Int. J. Fuzzy Syst.
https://doi.org/10.1007/s40815-023-01662-3

Enhancing Learning Efficiency in FACL: A Novel Fuzzy Rule Transfer Method for Transfer Learning

Dawei Ni[1] · **Howard M. Schwartz**[2]

Received: 19 July 2023 / Revised: 22 November 2023 / Accepted: 27 November 2023

Abstract The concept of leveraging knowledge from previous experience to accelerate learning forms the crux of transfer learning. Within the realm of reinforcement learning (RL), the agent typically requires protracted interaction with the environment, which can be time-consuming and can lead to slow convergence. Transfer learning offers a promising solution in such settings. In this paper, we investigate the application of transfer learning in the fuzzy reinforcement learning domain, specifically within the context of differential games. We introduce a novel approach for knowledge transfer across analogous tasks, employing fuzzy logic controllers as function approximators, notably within the Fuzzy Actor-Critic Learning (FACL) algorithm. Specifically, we propose a strategy for fuzzy rule transfer aimed at mapping fuzzy rules between the source and target tasks. The target task is assumed to be related to the source task yet it contains more complex states. Our approach has been implemented and tested within the domain of differential games in which all state space and action space are continuous. The simulation outcomes demonstrate that the application of knowledge transfer enables RL agents to learn faster and achieve asymptotic performance more rapidly in the target task.

Keywords Transfer learning · Fuzzy Actor-Critic Learning · Differential games · Reinforcement learning

✉ Dawei Ni
david_davidni@yahoo.ca

[1] Global Artificial Intelligence Accelerator, Ericsson, Montreal, Canada

[2] Department of System and Computer Engineering, Carleton University, Ottawa, Canada

1 Introduction

Reinforcement learning (RL) offers a structured approach for an agent to learn the mapping of situations to actions with the aim to maximize a numerical reward signal [1]. In RL, if both the state and action spaces are discrete and of modest scale, the policy can be conveniently stored in a look-up table. However, this approach becomes intractable when the state and action spaces occupy the continuous domain, implying an infinite array of states and actions to assess. Managing such an expansive look-up table becomes infeasible. To deal with this situation, function approximation is employed to represent the continuous state and action spaces in an effective way [2, 3]. In this study, we focus on one of the extensively utilized function approximators, namely, fuzzy logic controllers (FLC) or fuzzy inference systems (FIS). FLCs are universal approximators that can estimate any continuous function to any degree of accuracy [4]. By integrating RL with FLCs, we can address problems situated within the continuous domain. When reinforcement learning is associated with FLCs, the resultant approach is typically referred to as fuzzy reinforcement learning [5].

Fuzzy reinforcement learning is gaining popularity in both academic and applied sectors, including control for multi-robot system [6, 7] and autonomous underwater vehicle (AUV) [8]. Challenges arise when it comes to mastering intricate tasks. One notable challenge is the slow convergence towards a good policy, which can be attributed to several factors, including the complexity of the problem, the high dimensionality of the state and action spaces. For instance, temporal difference (TD) learning methods [1], a variant of RL, have demonstrated some success in different machine learning tasks, primarily due to its capability to learn in situations characterized by

Published online: 17 February 2024

limited prior knowledge and minimal environmental feedbacks [9]. However, TD frequently exhibits a rather slow convergence in practice to produce near-optimal behaviors. Thus, a large amount of research and techniques has been targeted at improving the learning speed of agents. Methods such as experience replay [10], reward shaping [11], and direct algorithmic improvements [12] are quite common. Notably, transfer learning stands out as a pivotal technique in this domain. The insight behind transfer learning is that generalizations may occur not only within tasks, but also across tasks [13]. This concept mirrors the mechanisms inherent in human learning processes, where we leverage our past experiences to efficiently navigate new tasks. Similarly, the method of learning can also benefit the RL agents, provided that they are given the appropriate experience from a similar task they have previously encountered. Within the context of reinforcement learning, Our aim is to transfer knowledge acquired by the agent from a preceding task (source task) to a new task (target task) to speed up learning.

To the best of our knowledge, this study is the first methodological exploration of transfer learning within the domain of fuzzy reinforcement learning. Specifically, We propose an innovative strategy for knowledge transfer across analogous tasks, utilizing the Fuzzy Actor-Critic Learning (FACL) algorithm at its core. Additionally, we introduce the concept of fuzzy rule transfer (FRT), specifically tailored for mapping of fuzzy rules between the source and target tasks. The method is based on the assumption that the target task, while related to the source task, encapsulates a more complex set of states. Our proposed methodology has been implemented and validated within the context of differential games, wherein all state and action spaces exist in a continuous domain. The outcomes of our research reveal that the application of FRT enables RL agents to achieve asymptotic performance at a notably accelerated rate within the target task.

The structure of the paper is organized as follows. The background and related work is presented in Sect. 2. We introduce the technical details of the Fuzzy Rule Transfer (FRT) method in Sect. 3. The simulation and results are demonstrated in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 Background and Related Work

2.1 Transfer Learning

Transfer learning operates on the insight that generalizations are not strictly confined within individual tasks but can potentially extend across tasks. This concept shares similar mechanisms in human learning processes, where we leverage our past experiences to effectively navigate new tasks. Analogously, the method of learning can also benefit the RL agents, given the appropriate experience from a similar task they have previously mastered. The most intuitive application of transfer learning to RL involves the reuse of solutions from tasks previously encountered by the agent [14]. However, many methodologies also emphasize the reuse of knowledge derived from external sources, such as demonstrations by human operators or advice from other learning agents [15]. Within the context of RL, we aim to transfer knowledge acquired by agents in a preceding task (source task) to a new task (target task) to speed up learning, and ultimately enhance the efficiency and effectiveness of RL agents. Transfer learning in RL tasks has been extensively explored through numerous studies [16, 17]. The wide array of transfer learning algorithms typically vary based on the degree of similarity between the target and source tasks [13]. The allowed task differences are a major consideration in RL transfer learning methods. To mimic a real-world situation, it is desirable to utilize methodologies that accommodate differences in state variables between the source and target task. An effective approach in this aspect is *transfer via inter-task mapping* (TVITM), which enables transfer between pairs of tasks by mapping the state variables and actions in the target task onto the source task [14]. In their research, various function approximators are investigated including the artificial neural network (ANN) and cerebellar model articulation controller (CMAC), and successfully employed within the TVITM framework. Another well-suited methodology for this scenario is *transfer via inter-task mapping for policy search methods* (TVITM-PS) [18]. In this approach, rather than transferring the action-value functions in TVITM, entire policies are transferred between the source and target tasks. Again, the transfer process within TVITM-PS is executed and evaluated utilizing a neural network function approximator. These approaches underscore the potential of transfer learning in the domain of RL tasks.

2.2 Evaluating Transfer Learning Methods

One of the key challenges in transfer learning is to define evaluation metrics. Most evaluation metrics are goal driven. For example, if the goal is reusing the knowledge from the past to accelerate training in a new task, then only a target task scenario should be considered. In this scenario, we focus on learning spent in the target task only, not the source task. This perspective is central to our study. The possible measurement options should always depend on the algorithms focus and the goal of transfer. We conclude the most common metrics as follows [13].

1. *Jumpstart*: This is the improvement in performance that occurs during the initial phase of a target task learning when transfer learning is applied.
2. *Asymptotic performance*: This is the improvement in performance that occurs at the final stage of a target task when transfer learning is applied.
3. *Total reward*: The total reward accumulated by an agent in the target task when transfer learning is applied. We compare the accumulated reward with and without transfer learning in the target task
4. *Transfer ratio*: This is similar to total reward, but instead of using it directly, we calculate the ratio of the total reward accumulated with and without the transfer in place.
5. *Time to threshold*: The elapsed time in the the target task for an agent to achieve a pre-specified performance level when transfer learning is applied.

While other metrics have been suggested in literature, we prioritize these five because they are sufficient to describe most transfer learning methods [13]. We present these performance metrics in Fig. 1, showcasing that various metrics can measure transfer learning. Specifically, the graph highlights enhancements in jumpstart, asymptotic performance, time to threshold, and total reward, represented by the area under the learning curve.

2.3 Fuzzy Actor-Critic Learning

When state and action spaces are either large or reside within the continuous domain, maintaining a lookup table typically becomes computationally infeasible. Consequently, traditional reinforcement learning algorithms, such as Q-learning, might lead to the curse of dimensionality. To overcome this issue, function approximators such as a cerebellar model arithmetic computer (CMAC) or an artificial neural network (ANN) can be used to effectively represent the continuous state and action spaces [1]. Another prevalent choice among function approximators is the Fuzzy Logic Controller (FLC) or Fuzzy Inference Systems (FIS) [2]. FLC are popular computing frameworks based on the concepts of fuzzy set theory, which have been extensively applied with success in a wide range of fields, including control, decision support, and system identification, among others [19]. The FLC exhibits two advanced attributes when employed for function approximation: Firstly, fuzzy sets are able to cope with the vagueness present in real-world environment and brings human heuristic knowledge into the control design, thereby creating an intuitive connection between the designer and the system. Subsequently, it provides the advantage of focusing the learning process strictly on the consequent parameters [20, 21]. This feature promotes a more efficient and targeted learning experience.

