reward is -0.1 everywhere except the goal state which is 10. Therefore the goal is to maximize the reward.

We test the idea of RL-KS for two cases: 1) wind condition change shown in the center panel of Fig. 1, and 2) grid size and goal state change shown in the right panel of Fig. 1. In the first case, the nine windy cells ranging from (2, 3) to (4, 5) each has a probability of 0.2 to become a wind-less cell and a probability of 0.2 to change wind direction (here (a, b) means ath row and bth column, counting from the top-right corner). In the second case, the size of the map is extended from a 5×6 to a size of 7×8. Meanwhile, the goal state is randomly chosen from cells (5, 5), (5, 6), (6, 5), and (6, 6).

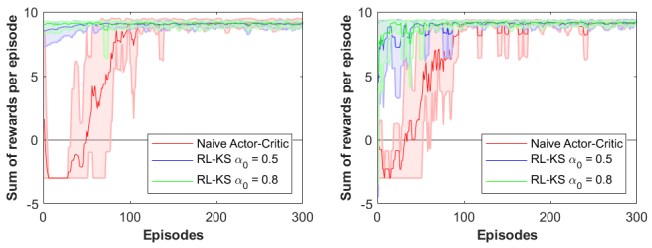


Fig. 2. Comparison of average episode reward of ten episodes. (a) Changed wind, (b) Changed size and goal. The shaded areas indicate the resulted performance experienced in the ten simulation episodes.

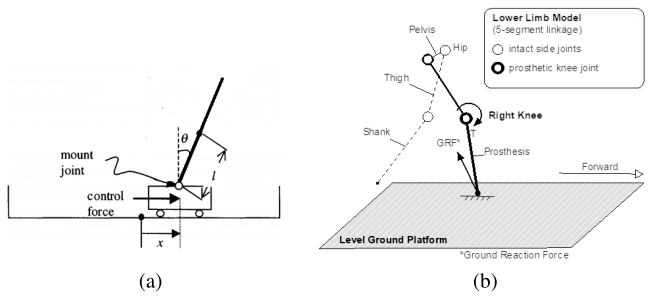


Fig. 3. Illustrations of (a) cart-pole problem, (b) OpenSim human-prosthesis problem.

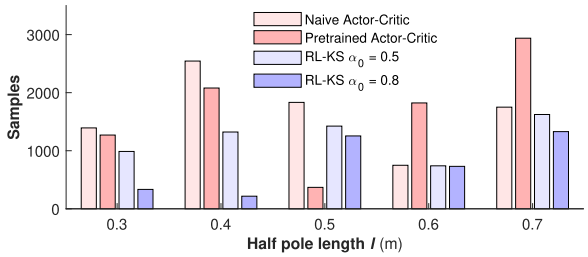


Fig. 4. Average number of samples required for the cart-pole problem. The source task is with half pole length $l = 0.5$ m, and the target tasks are with $l = 0.3$, 0.4, 0.5, 0.6, and 0.7 m, respectively.

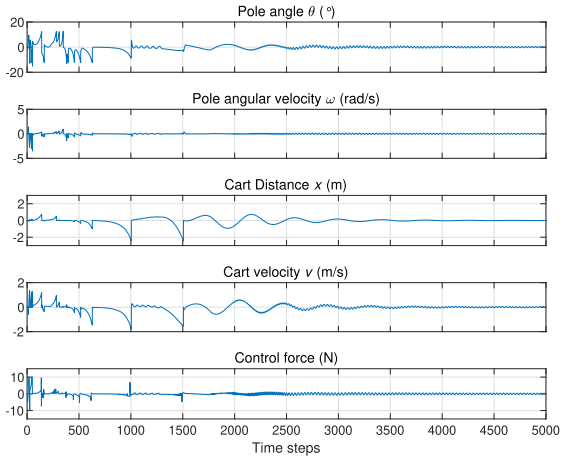


Fig. 5. Example trajectories of states and control during training where the states are initialized with zero values except the initial pole angle is 5°.

We solve this windy gridworld problem using RL-KS and the naive actor-critic. Both algorithms here use a tabular form to store the Q-values, and an ϵ-greedy exploration strategy with $\epsilon = 0.1$. The learning rate a is 0.1 and the discount rate $\gamma = 0.9$. For RL-KS, we use $N = 20$ and we evaluate $\alpha_0 = 0.5$ and 0.8 during experiments.

We set up our experiments as follows. For each of the "changed wind" and "changed goal and size" experiments, we first randomly generate ten source tasks and ten target tasks, respectively. We set the task similarity threshold $\tau = 10$. From the ten source and target tasks, we chose the pair that is the most similar. For the source task, we obtain Q' following (17)–(20). Then, the Q-table of the target task is updated following an RL-KS update as in (16).