When RL is associated with the FLC, the approach is typically referred to as fuzzy reinforcement learning [5]. This approach encapsulates the benefits of both traditional RL and fuzzy logic, providing an effective means to address problems in areas where state and action spaces are continuous or otherwise complex. The fuzzy Q-learning (FQL) and FACL algorithm are amongst the most widely used algorithms in fuzzy reinforcement learning. A comparison between the FQL and the FACL algorithms was conducted in [20], where FACL leads to better performance and shorter learning length than those obtained with FQL. Given that FQL is a more compact method than FACL, the FACL's superiority may stem from its additional degrees of freedom relative to FQL [20]. Thus, in this study, we will primarily focus on the FACL algorithm.

In the FACL, the FLC is implemented as a function approximator for both actor and critic. The role of the actor is to generate continuous control action while the role of the critic is to predict the sum of future discounted rewards. Figure 2 illustrates the FACL agent and its environment to the system block. At every timestep, each learning agent observes the environment's state, $s_t \in S$. The actor, using this observation, chooses an optimal action $a_t \in A(s)$ Concurrently, the critic assesses anticipated performance through the value function relative to the agent's objectives. After this action, the agent receives a reinforcement

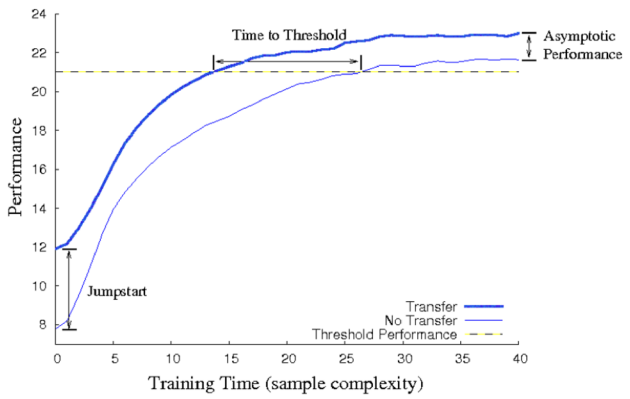


Fig. 1 Various metrics used in measuring TL [13]

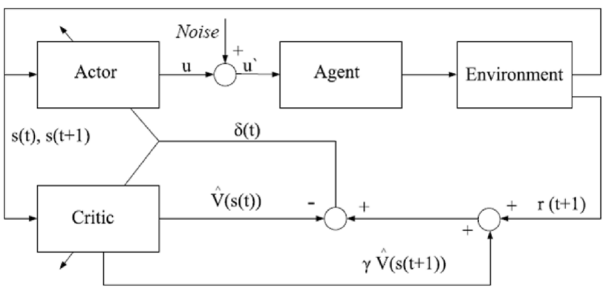


Fig. 2 The fuzzy actor-critic learning structure [27]

signal, $r_{t+1} \in R$ transitioning to a new state s_{t+1}. Using this reinforcement signal, the current value function, $\hat{V}_t$, and the subsequent state's value function, $\hat{V}_{t+1}$, the temporal difference (TD) error is computed. This error can be seen as an indication of prediction accuracy, is then used to adjust both the actor's and critic's inputs.

In this study, we have employed the zero-order Sugeno fuzzy model for the implementation of the FLC [22]. In our previous research, we have applied the fuzzy reinforcement learning methods to the pursuer-evader differential game in [23, 24]. In [23], a fuzzy actor-critic learning (FACL) algorithm is also applied to the pursuit-evasion differential game. It is shown that the adaptive fuzzy critic in [24] performed better than the neural network proposed in [25]. In the configuration proposed in this thesis, we only adapt the output parameters of the fuzzy system, whereas in [24, 26] the input and output parameters of the fuzzy system are adapted which is a more complex adaptive algorithm. In the subsequent section, we will briefly present the overview of FACL algorithms.

A RL agent interacts with the MDP environment and receives a reward signal r_t at each time step. The ultimate goal of an agent is to maximize the discounted return R_t in the long run [1]. The value function is the expected rewards at time step t, and can also be rewritten recursively as

$$V_t = r_{t+1} + \gamma V_{t+1}, \tag{1}$$

where $\gamma \in [0, 1)$ is a discount factor. The agent immediately receives a reward from the current environment based on its action in that state. We use a FLC as the agent's critic to estimate the value of the state at time t, as $\hat{V}(s(t))$. The error in the estimation of the state value is represented by the TD. As shown in Fig. 2, the TD error δ_t, is given as

$$\delta_t = r_{t+1} + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t). \tag{2}$$

We represent the actor by an adaptive fuzzy controller to effectively select an action and also update the policy parameter in the FLC in the direction suggested by the critic. The output of the fuzzy controller is,

$$u_t = \sum_{l=1}^{M} \Phi^l w_t^l, \tag{3}$$

where M is the number of rules, w^l is the output or consequent parameter of the actor, and Φ^l is the firing strength of rule l. The firing strength for rule l is defined as,

$$\Phi^l = \frac{\prod_{i=1}^{n} \mu^{F_i^l}(x_i)}{\sum_{l=1}^{M} (\prod_{i=1}^{n} \mu^{F_i^l}(x_i))}, \tag{4}$$

where n is the number of inputs and $\mu^{F_i^l}$ is the membership degree of input x_i, in the fuzzy rule F_i^l. We choose only triangular membership functions where only two membership functions overlap for any given input and the maximum value of any membership function is 1 [27]. Therefore the denominator of (4) is always equal to 1. For example, the set of membership functions for distance are as shown in Fig. 3. We can then rewrite (4) as [27],

$$\Phi^l = \prod_{i=1}^{n} \mu^{F_i^l}(x_i). \tag{5}$$

In order to promote exploration of the action space, a random white noise is chosen from a Gaussian distribution with a mean of 0 and a variance of σ given by $N(0, \sigma)$ is added to the generated control signal u. The output parameter of the actor FLC, w^l is adapted as [27]

$$w_{t+1}^l = w_t^l + \beta_L \delta_t \left(u_t' - u_t \right) \frac{\partial u}{\partial w^l}, \tag{6}$$

where δ_t is the TD error mentioned earlier, $\beta_L \in (0, 1)$ is the learning rate for the actor and where

$$\frac{\partial u}{\partial w^l} = \Phi_t^l. \tag{7}$$

Upon examining (6) we see that the term $u_t' - u_t$ is equal to the added noise. As such we modify the update for the actor FLC output parameters as,

$$w_{t+1}^l = w_t^l + \beta_L \delta_t (noise) \Phi_t^l. \tag{8}$$

Once the action is executed, the critic evaluates the new state to determine the expected new state value. The output of the critic $\hat{V}$ is an approximation to V given by

$$\hat{V}_t = \sum_{l=1}^{M} \Phi^l \zeta_t^l, \tag{9}$$

where ζ_t^l is the output or consequent parameter of the critic and Φ^l is defined in (4). The critic output parameter ζ^l is adapted as,

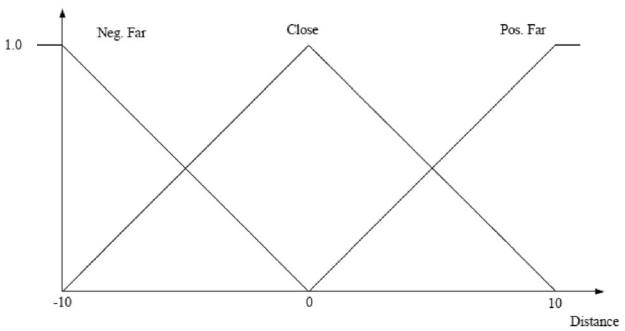


Fig. 3 Membership function for input of distance [27]

$$\zeta_{t+1}^{l} = \zeta_{t}^{l} + \alpha_L \delta_t \frac{\partial \hat{V}}{\partial \zeta^{l}}, \tag{10}$$

where $\alpha_L \in (0, 1)$ is the learning rate for the critic. The learning rate in FACL is set as $\beta_L < \alpha_L$, so that the actor will converge slower than the critic to prevent instability in the actor [28]. The partial derivative is calculated as,

$$\frac{\partial \hat{V}}{\partial \zeta^{l}} = \Phi_{t}^{l}. \tag{11}$$

We update the parameters in critic as

$$\zeta_{t+1}^{l} = \zeta_{t}^{l} + \alpha_L \delta_t \Phi^{l}. \tag{12}$$

The FACL learning algorithm is presented in Algorithm 1.

1: Initialize $\hat{V} = 0$, $\zeta^{l} = 0$ and $w^{l} = 0$ for $l = 1, ..., M$.
2: **for** each time step in the current episode **do**
3: Obtain the inputs $\bar{x}_t$.
4: Calculate the output of the actor u_t in (3).
5: Calculate the output of the critic $\hat{V}_t$ in (9).
6: Run the game for the current time step.
7: Obtain the reward r_{t+1} and new inputs $\bar{x}_{t+1}$.
8: Calculate $\hat{V}_{t+1}$ based on (9).
9: Calculate the temporal error δ_t in (2).
10: Update ζ_{t+1}^{l} in (12) and w_{t+1}^{l} in (8).
11: **end for**

Algorithm 1 FACL algorithm

3 Transfer Learning in FACL Using FRT

In the preceding section, we discussed the Fuzzy Actor-Critic Learning algorithm, a TD-based learning methodology, wherein both the actor and the critic are adapted in accordance with TDs. Additionally, we briefly explored transfer learning in RL, a strategy that speeds up an agent's learning process by applying knowledge from prior experiences to new tasks.

In this section, our focus is directed towards a specific scenario: knowledge transfer within the FACL algorithm. Drawing inspiration from Transfer via Inter-Task Mapping (TVITM) and Transfer via Inter-Task Mapping for Policy Search Methods (TVITM-PS), we propose an innovative FRT method. This method is specifically designed to map fuzzy rules between the source and target tasks, leveraging the unique FLC structure within the Fuzzy Actor-Critic Learning framework.

To illustrate the FRT method, we will initially discuss the overall approach and framework specific to FRT. Subsequently, we will dive into the details of the mapping process including the vectorization of fuzzy rules and the employment of similarity measurement. The latter is utilized to evaluate the degree of similarity between the vector representations of fuzzy rules.

3.1 Knowledge Transfer in Fuzzy Rules

We introduce a novel method for transferring knowledge between a previously learned task and a target task, specifically within the context of the FACL algorithm. This is achieved through the application of the FRT method.

For the FACL algorithm employed in this study, both the critic and actor are instantiated with zero-order Takagi–Sugeno–Kang (TSK) FLC that incorporates constant consequent parameters. Each FLC consists of L rules. The inputs for each rule comprise n fuzzy input variables, while the consequent of each rule is a numerical constant. Each rule $l (l = 1, \ldots, L)$ has the following form:

$$R_l: \text{ IF } s_1 \text{ is } A_1^l, \ldots, \text{ and } s_n \text{ is } A_n^l \text{ Then } z_l = k_l, \tag{13}$$

where the variable z_l represents the output variable of rule l, and k_l is the consequent parameter of rule l. The variable s_i $(i = 1, \ldots, n)$ is the ith input state variable of the fuzzy system, and n is the number of input state variables. The A_i^l represents the linguistic value of the input s_i at the rule l.

The total number of rules can be derived based on the number of input variables and membership functions (MF). We denote h_i as the number of membership functions for each state variable s_i. Given n input state variables, the total number of rules L can be calculated as (14).

$$L = h_1 \times h_2 \times h_3 \ldots \times h_n. \tag{14}$$

In this study, we consider a typical scenario where the input states are different in the source and target tasks. This implies the fuzzy rules associated with each task are distinct. As described previously, we use L_{source} as the total number of rules in the source task and L_{target} as the total number of rules in the target task. Thus the consequent parameter for source and target tasks are k_i $(i = 1, \ldots, L_{\text{source}})$ and k_j $(j = 1, \ldots, L_{\text{target}})$ respectively. We consider L_{target} to be much larger than L_{source} as target tasks are assumed to have more complex states than the source tasks.

Our objective is to facilitate knowledge transfer between the source and target tasks. In the context of the FACL, the acquired knowledge is encapsulated in the consequent parameters of the Fuzzy Logic Controller (FLC). Given the difference in rule quantity and rule structure previously explained, we introduce a mapping function, denoted as ψ,

designed to align the initial consequent parameters in a target task with the learned consequent parameters in a source task, based on rule similarity. This initiation of the FLC within a target task, utilizing previously acquired knowledge, enables an immediate enhancement in both the state estimation of the critic and the action selection of the actor during the initial learning phase. This improvement navigates the learning agent towards more reasonable action choices, as opposed to those selected at random, thus accelerating the learning process.

Similar to the mapping procedures outlined in TVITM and TVITM-PS, as described in Sect. 2, we perform mapping functional ψ on both the critic and the actor of the agent in FACL. However, a fundamental difference is that the mappings in TVITM and TVITM-PS are performed on the state variables and actions, whereas in the FLC function approximator, the mapping is directly performed on the fuzzy rules space between the source and target task. The consequent parameters of fuzzy rules pertaining to a target task's critic FLC are mapped to the learned consequent parameters of the most similar rules pertaining to a source task's critic FLC, via ψ_{critic} in (15)

$$\psi_{\text{critic}}(k^{\text{critic}}_{i,\text{target}}) = k^{\text{critic}}_{j,\text{source}}, \tag{15}$$

where $k^{\text{critic}}{}_{i,\text{target}}$ $(i = 1, \ldots, L_{\text{target}})$ is the consequent parameter of the critic's FLC in a target task and $k^{\text{critic}}{}_{j,\text{source}}$ $(j = 1, \ldots, L_{\text{source}})$ is the consequent parameter of the critic's FLC in a source task. Furthermore, we employ ψ_{actor} to establish a mapping between the consequent parameters of fuzzy rules in the actor's FLC within a target task and the corresponding learned parameters of the most similar rules in the actor's FLC from a source task, as given in (16)

$$\psi_{\text{actor}}(k^{\text{actor}}_{i,\text{target}}) = k^{\text{actor}}_{j,\text{source}}, \tag{16}$$

where $k^{\text{actor}}{}_{i,\text{target}}$ $(i = 1, \ldots, L_{\text{source}})$ is the consequent parameter of the actor's FLC in a target task and $k^{\text{actor}}{}_{j,\text{source}}$ $(j = 1, \ldots, L_{\text{target}})$ is the consequent parameter of the actor's FLC in a source task. As can be seen, both mappings are performed based on the similarity of the fuzzy rules between the source and target tasks. We will discuss the mapping process in the following subsections.

One prerequisite for our transfer method to succeed is that the source and target task must be related or similar to some degree. Transfer learning simply would not work on a pair of arbitrary tasks. In the context of FACL, this means there is at least one common state variable, and its MFs are shared between the source and target task. As described in Equations (15) and (16), we execute the mapping procedure of the consequent parameters associated with fuzzy rules in the target task, aligning them with the corresponding parameters in the most similar fuzzy rules drawn from the source tasks. This mapping process consists of two components: (1) the vectorization of fuzzy rules, which is critical in transforming the complex structure of these rules into a computationally tractable form; (2) the employment of similarity measurement for evaluating the degree of similarity between the vector representations of fuzzy rules. The complete FRT procedure is outlined in Algorithm 2.

1: Train the FACL agents in the source task. Identify the fuzzy rules associated with the state variables and MFs.
2: Configure the FACL agents in the target task:
   1. To facilitate knowledge transfer, reuse the same state variables and MFs from the source task, introducing new state variables and MFs as necessary.
   2. Determine the fuzzy rules associated with the new state variables and MFs.
3: **for** each fuzzy rule R_i in critic's FLC in the target task **do**
   1. Identify the most similar fuzzy rule R_j in the source task according to (23)
   2. Map consequent parameters in critic: $\psi_{critic}(k^{\,critic}_{i,target}) = k^{\,critic}_{j,source}$ in (15)
4: **end for**
5: **for** each fuzzy rule R_i in actor's FLC in the target task **do**
   1. Identify the most similar fuzzy rule R_j in the source task according to (23)
   2. Map consequent parameters in actor: $\psi_{actor}(k^{\,actor}_{i,target}) = k^{\,actor}_{j,source}$ in (16)
6: **end for**

Algorithm 2 Fuzzy rule transfer (FRT) procedure

3.2 Vectorization of Fuzzy Rules

Similar to (13), consider a given fuzzy rule defined in the following form,

$$R_l : \text{IF } s_1 \;\; is \;\; A^{s_1}_m, \ldots, \text{ and } \;\; s_n \text{ is } A^{s_n}_m \text{ Then } z_l = k_l. \tag{17}$$

Let $\mathcal{S}$ represent the set of state variables $\{s_1, s_2, s_3, \ldots, s_n\}$, $\mathcal{A}^{s_n}$ represent the set of all membership functions $\{A^{s_n}_0, A^{s_n}_1, A^{s_n}_2, \ldots, A^{s_n}_m\}$ associated with state variable s_n, and m denotes the zero-based index for the membership function in the set.

We can map each fuzzy rule into a vector $\mathbf{v}$ of length $|\mathcal{S}|$, where each element v_i in $\mathbf{v}$ is the unique index m of the membership function in $\mathcal{A}^{s_n}$.

Under these assumptions, the vector form [0, 1, 2] corresponds to a rule "IF s_1 is $A_0^{s_1}$ and s_2 is $A_1^{s_2}$ and s_3 is $A_2^{s_3}$ Then..."