Fig. 2 summarizes performance of RL-KS under two different knowledge weighting factors, and a benchmark case using naive actor-critic (which is dHDP). Performances are respectively shown based on average episode reward (the total reward an agent has collected in each of ten episodes) for changed wind or changed size and goal. In both scenarios, RL-KS has outperformed the baseline. We also notice that the larger the α_0 is the more beneficial to the target task from transferring knowledge.

B. Inverted Pendulum Cart-Pole Balancing Problem

We use the same cart-pole system [Fig. 3(a)] as in [26], except that the control force that pushes or pulls the cart is continuous rather than binary. We test RL-KS performance for changing the pole length from source task to target tasks.

This cart-pole model has four state variables: pole angle, pole angular velocity, cart distance, and cart velocity. We have adopted the settings from [26] where a run consists of a maximum of 1000 consecutive trials. It is considered successful if the last trial (trial number less than 1000) of the run has lasted 6000 time steps. Otherwise, if the controller is unable to learn to balance the cart-pole within 1000 trials (i.e., none of the 1000 trials has lasted over 6000 time steps), then the run is considered unsuccessful. In our simulations, we have used 0.02 s for each time step, and a trial is a complete process from start to fall. A pole is considered fallen when the pole is outside the range of $[-12°, 12°]$ and/or the cart is beyond the range of $[-2.4, 2.4]$ m in reference to the central position on the track.

The default value of the half-pole length l is 0.5 m. We first obtain the system dynamics F_s with default length using linear regression $x_{k+1} = F_s(x_k, u_k)$. The action variable $u_k \in \mathbb{R}$ is the force applied on the cart. 200 samples of (x_k, u_k, x_{k+1}) are obtained by performing Monte-Carlo sampling on the cart-pole model, and the regression model F_s is built using these samples. Then the transferred knowledge Q' is computed based on (23).

The critic network and actor network for the cart-pole problem are implemented using feed-forward neural networks with 5-6-1 neurons and a 4-6-1 neurons, respectively. During training, both network weights are randomly initialized subject to a uniform distribution between -0.1 and 0.1. The discount factor γ in (15) is 0.9 and the maximum iteration number $i_{\max}$ is 5000. Finally, we let the similarity threshold $\tau = 25$, $N = 100$, and two α_0 choices (0.5 and 0.8) are evaluated.

With the regression model we obtain the prior knowledge Q' for transfer. We apply RL-KS to solve target tasks with different half-pole lengths. Specifically, the target tasks considered are $l = 0.3, 0.4, 0.5, 0.6, 0.7$ m. Note that to solve the source task of length 0.5 m, the RL-KS agent still needs learning to obtain the optimal value function Q^* as Q' represents partial knowledge. We have performed ten runs for each target task with the cart initially at center position and the pole initially placed at uniformly randomized angles between $[-5.7°, 5.7°]$. We record the average number of total samples (x_k, u_k, x_{k+1}) required to successfully complete each of the target tasks. Fig. 4 summarizes the average results from ten runs with randomly initialized network weights and initial pole positions for each algorithm setting. In the figure, naive actor-critic is based on [26] with identical network settings and other parameters (e.g. γ). The pretrained actor-critic is obtained by training the naive actor-critic in a run at $l = 0.5$ m with 200 samples which is comparable to RL-KS. Fig. 4 shows that RL-KS requires fewest samples to reach respective optimal solutions, except the case that the pretrained actor-critic works better when $l = 0.5$ m as the source and target tasks are identical. Fig. 5 is an example during training where the states are initialized with zero values except the initial pole angle is 5°. It is a result corresponding to the target task of $l = 0.6$ m.

C. Robotic Prosthesis Control With Human in the Loop

Robotic prosthesis mimics biological joints to generate torques to enable the robotic knee motion for an amputee user. The device needs to be adjusted by its knee joint impedance values based on mechanical sensor measurements in the prosthesis. The intrinsic controllers built into the devices provide generic, low-level automatic control of joint torques. As human users differ from weight to size and physical conditions, and as users have different life-style needs, extrinsic control is needed to customize the device by providing impedance parameter settings for individual users. These new powered devices signify the future of rehabilitation. But automatically fitting the device to a human user remains a major challenge to unleash the full potential of the robotic device. Few automatic tuning technologies are currently available. The only functioning solution relies on multiple sessions of manually tuning a small subset of a large number of impedance parameters one at a time, unable to account for the interacting effects of the parameters during each tuning session. This highly heuristic approach is time-consuming, costly, and not scalable to reaching the full potential of this powerful robotic device.