3.3 Similarity Measurement of Fuzzy Rules

Suppose we have two fuzzy rules R1 and R2. R1 is from the source task and with two input state variables,

$$R_1 : \text{IF } s_1 \text{ is } A_1^{s_1} \text{ and } s_2 \text{ is } A_2^{s_2} \text{ Then } z_1 = k_1. \tag{18}$$

R2 is from the target task and we assume the target task is more complex than the source task. Thus, R2 consists of three input state variables,

$$\begin{aligned} R_2 :& IF \ \ s_1 \ \ is \ \ A_1^{s_1} \ \ and \ \ s_2 \ \ is \ \ A_2^{s_2} \ \ and \ \ s_3 \ \ is \ \ A_1^{s_3} \\ & \text{Then } \ z_2 = k_2. \end{aligned} \tag{19}$$

after applying vectorization to fuzzy rule R1 and R2, we can derive the following vector forms,

$$\begin{aligned} \mathbf{V}_{R_1} &= [1,\ 2] \\ \mathbf{V}_{R_2} &= [1,\ 2,\ 1]. \end{aligned} \tag{20}$$

As mentioned previously, one prerequisite in the FACL context is that there should be one or more common state variables between the source and target task, and we only consider the dimension of vectors that correspond to the shared state variables. To this end, we define a transformation process as a function T,

$$\mathbf{v}'_1,\ \mathbf{v}'_2 = \mathrm{T}(\mathbf{v}_1,\ \mathbf{v}_2), \tag{21}$$

and the function T operates on the original vectors $\mathbf{v}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{v}_2 \in \mathbb{R}^{n_2}$ which correspond to distinct sets of state variables $\mathbf{s}_1 = [s_{1,1}, s_{1,2}, \ldots, s_{1,n_1}]$ and $\mathbf{s}_2 = [s_{2,1}, s_{2,2}, \ldots, s_{2,n_2}]$, respectively. We denote the intersection of the state variables as $\mathbf{s}_\cap = [s_{\cap,1}, s_{\cap,2}, \ldots, s_{\cap,k}]$, where k is the number of shared state variables. The goal is to reorganize the vectors based on the shared state variables, $\mathbf{s}_\cap$. To this end, we map the elements of $\mathbf{v}_1$ and $\mathbf{v}_2$ corresponding to $\mathbf{s}_\cap$, obtaining the transformed vectors $\mathbf{v}'_1$ and $\mathbf{v}'_2$. Specifically, $\mathbf{v}'_1 = [v_{1,\cap,1}, v_{1,\cap,2}, \ldots, v_{1,\cap,k}]$ and $\mathbf{v}'_2 = [v_{2,\cap,1}, v_{2,\cap,2}, \ldots, v_{2,\cap,k}]$, where $v_{1,\cap,j}$ and $v_{2,\cap,j}$ are the elements of $\mathbf{v}_1$ and $\mathbf{v}_2$ corresponding to the shared state variable $s_{\cap,j}$, $j = 1, 2, \ldots, k$. The resulting $\mathbf{v}'_1$ and $\mathbf{v}'_2$ should share the same dimension. We apply the aforementioned transformation (21) on the vectors in (20), and derive $\mathbf{V}'_{R_1}$ and $\mathbf{V}'_{R_2}$,

$$\begin{aligned} \mathbf{V}'_{R_1} &= [1,\ 2] \\ \mathbf{V}'_{R_2} &= [1,\ 2], \end{aligned} \tag{22}$$

We propose a method to measure the similarity between fuzzy rules based on the representation of these rules in vector form. The underlying hypothesis is that two fuzzy rules, R_1 and R_2, are most similar if the similarity measurement of their vector forms is 1.

Let V_1 and V_2 be the vector forms of the fuzzy rules R_1 and R_2, respectively, and suppose they possess the same dimension. We quantify their similarity $S(V_1, V_2)$ by computing the euclidean distance between V_1 and V_2, and subsequently normalized to lie within the interval [0, 1]. The formal definition of the similarity measure is:

$$S(V_1, V_2) = \frac{1}{1 + d(V_1, V_2)} = \frac{1}{1 + \sqrt{\sum_{i=1}^{n} (v_{1,i} - v_{2,i})^2}}, \tag{23}$$

Where d is the euclidean distance, $v_{1,i}$ and $v_{2,i}$ represent the components of vectors V_1 and V_2, respectively, and n is the dimensionality of the vectors. In essence, the closer the value of $S(V_1, V_2)$ is to 1, the more similar the two fuzzy rules are deemed to be.

For the given example, we perform the similarity computation using (23) on V'_{R_1} and V'_{R_2},

$$S(V'_{R_1}, V'_{R_2}) = \frac{1}{1 + d(V'_{R_1}, V'_{R_2})} = 1. \tag{24}$$

This means R_2 in the target task is most similar to the source task's pre-trained fuzzy rule R_1. Hence the mapping should take place, as shown in (15) and (16), on their consequent parameters. Hence, the consequent parameter k_2 is mapped to the pre-trained value of k_1,

$$\begin{aligned} \psi_{\text{actor}}(k_2) &= k_1 \\ \psi_{\text{critic}}(k_2) &= k_1. \end{aligned} \tag{25}$$

3.4 FRT: Systematic Illustration, Complexity and Example

In Fig. 4, we present the system architecture illustrating the FRT process. Within this framework, FRT is applied to both critic and actor components of the FACL agents. As shown in the figure,the fuzzy rules 3 and 4 from the target FLC are mapped to fuzzy rules 1 and 2 from the source task, drawing upon the similarity metrics discussed in Sect. 3.3. Thus, the consequent parameter k'_3 (associated with fuzzy rule 3) takes the pretrained value of the consequent parameter k_1 from the source FLC, and similarly, k'_4 (associated with fuzzy rule 4) adopts the pretrained value of k_2 from the source FLC.

In the context of the FRT application, it is worth noting that the FRT algorithm only needs to run once at the

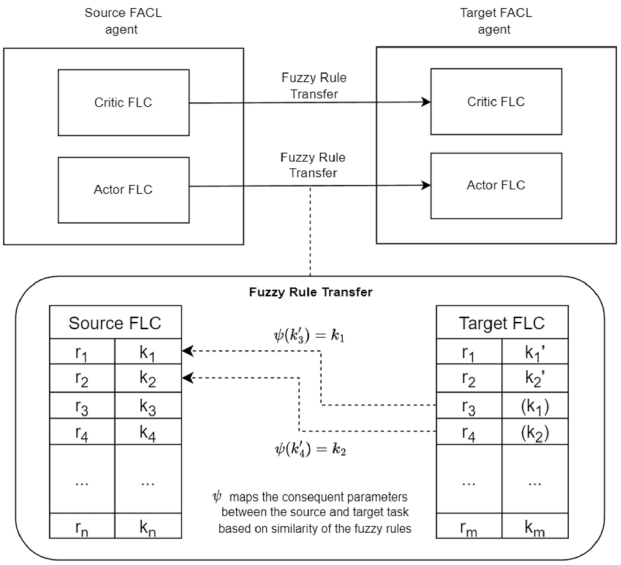


Fig. 4 FRT system diagram showing how ψ maps the consequent parameters between the source and target task based on similarity of the fuzzy rules

beginning of training for the target task. Under the most extreme circumstances, the computational complexity of the FRT could be denoted as $L_{\text{source}} \times L_{\text{target}}$, where L_{source} is the total number of rules in the source task and L_{target} represents those of the target task. However, empirical observations suggest that the real-world complexity is often substantially less, due to the sparsity of the consequent parameters in the source FLC; a significant majority of these parameters are zero, with only a small fraction having non-zero values that need to be transferred. Consequently, the real computational cost is likely much less than the maximum estimated in the most extreme circumstances. Furthermore, storing fuzzy rules in a structured way can help save computation time. For instance, by arranging similar target rules together in memory, we can more efficiently transfer data between tasks, e.g. bulk memory transfers. However, it is worth noting that the exact software design can affect this. For the time being, discussing those details is beyond the scope of this study.

For enhanced clarity, we present an illustrative example. Consider a basic model of an autonomous driving agent. In the source task, the primary objective is training the agent to circumvent pedestrians. Given only two pedestrians, we account for two state variables, representing the distances between the agent and each pedestrian. The target task, however, introduces added complexity. Here, the agent is trained to avoid pedestrians while also navigating to a specified destination. In such a way, a third state variable is created. A detailed representation of this scenario is provided in the subsequent Table 1.

Table 1 State variable in the source task and target task

State variable	Description
Source task	
s_1	Distance between agent and pedestrian 1
s_2	Distance between agent and pedestrian 2
Target task	
s_1	Distance between agent and pedestrian 1
s_2	Distance between agent and pedestrian 2
s_3	Distance between agent and destination

Recall the rule form of a zero-order TSK Fuzzy Logic Controller (13), the fuzzy rule space in the source tasks are shown as follow:

$$\begin{aligned}
R_1 &: \text{IF } s_1 \text{ is } A_1 \text{ and } s_2 \text{ is } B_1 \text{ Then } z_1 = k_1^{\text{source}} \\
R_2 &: \text{IF } s_1 \text{ is } A_2 \text{ and } s_2 \text{ is } B_1 \text{ Then } z_2 = k_2^{\text{source}} \\
R_3 &: \text{IF } s_1 \text{ is } A_1 \text{ and } s_2 \text{ is } B_2 \text{ Then } z_3 = k_3^{\text{source}} \\
R_4 &: \text{IF } s_1 \text{ is } A_2 \text{ and } s_2 \text{ is } B_2 \text{ Then } z_4 = k_4^{\text{source}} .
\end{aligned} \tag{26}$$

In our illustrative example, each input utilizes two membership functions (MFs). Specifically, for state variable s_1, we have MFs A_1 and A_2, and for s_2, MFs B_1 and B_2 are used. Similarly, in the target task we introduce a new state variable s_3 with two MFs, C_1 and C_2. The fuzzy rule space in the target task is shown in below:

$$\begin{aligned}
&R_1' : \text{IF } s_1 \text{ is } A_1 \text{ and } s_2 \text{ is } B_1 \text{ and } s_3 \text{ is } C_1 \\
&\quad \text{Then } z_1 = k_1^{\text{target}} \\
&R_2' : \text{IF } s_1 \text{ is } A_2 \text{ and } s_2 \text{ is } B_1 \text{ and } s_3 \text{ is } C_1 \\
&\quad \text{Then } z_2 = k_2^{\text{target}} \\
&R_3' : \text{IF } s_1 \text{ is } A_1 \text{ and } s_2 \text{ is } B_2 \text{ and } s_3 \text{ is } C_1 \\
&\quad \text{Then } z_3 = k_3^{\text{target}} \\
&R_4' : \text{IF } s_1 \text{ is } A_2 \text{ and } s_2 \text{ is } B_2 \text{ and } s_3 \text{ is } C_1 \\
&\quad \text{Then } z_4 = k_4^{\text{target}} \\
&R_5' : \text{IF } s_1 \text{ is } A_1 \text{ and } s_2 \text{ is } B_1 \text{ and } s_3 \text{ is } C_2 \\
&\quad \text{Then } z_5 = k_5^{\text{target}} \\
&R_6' : \text{IF } s_1 \text{ is } A_2 \text{ and } s_2 \text{ is } B_1 \text{ and } s_3 \text{ is } C_2 \\
&\quad \text{Then } z_6 = k_6^{\text{target}} \\
&R_7' : \text{IF } s_1 \text{ is } A_1 \text{ and } s_2 \text{ is } B_2 \text{ and } s_3 \text{ is } C_2 \\
&\quad \text{Then } z_7 = k_7^{\text{target}} \\
&R_8' : \text{IF } s_1 \text{ is } A_2 \text{ and } s_2 \text{ is } B_2 \text{ and } s_3 \text{ is } C_2 \\
&\quad \text{Then } z_8 = k_8^{\text{target}} .
\end{aligned} \tag{27}$$

Apply the vectorization described in Sect. 3.2 and transformation process in (21), the vector form of the fuzzy rules in the source tasks now become:

$$
\begin{aligned}
V_{R_1} &= [1,\ 1] \\
V_{R_2} &= [2,\ 1] \\
V_{R_3} &= [1,\ 2] \\
V_{R_4} &= [2,\ 2].
\end{aligned} \tag{28}
$$

For the vector form of the fuzzy rules in the target tasks, as s_1 and s_2 are the only shared state variables between source and target task, we remove the s_3's dimension from the vector according to (21). The derived vector form are as follows:

$$
\begin{aligned}
V_{R'_1} &= [1,\ 1] \\
V_{R'_2} &= [2,\ 1] \\
V_{R'_3} &= [1,\ 2] \\
V_{R'_4} &= [2,\ 2] \\
V_{R'_5} &= [1,\ 1] \\
V_{R'_6} &= [2,\ 1] \\
V_{R'_7} &= [1,\ 2] \\
V_{R'_8} &= [2,\ 2].
\end{aligned} \tag{29}
$$

Using the similarity computation from (23), we can measure the similarity between the fuzzy rules. for instance, if we compute the similarity between R'_1 and all the fuzzy rules in the source task, we derive the following:

$$
\begin{aligned}
S(V_{R'_1}, V_{R_1}) &= 1 \\
S(V_{R'_1}, V_{R_2}) &= 0.5 \\
S(V_{R'_1}, V_{R_3}) &= 0.5 \\
S(V_{R'_1}, V_{R_4}) &= 0.414.
\end{aligned} \tag{30}
$$

We can clearly see R_1 is the most similar rule to R'_1 in the source task. Hence, the mapping of the consequent parameter takes place, as described in (15) and (16):

$$
\psi(k_1^{\text{target}}) = k_1^{\text{source}}. \tag{31}
$$

The core idea is that even with the expanded rule space from additional state variables, there is still foundational knowledge relies on the source tasks' rules. Essentially, the shared state variables form the connection of common knowledge, enabling knowledge transfer between source and target tasks. We apply this approach to all fuzzy rules in our example, and Table 2 provides a detailed mapping.

Table 2 Mapping of the consequent parameters in FRT

Consequent parameters mapping ψ	
Target task	Source task
k_1^{target}	k_1^{source}
k_2^{target}	k_2^{source}
k_3^{target}	k_3^{source}
k_4^{target}	k_4^{source}
k_5^{target}	k_1^{source}
k_6^{target}	k_2^{source}
k_7^{target}	k_3^{source}
k_8^{target}	k_4^{source}

4 Simulation and Result

In this section, we assess the proposed algorithm in two distinct sets of differential games; the pursuit-evasion game and the guarding-a-territory game. All the simulation codes were designed and implemented from scratch in Python, due to the absence of an existing open-source library for the FACL algorithm and its corresponding simulation environment at the time of this study. Within the Pursuit-Evasion Game, the objective of the evader is to identify the fastest escape route, while the pursuer endeavors to capture the evader. We modeled the Pursuit-Evasion Game under two scenarios: single-agent learning and multi-agent learning. In the single-agent learning scenario, only the evader is subjected to the learning process, while the pursuer employs a simple control strategy, moving constantly in the direction of line-of-sight towards the evader. In contrast, within the multi-agent learning scenario, both the evader and pursuer are learning using FACL. Conversely, for the Guarding-a-Territory Game, we solely consider the multi-agent learning scenario, given the nature of the game. Within this setting, both guards and the invader learn to identify their optimal strategies. The guards aim to intercept the invader at the greatest possible distance from the territory, while the invader seeks to approach as close as possible to the territory without being captured. This game begins with a source task featuring a single invader versus a single guard and subsequently extends to a target task involving one invader versus two guards.

4.1 Pursuit-Evasion Game

The Pursuit-Evasion game is a form of differential game involving a high-velocity evader and multiple slower pursuers. The objective of the pursuers is to apprehend the evader, while the evader's primary goal is to accomplish a rapid escape from the pursuers. Initially, the agents are engaged in the game within a configuration of one evader versus one pursuer, serving as the source task. Subsequently, the game's complexity is increased in the target task, featuring an expanded scenario of one evader engaged

with three pursuers. The agent kinematics is defined as (32),

$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega, \end{aligned} \tag{32}$$

where v represents the agent speed, ω is the angular speed of the agent, θ is the orientation of the agent with respect to x axis, x and y are the position of the agent on the global x and y axis. In the design of our FACL agent, ω is the output of the actor's FLC. The kinematics equations of the players given above are solved in simulation using the second-order Runge-Kutta method.

4.1.1 Single-Agent Pursuit-Evasion Game: Setup

In the single-agent model, the pursuer employs a simple control strategy which always moves in the direction of the evader, while the evader is learning its optimal strategy using the FACL. Initially, we start our training in a source task featuring a single evader versus a single pursuer. The speed of the pursuer is set at 1 unit/second and the evader is set at 1.2 units/second. Thus, the evader should always be able to escape. The game is lost when the evader is captured and won when the Euclidean distance between the evader and pursuers becomes larger than 15 units. This means the evader has escaped from the pursuer. To enhance the efficiency of the agent's learning process, the introduction of immediate rewards within the environment could be beneficial [29, 30]. We use a shaping reward function for the high-speed evader and it has two components, given as (33),

$$R_{t+1} = d_{\text{ep}}(t+1) - d_{\text{ep}}(t) - p, \tag{33}$$

where d_{ep} is the Euclidean distance between the evader and the pursuer. The component $d_{\text{ep}}(t+1) - d_{\text{ep}}(t)$ provides a positive reward if the evader moves farther from the pursuer and a negative reward otherwise. The other component p is a constant and it appends a penalty to the reward at every time step and pushes the agent to escape as soon as possible. The value of p is chosen based on the grid search between 0 and 1 and in this case we found the evader agent performs the best when p is set to 0.2.

Our source task utilizes three input state variables within the FLCs, as outlined in Table 3. Triangular membership functions are employed to define all fuzzy sets across the state variables. Ten symmetrical and uniformly spread triangular membership functions are defined for the heading angle of the agent θ_{hd}, over the interval of [-π, π]. Meanwhile, seven triangular membership functions were defined for each the Manhattan distance variable d, over the interval of [− 40, 40]. Additionally, We set the initial learning rates of the actor and the critic in the FACL algorithm as follows: $\beta_0 = 0.05$ and $\alpha_0 = 0.1$. We set $\beta_0 < \alpha_0$ so that the actor will converge slower than critic to prevent instability in the actor [28]. The variance of the initial exploration noise σ and the discount factor γ are set as follows: $\sigma = 3$ rad/second and $\gamma = 0.25$. Finally, the decay rates of the learning factors and exploration noise after each episode k are set as follows: $\beta_{k+1} = 0.995^k\beta_0$, $\alpha_{k+1} = 0.999^k\alpha_0$, and $\sigma_{k+1} = 0.994^k\sigma_0$. The values for the learning rates, the exploration noise, and the discount factors were carefully chosen based on a priori knowledge of the problem and heuristics from the experiments. These configurations have demonstrated effectiveness in our simulation. To ensure an appropriate comparison when evaluating the FRT, these configurations are set consistently in the target task throughout the simulation.

Firstly, we set the individual pursuer and evader to engage in the game for a total of 1000 episodes, the final trajectory of which can be observed in Fig. 5. The result shows that the evader agent was successful in determining the fastest escape path, specifically by moving as a linear trajectory in the opposite direction of the pursuer. For the target task, our agent is engaged in a game exhibiting strong similarities to our source task, with the key distinction being that the three pursuers now start from varying positions surrounding the evader, as depicted in Fig. 6. All three pursuer agents are identical, following the same simple control strategy applied in the source task. However, the state space for the evader is altered due to the presence of additional pursuers. Instead of maintaining three state variables as in the source task previously, in the target task, we incorporate seven state variables into the FLCs. The full list of state variables present in the target task is described in Table 4.