Reinforcement learning, especially transfer reinforcement learning is expected to provide the optimal impedance parameter tuning adaptively to meet individual user's needs. With the knowledge obtained from impedance control tuning of multiple users over multiple sessions, it is expected the improve tuning performance by transferring such knowledge to similar prosthesis users. In the following, we provide simulation evaluations to validate the transfer learning framework, RL-KS, in this important application problem. Specifically, We consider transferring knowledge of impedance control from one subject at a given body weight to several other subjects at different body weights.

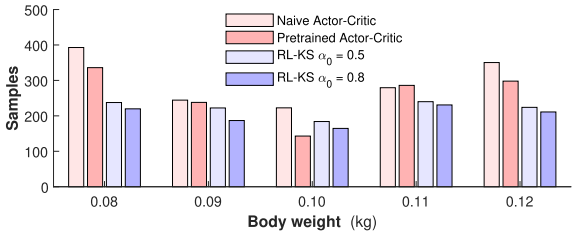


Fig. 6. Average number of samples required for the robotic prosthesis control problem simulated in OpenSim [29]. The source task is with a normalized body weight of 0.1 kg, and the target tasks are with normalized body weights of 0.08, 0.09, 0.10, 0.11, and 0.12 kg, respectively.

We use OpenSim [29], a widely used human locomotion simulation platform, to simulate level-ground walking of a human-prosthesis system. Fig. 3(b) shows the OpenSim lower limb walking model. In this model, five rigid-body segments linked by one-degree-of-freedom pin joints are used to represent the human body. The right knee is treated as a prosthetic knee and controlled by a finite-state impedance controller (FS-IC) modeled after a biological joint, while the other joints follow prescribed motions. The FS-IC divides a gait cycle into four phases, in each of the four phases the torque at the prosthetic knee is determined by the following impedance control law, $T = K(\theta - \theta_e) + B\omega$, where K, B and θ_e are the impedance parameters: K is stiffness, B is damping coefficient and θ_e is equilibrium position. The goal of automatically customizing a robotic knee for a user is to tune these impedance parameters optimally and adaptively to meet individual user's needs. Further comprehensive details can be found from [30], [31], [32], [33], and [34].

Our RL-KS-based tuning approach proceeds as follows. We first collect 50 samples of (x_k, u_k, x_{k+1}) from the OpenSim simulation model. As the source task, we set the subject's body weight at OpenSim normalized scale of 0.1 kg. The regression model F_s is determined by supervised learning: $x_{k+1} = F_s(x_k, u_k)$. Then prior knowledge Q' is computed based on F_s from $Q'(x_k, u_k) = \mathcal{Q}(F_s(x_k, u_k)) = \mathcal{Q}(x_{k+1}) = x_{k+1}^2$. After each gait cycle k, the differences (errors) between the measured knee angle profile and the target knee profile at the feature points, $x_k \in \mathbb{R}^2$, are computed and treated as the states of an RL controller. The adjustments to the three impedance parameters are the action denoted by $u_k \in \mathbb{R}^3$. We use a quadratic error to denote stage cost based on the state error and the control. The state, control, and cost objective are set up the same way as in [30].

The critic network and actor network for the OpenSim experiment are implemented using feed-forward neural networks with 5-8-1 neurons and 2-8-3 neurons, respectively. During training, both network weights are randomly initialized subject to a uniform distribution between -0.1 and 0.1. The discount factor γ in (15) is 0.95 and the maximum iteration number $i_{\max}$ is 200. We set $\tau = 6$, $N = 20$ and have evaluated two conditions of α_0 at 0.5 and 0.8.

We are now ready to transfer prior knowledge from the source task of a subject with an OpenSim normalized body weight of 0.1 kg to target tasks with different normalized body weights respectively at: 0.08, 0.09, 0.10, 0.11, and

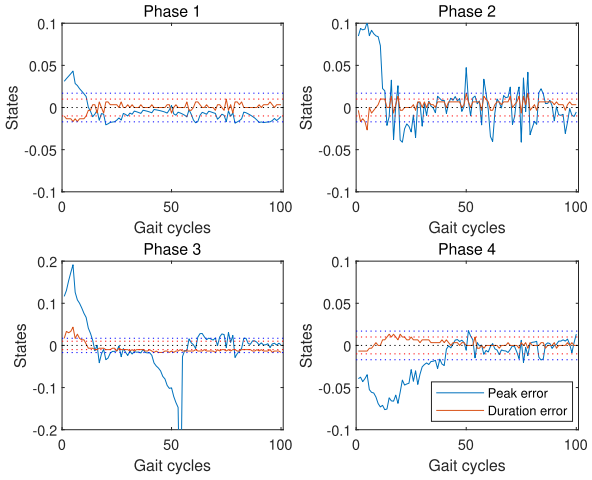


Fig. 7. Example trajectories of states during training in all four phases where the initial impedance parameters are 10% higher than the prescribed values. It is a result corresponding to the target task of normalized body weight at 0.08 kg.