Meanwhile, our reward function is also expanded to account for the distance between the evader and the additional pursuers, given as (34),

$$\begin{aligned} R_{t+1} = \; & d_{\text{ep1}}(t+1) - d_{\text{ep1}}(t) + d_{\text{ep2}}(t+1) \\ & - d_{\text{ep2}}(t) + d_{\text{ep3}}(t+1) - d_{\text{ep3}}(t) - p, \end{aligned} \tag{34}$$

where d_{ep1} is the Euclidean distance between the evader and pursuer 1, d_{ep2} is the Euclidean distance between the evader and pursuer 2 and so forth. Similar to the 1 evader vs. 1 pursuer case, we introduce a constant p in the reward function to push the evader agent to escape as soon as possible. As the number of pursuers in the target task tripled, we raised the value of parameter p to 0.6, leading to favorable performance outcomes in our trials.

Furthermore, owing to the rise in the number of state variables in the target task, there is a corresponding expansion in the fuzzy rule space, which now has

Table 3 Agents' state variables in 1 evader vs. 1 pursuer

Source task: 1 evader vs. 1 pursuer		
State variable	Description	MFs and range
d_{ep}^{x}	Manhattan distance w.r.t. x-axis between evader and pursuer	7 uniform triangular MFs over the range of [− 40, 40]
d_{ep}^{y}	Manhattan distance w.r.t. y-axis between evader and pursuer	7 uniform triangular MFs over the range of [− 40, 40]
θ_{hd}	Heading angle of the agent w.r.t. x-axis	10 uniform triangular MFs over the range of [-π, π]

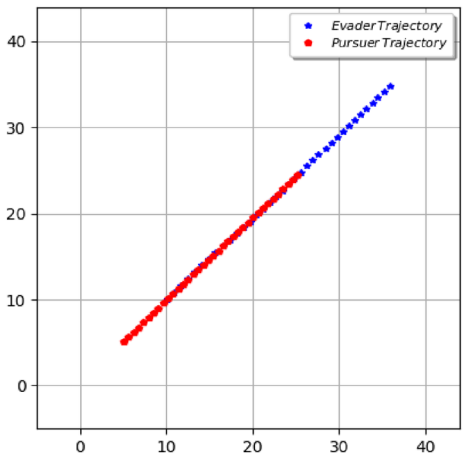


Fig. 5 Game trajectory 1 evader vs. 1 pursuer with only the pursuer is learning. The evader learned to escape

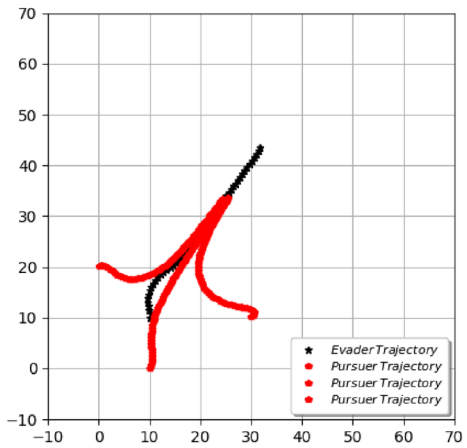


Fig. 6 Game trajectory 1 evader vs. 3 pursuer with only the pursuer is learning. The evader learned to escape

1,176,490 rules as opposed to the 490 rules contained within the source task. The aggregate number of rules is computed based on the number of Membership Functions (MFs) and state variables as shown in (14). Although the target task comprises a substantial number of rules, the employment of triangular membership functions in our design ensures that, at any given time step, only 2^n rules are fired, where n denotes the number of inputs [27]. Accordingly, our objective is solely to pinpoint the sets of rules activated based on the input state variables, followed by the computation of the firing strength for the rules within these rule sets. This renders the process computationally feasible. However, given the seven input state variables, we must compute 256 rules collectively for both the critic and actor within the FACL at each time step. This presents a computationally intensive process and the agent may not converge towards an optimal policy in a brief period. Nevertheless, if we adopt a different type of membership function such as the Gaussian membership function, the computational complexity will surge exponentially, albeit the transfer process remains unaffected since the number of fuzzy rules and consequent parameters remain unchanged.

The process of FRT, as outlined in Sect. 3 and Algorithm 2, is implemented in the FACL agent within the target task. This procedure involves the mapping of fuzzy rules on both the critic and the actor of the FACL agent, as presented in (15) and (16), through which the initial value of the consequent parameters in the target task is established based on the learned consequent parameters in the source task and the degree of similarity between the fuzzy rules, as described in Sect. 3.3. In our simulation software, this mapping can be accomplished via a simple *For* loop that traverses the fuzzy rules space in the target task to allocate weights to the consequent parameters, conditioned on the similarity measurement. This process needs to be executed only once at the beginning of the simulation in the target task. To enhance understanding, we provide an illustrative example of how the mapping and assignment are executed for a single fuzzy rule. Suppose we have a fuzzy rule R_m in the target task,

$$\begin{aligned} R_m : \text{IF } & d_{\text{ep1}}^{x} \text{ is } MF_{d_{\text{ep1}}^{x}}^{1} \text{ and } d_{\text{ep1}}^{y} \text{ is } MF_{d_{\text{ep1}}^{y}}^{1} \\ & \text{and } d_{\text{ep2}}^{x} \text{ is } MF_{d_{\text{ep2}}^{x}}^{1} \text{ and } d_{\text{ep2}}^{y} \text{ is } MF_{d_{\text{ep2}}^{y}}^{1} \\ & \text{and } d_{\text{ep3}}^{x} \text{ is } MF_{d_{\text{ep3}}^{x}}^{1} \text{ and } d_{\text{ep3}}^{y} \text{ is } MF_{d_{\text{ep3}}^{y}}^{1} \\ & \text{and } \theta_{\text{hd}} \text{ is } MF_{\theta_{\text{hd}}}^{1} \text{ Then } z_m = k_m^{\text{target}}, \end{aligned} \tag{35}$$

where d_{ep1}^{x} to θ_{hd} are the input state variables presented in Table 4. $MF_{d_{\text{ep1}}^{x}}^{1}$ is the first membership function of the state

Table 4 Evader's state variables in 1 evader vs. 3 pursuer

Target task: 1 evader vs. 3 pursuer		
State variable	Description	MFs and range
d^x_{ep1}	Manhattan distance w.r.t. x-axis between evader and pursuer 1	7 uniform triangular MFs over the range of [− 40, 40]
d^y_{ep1}	Manhattan distance w.r.t. y-axis between evader and pursuer 1	7 uniform triangular MFs over the range of [− 40, 40]
d^x_{ep2}	Manhattan distance w.r.t. x-axis between evader and pursuer 2	7 uniform triangular MFs over the range of [− 40, 40]
d^y_{ep2}	Manhattan distance w.r.t. y-axis between evader and pursuer 2	7 uniform triangular MFs over the range of [− 40, 40]
d^x_{ep3}	Manhattan distance w.r.t. x-axis between evader and pursuer 3	7 uniform triangular MFs over the range of [− 40, 40]
d^y_{ep3}	Manhattan distance w.r.t. y-axis between evader and pursuer 3	7 uniform triangular MFs over the range of [− 40, 40]
θ_{hd}	Heading angle of the agent w.r.t. x-axis	10 uniform triangular MFs over the range of [-π, π]

variable d^x_{ep1}, $MF^1_{d^y_{\text{ep1}}}$ is the first membership function of the state variable d^y_{ep1}, and so on. Based on the similarity measurement of fuzzy Rules presented in Sect. 3.3, we map this rule to the most similar rule R_n in the source task,

$$R_n: \text{ IF } d^x_{\text{ep1}} \text{ is } \text{MF}^1_{d^x_{\text{ep1}}} \text{ and } d^y_{\text{ep1}} \text{ is } \text{MF}^1_{d^y_{\text{ep1}}} \text{ and } \theta_{\text{hd}} \text{ is } \text{MF}^1_{\theta_{\text{hd}}} \text{ Then } z_n = k_n^{\text{source}}, \tag{36}$$

where k_n^{source} is the learned value of the consequent parameter in the source task. Thus, the mapping should take place, as shown in (15) and (16), on their consequent parameters. Particularly, for both actor and critic, the consequent parameter k_m^{target} is mapped to the pre-trained value of k_n^{source},

$$\begin{aligned} \psi_{\text{actor}}(k_m^{\text{target}}) &= k_n^{\text{source}} \\ \psi_{\text{critic}}(k_m^{\text{target}}) &= k_n^{\text{source}}. \end{aligned} \tag{37}$$

4.1.2 Single-Agent Pursuit-Evasion Game: Results

We have discussed several performance metrics for transfer learning in Sect. 2.2. In this study, our goal is reusing the knowledge from the past to accelerate training in a new task. Thus, only a target task scenario should be considered. for the experiment, the same pursuit-evasion game scenario (involving one evader versus three pursuers) is simulated under two conditions: with and without the application of FRT. Each simulation ran for a total of 1000 episodes. The results, as illustrated in Fig. 6, indicated that in both scenarios, the agents successfully learned and converged to a uniform policy that led to the evader's escape from the three pursuers. This evaluation was reiterated in 10 separate trials, each spanning 1000 episodes.

In Sect. 2.2, we introduce *Total Reward* as a pivotal performance metrics for transfer learning. Figure 7 depicts the accumulated rewards collected by the agent across these trials when learning with the transfer, while Fig. 8 portrays the same under learning without the transfer.

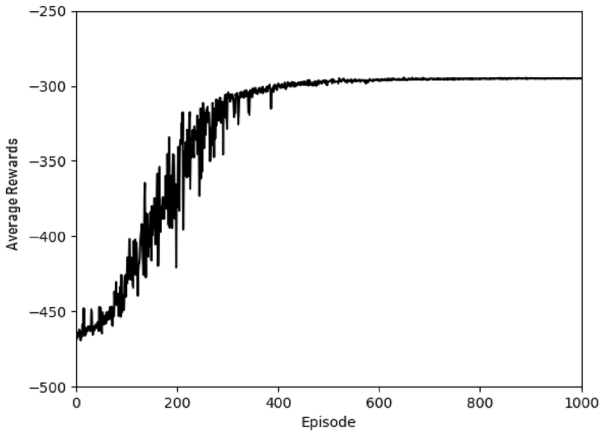


Fig. 7 Average reward using FRT in 1 evader vs. 3 pursuers game (only the evader is learning)

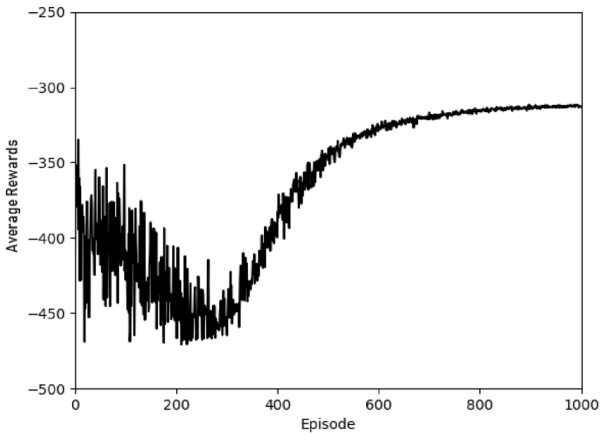


Fig. 8 Average reward without Transfer in 1 evader vs. 3 pursuers game (only the evader is learning)

These rewards were averaged over the ten trials to ensure a comprehensive understanding of the agent's performance under the influence of FRT. The presented figures highlight

that the agents accumulate more rewards when FRT is applied, as evidenced by the area under the reward curve.

Another key metric discussed in Sect. 2.2 is *Time to Threshold*, the learning time needed for the target agent to reach certain performance threshold. In the context of this simulation, it pertains to the convergence to an effective escape policy of the agent. The presented figures illustrate the enhanced convergence speed of the escape policy, as evidenced by the reward trajectory in the scenario employing FRT. This faster convergence speed indicates that the agent benefits from a more rapid learning curve when FRT is utilized.

Moreover, Figs. 9 and 10 provide a comparative view of the agents' movement trajectories at the halfway mark of 500 episodes, both with and without the application of FRT. Figure 9 illustrates the case with FRT, where the evader agent manages to learn an effective policy after approximately 500 episodes of training, leading to a successful escape within roughly 350 time steps in that particular episode. In contrast, as shown in Fig. 10, the agent, when not benefiting from FRT, struggles to acquire an effective policy even after a similar number of training episodes, taking a notably longer time of around 400 steps within the same episode to escape. This comparison clearly demonstrates the utility and efficiency of the FRT application in promoting faster learning in the scenario of single-agent Pursuit-Evasion Game.

4.2 Guarding-a-Territory Game

The guarding-a-territory differential game was introduced by Isaacs [31]. It is a zero-sum differential game with continuous state space and action space. Both guards and the invader are learning with the FACL algorithm to find their optimal strategies in this game. The guard's objective is to intercept the invader as distantly from the territory as possible, while the invader aims to reach the territory as closely as possible, evading capture in the process. The dynamics of the FACL agents in this game mirror those of the previously described pursuit-evasion game, defined in equation (32). The invader is running at a higher speed than the guard. The speed of the guard is set at 1 unit/second and the invader is set at 1.3 units/second. For each episode in the training, the game terminates when the invader reaches the territory or gets caught by the guard. This game has been studied and implemented using the FACL learning algorithm in [32].

4.2.1 Multi-agent Guarding-a-Territory Game: Setup

Similar to the pursuit-evasion game presented in Sect. 4.1, we use triangular membership functions to define the fuzzy sets in the state variables. Ten symmetrical and uniformly spread triangular membership functions are defined for the angle θ_T and seven triangular membership functions are defined for each the Manhattan distance variable d_{IG}. In the game of a single invader versus a single guard, the configuration of the state variables for FACL agents is identical between the invader and the guard. We show the state variables in Table 5. We set the initial learning rates of the actor and the critic in the FACL algorithm as follows: $\beta_0 = 0.05$ and $\alpha_0 = 0.1$. We set $\beta_0 < \alpha_0$ so that the actor will converge slower than critic to prevent instability in the actor [28]. The variance of the initial exploration noise σ and the discount factor γ are set as follows: $\sigma = 3$ rad/second and $\gamma = 0.25$. The decay rates of the learning factors and exploration noise after each episode k are set as follows: $\beta_{k+1} = 0.995^k \beta_0$, $\alpha_{k+1} = 0.999^k \alpha_0$, and $\sigma_{k+1} = 0.994^k \sigma_0$. The values for the learning rates, the exploration noise, and the discount factors were carefully chosen based on a priori knowledge of the problem and heuristics from the experiments. These configurations have

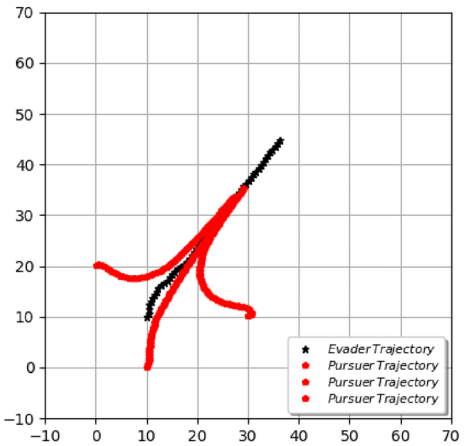


Fig. 9 Episode 500 using FRT in 1 evader vs. 3 pursuers game (only the evader is learning)

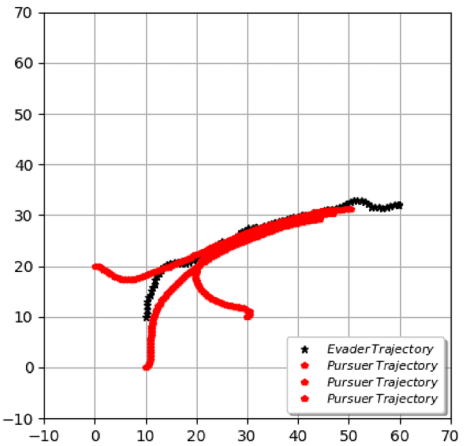


Fig. 10 Episode 500 without FRT in 1 evader vs. 3 pursuers game (only the evader is learning)

Table 5 The agent's state variables in 1 invader vs. 1 guard

Source task: 1 invader vs. 1 guard		
State variable	Description	Mfs and range
d_{IG}^{x}	Manhattan distance w.r.t. x-axis between invader and guard	7 uniform triangular MFs over the range of [− 40, 40]
d_{IG}^{y}	Manhattan distance w.r.t. y-axis between invader and guard	7 uniform triangular MFs over the range of [− 40, 40]
θ_T	Angle between the x-axis and Agent's line of sight towards the territory	10 uniform triangular MFs over the range of [-π, π]

demonstrated effectiveness in our simulation. In accordance with prior research in [32], the reward function for the invader in the 1 invader vs. 1 guard case is designed as:

$$R_{t+1} = D(d_{it}(t) - d_{it}(t+1)) - J(d_{ig}(t) - d_{ig}(t+1)), \tag{38}$$

where d_{it} is the Euclidean distance between the invader and territory and d_{ig} is the Euclidean distance between the invader and guard. The component $D(d_{it}(t) - d_{it}(t+1))$ provides a positive reward if the invader moves closer to the territory and a negative reward otherwise. Similarly, the component $-J(d_{ig}(t) - d_{ig}(t+1))$ provides a negative reward if the invader moves closer to the guard and a positive reward otherwise. The constant D and J determines the importance attached to the various components of the reward function. According to [32], we set D = 3, J =1 and the reward function for the guard is defined as:

$$R_{t+1} = P(d_{ig}(t) - d_{ig}(t+1)) + M(d_{gt}(t) - d_{gt}(t+1)), \tag{39}$$

where d_{gt} is the Euclidean distance between the guard and the territory and d_{ig} is the Euclidean distance between the invader and the guard. Likewise, we set P = 1.1 and M= 1.0 as in [32].

Initially, we establish a game scenario featuring a single invader versus a single guard. Figure 11 depicts the trajectory of the agents' movements after 600 episodes. Subsequently, we simulate the target task, introducing an additional guard into the gameplay. Therefore, the input state space for the invader agent changes in the target task. Table 6 illustrates the design of the new state variables and membership functions (MFs).