0.12 kg. We perform ten runs for each target task with initial impedance parameters by adding uniform random noise with a 20% variance of the prescribed impedance parameters [32]. We record the average number of total samples (x_k, u_k, x_{k+1}) required to successfully complete each of the target tasks. Fig. 6 summarizes the average results from ten runs with randomly initialized network weights and initial impedance parameters for each algorithm setting. In the figure, naive actor-critic is based on [26] with identical network settings and other parameters (e.g. γ) compared to RL-KS. The pretrained actor-critic is obtained by training the naive actor-critic in a run with a body weight of 0.10 kg using 50 samples which is comparable to RL-KS. Fig. 6 shows that RL-KS has improved learning performance over both the naive and pretrained actor-critics with less number of training samples, except for the case when the body weight is 0.10 kg, which means the source task and the target task are identical. Fig. 7 is an example during training where the initial impedance parameters are 10% higher than the prescribed values. It is a result corresponding to the target task of body weight 0.08 kg. In Fig. 7, RL-KS has successfully reduced the errors in all four gait phases.

VIII. Conclusion

We have provided a set of new and comprehensive results on knowledge transfer within an actor-critic reinforcement learning construct. Our RL-KS has not only formalized a value function-based transfer learning for RL control framework but also provided multiple and flexible ways to represent prior knowledge. Additionally, our approach ensures positive knowledge transfer. Our analysis and experimental evaluation demonstrate the effectiveness of the proposed RL-KS. Strong results of this nature are not available in the current literature. Note that, our analysis is built on the introduction of a "task similarity" measured against a threshold factor τ. This notion has allowed us to develop a new positive transfer result. While Theorem 6 has provided a specification of τ, how to constructively and properly select τ, and how to maximize transfer efficiency remain to be investigated in the future. We speculate that this hyperparameter may be task dependent and thus be dictated by specific applications. Our RL-KS, however, has provided an important bridge to potentially addressing those synthesis questions.

References

[1] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jul. 2009.

[2] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning* (Adaptation, Learning, and Optimization), M. Wiering and M. Van Otterlo, Eds. Berlin, Germany: Springer, 2012, ch. 12, pp. 143–173.

[3] F. L. D. Silva and A. H. R. Costa, "A survey on transfer learning for multiagent reinforcement learning systems," *J. Artif. Intell. Res.*, vol. 64, pp. 645–703, Jan. 2019.

[4] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," 2020, *arXiv:2009.07888*.

[5] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2008, pp. 544–551.

[6] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *J. Mach. Learn. Res.*, vol. 8, no. 9, pp. 2125–2167, 2007.

[7] C.-A. Cheng, A. Kolobov, and A. Swaminathan, "Heuristic-guided reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 13550–13563.

[8] A. Barreto et al., "Successor features for transfer in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[9] A. Barreto et al., "Transfer in deep reinforcement learning using successor features and generalised policy improvement," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 501–510.

[10] M. Barekatain, R. Yonetani, and M. Hamaya, "MULTIPOLAR: Multi-source policy aggregation for transfer reinforcement learning between diverse environmental dynamics," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3108–3116.

[11] H. Yin and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proc. 31st AAAI Conf. Artif. Intell.*, vol. 31, 2017, pp. 1640–1646.

[12] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 278–287.

[13] E. Wiewiora, G. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 792–799.

[14] S. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proc. 11th Int. Conf. Auton. Agent. Multi Agent. Syst.*, Richland, SC, USA, 2012, pp. 433–440.

[15] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowe, "Expressing arbitrary reward functions as potential-based advice," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2652–2658.

[16] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Policy transfer using reward shaping," in *Proc. Int. Conf. Auton. Agent. Multi Agent. Syst.*, Richland, SC, USA, 2015, pp. 181–188.

[17] E. Wiewiora, "Potential-based shaping and Q-value initialization are equivalent," *J. Artif. Intell. Res.*, vol. 19, pp. 205–208, Sep. 2003.

[18] W. Sun, J. A. Bagnell, and B. Boots, "Truncated horizon policy search: Combining reinforcement learning and imitation learning," in *Proc. Int. Conf. Learn. Represent.*, Apr. 2018, pp. 1–14.

[19] S. Koenig and R. G. Simmons, "The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms," *Mach. Learn.*, vol. 22, nos. 1–3, pp. 227–250, Jan. 1996.

[20] F. L. Da Silva, G. Warnell, A. H. R. Costa, and P. Stone, "Agents teaching agents: A survey on inter-agent transfer learning," *Auto. Agents Multi-Agent Syst.*, vol. 34, no. 1, pp. 1–17, Apr. 2020.

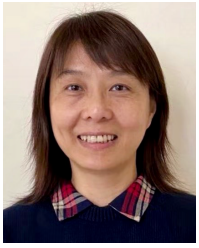
[21] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.

[22] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[23] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 779–789, Apr. 2013.
[24] Q. Wei, F.-Y. Wang, D. Liu, and X. Yang, "Finite-approximation-error-based discrete-time iterative adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2820–2833, Dec. 2014.
[25] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[26] J. Si and Y. T. Wang, "On-line learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.
[27] A. Ansari. (2013). *Windy-Grid-World-Q-Learning*. [Online]. Available: https://github.com/adilansari/Windy-Grid-World-Q-Learning
[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[29] S. L. Delp et al., "OpenSim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 11, pp. 1940–1950, Nov. 2007.
[30] X. Gao, J. Si, Y. Wen, M. Li, and H. Huang, "Reinforcement learning control of robotic knee with human-in-the-loop by flexible policy iteration," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 1–15, May 2021.
[31] M. Li, Y. Wen, X. Gao, J. Si, and H. Huang, "Toward expedited impedance tuning of a robotic prosthesis for personalized gait assistance by reinforcement learning control," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 407–420, Feb. 2022.
[32] Y. Wen, J. Si, X. Gao, S. Huang, and H. H. Huang, "A new powered lower limb prosthesis control framework based on adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 9, pp. 2215–2220, Sep. 2017.
[33] Y. Wen, J. Si, A. Brandt, X. Gao, and H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 1–11, Jan. 2019.
[34] X. Gao, J. Si, Y. Wen, M. Li, and H. Helen Huang, "Knowledge-guided reinforcement learning control for robotic lower limb prosthesis," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Aug. 2020, pp. 754–760.

Xiang Gao received the B.S. degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2011, and the M.S. and Ph.D. degrees in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2014 and 2020, respectively.

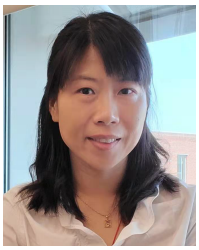
He is currently a Researcher with the Intelligent Robot Research Center, Jihua Laboratory, Foshan, Guangdong, China. His current research interests include robot learning, machine learning, computer vision, and rehabilitation robotics.



Jennie Si (Fellow, IEEE) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1985 and 1988, respectively, and the Ph.D. degree from the University of Notre Dame, Notre Dame, IN, USA, in 1992.

She has been a Faculty Member with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA, since 1991. She consulted for Intel, Arizona Public Service, and Medtronic, Phoenix, AZ, USA. She has served on several professional organizations' executive boards and international conference committees. She was an Advisor to the NSF Social Behavioral and Economical Directory. Her research interests focus on reinforcement learning control utilizing tools from optimal control theory, machine learning and neural networks. She is also interested in fundamental neuroscience of the frontal cortex and its role in decision and control processes.

Dr. Si was a recipient of the NSF/White House Presidential Faculty Fellow Award in 1995 and the Motorola Engineering Excellence Award in 1995. She is a Distinguished Lecturer of the IEEE Computational Intelligence Society. She served on several proposal review panels. She was a past Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and a past Action Editor of Neural Networks. She is a current Senior Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.



He (Helen) Huang (Fellow, IEEE) is a Jackson Family Distinguished Professor with the Joint Department of Biomedical Engineering, NC State University, Raleigh, NC, USA, and the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, and the Director for the Closed-Loop Engineering for Advanced Rehabilitation (CLEAR) Core. Her research interests lie in neural-machine interfaces, control of prosthetics and exoskeletons, human–robot interaction, and human movement control.

Prof. Huang is a fellow of AIMBE and a member of the Society for Neuroscience, AAAS, BMES, and ASB. She was the recipient of the Delsys Prize for Innovation in Electromyography, the Mary E. Switzer Fellowship with NIDRR (current NIDILRR), the NSF CAREER Award, the ASA SPES Award, and the ALCOA Foundation Distinguished Engineering Research Award.