With this altered state space, the rule counts in the target task rise from 490 to 24, 010. However, for the guards, the state space remains consistent between the source and target tasks, leaving the rule count at a constant 490. As outlined in Sect. 4.1's description of the pursuit-evasion game, we employ the same FRT approach to perform knowledge transfer on the invader agent, while the learned consequent parameters from the source task are directly copied to the target task for the guards. While the reward

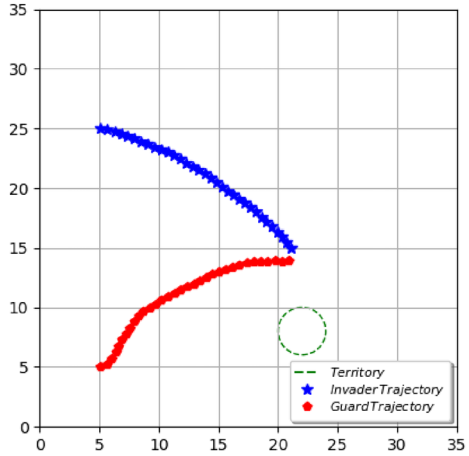


Fig. 11 Game trajectory 1 invader vs. 1 guard in Territory game as the source task

function for the guard remains unchanged, the invader's reward function in the target task evolves:

$$R_{t+1} = D(d_{it}(t) - d_{it}(t+1)) - J1(d_{ig1}(t) - d_{ig1}(t+1)) - J2(d_{ig2}(t) - d_{ig2}(t+1)), \tag{40}$$

where $J2(d_{ig2}(t) - d_{ig2}(t+1))$ is the new component introduced in the reward function due to the additional guard. d_{ig1} is the Euclidean distance between the invader and guard 1 and d_{ig2} is the Euclidean distance between the invader and guard 2. We set D = 3 and J1 = 1 and J2 = 1 according to [32].

In the target task featuring one invader versus two guards, we engage the FACL agents in the game both with and without the application of FRT. In both scenarios, all three participants are able to learn and converge towards their respective optimal strategies, as illustrated in Fig. 12. Our previous research [32] presents the optimal strategies for rational invaders and guards utilizing an Apollonius circle approach.

4.2.2 *Multi-agent Guarding-a-Territory Game: Results*

Similar to the single-agent scenario, we use the *time to threshold* metric, as introduced in Sect. 2.2, to assess the

Table 6 Invader's state variables in 1 invader vs. 2 guards

Target task: 1 invader vs. 2 guards		
State variable	Description	MFs and range
d_{IG1}^{x}	Manhattan distance w.r.t. x-axis between invader and guard 1	7 uniform triangular MFs over the range of [− 40, 40]
d_{IG1}^{y}	Manhattan distance w.r.t. y-axis between invader and guard 1	7 uniform triangular MFs over the range of [− 40, 40]
d_{IG2}^{x}	Manhattan distance w.r.t. x-axis between invader and guard 2	7 uniform triangular MFs over the range of [− 40, 40]
d_{IG2}^{y}	Manhattan distance w.r.t. y-axis between invader and guard 2	7 uniform triangular MFs over the range of [− 40, 40]
θ_T	Angle between the x-axis and invader's line of sight towards the territory	10 uniform triangular MFs over the range of [-π, π]

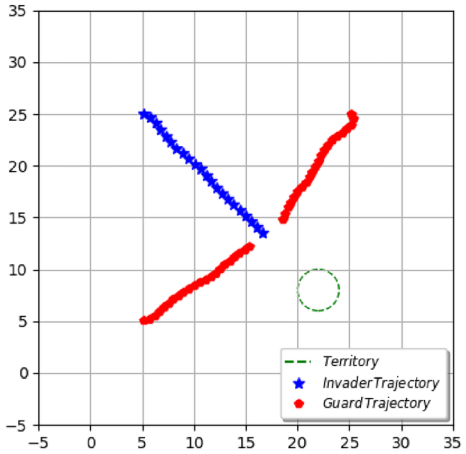


Fig. 12 Game trajectory 1 invader vs. 2 guard in Territory game as the target task

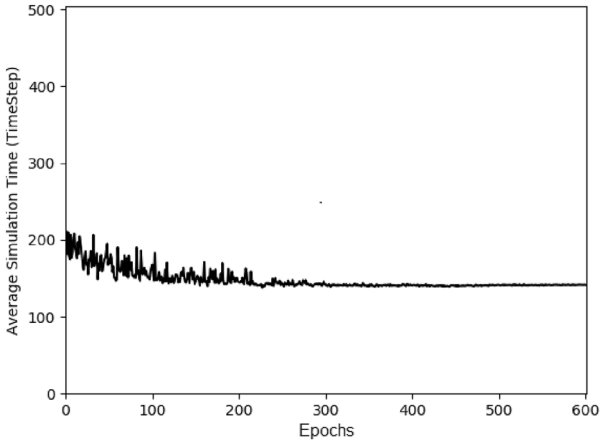


Fig. 13 Simulation time steps in 1 invader vs. 2 guards game when all agents are learning with FRT

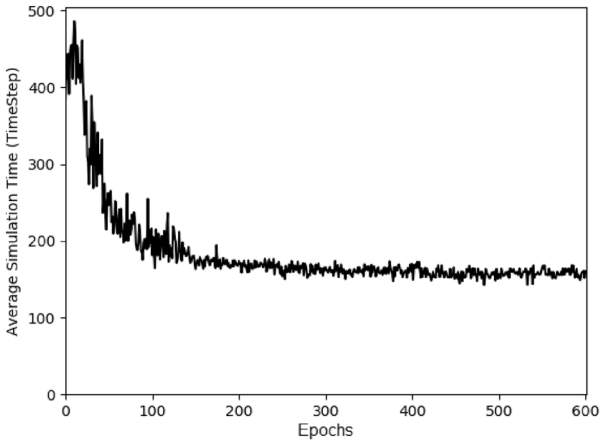


Fig. 14 Simulation time steps in 1 invader vs. 2 guards game when all agents are learning without FRT

performance of the transfer learning. Considering that this is a multi-agent, zero-sum differential game, our interest lies in the time taken for the game to attain its equilibrium state, in which the optimal strategies for rational invaders and guards are learned, and their game trajectories are illustrated in Fig. 12. Thus, in this particular instance, we select the game's equilibrium state as the pre-determined performance level. We utilize a metric of simulation time steps per episode, which encapsulates the trajectory of elapsed time in each episode over the span of the training. The simulation is run over 10 learning trials, each comprising 600 episodes. Figures 13 and 14 respectively depict the simulation time steps per episode, averaged over these 10 trials, for both the FRT and non-FRT scenarios.

When FRT is utilized, the differential game's equilibrium state is reached considerably faster, with the three competing agents learning their optimal strategies rapidly. Since the guard's state space remains unaltered between the source and target tasks, their final performance after 600 episodes in the target task could be considered the outcome of training spanning 1200 episodes, including 600 episodes from the source task, with a reset configuration (learning rate, exploration noise, and discount factors) preceding the target task's initiation. In contrast, the invader must adapt to a new state space, which

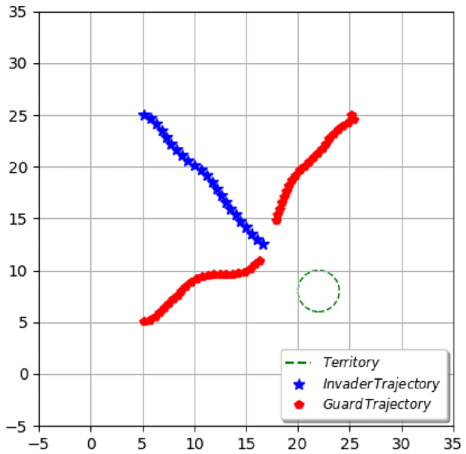


Fig. 15 Episode 300 in 1 invader vs. 2 guards game when all agents are learning with FRT

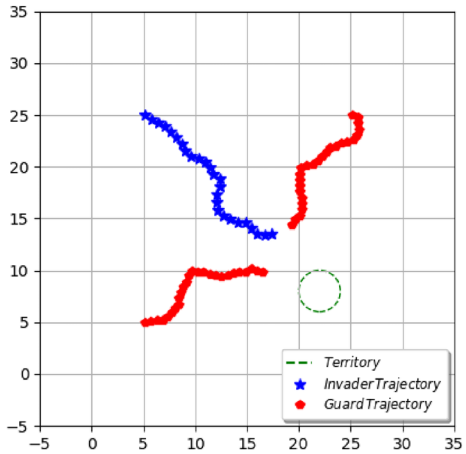


Fig. 16 Episode 300 in 1 invader vs. 2 guards game when all agents are learning without FRT

consequently benefits the guards more significantly from the transfer.

Nonetheless, FRT facilitates the invader in learning to move in a straight line towards the territory more rapidly rather than moving arbitrarily. This improvement accelerates the capture from the guards' viewpoint, thereby leading to quicker game completion, as depicted by fewer time steps for each episode in Fig. 13. In contrast, in the absence of FRT, agents tend to struggle initially before slowly converging to their optimal strategies. This behavior accounts for the higher number of time steps per episode seen in Fig. 14, particularly in the game's early stages. Furthermore, we depict the game trajectory of the agents' movements at 300 episodes in both scenarios in Figs. 15 and 16. As highlighted in these figures, the agents' performance is notably superior at 300 episodes when FRT is employed.

5 Conclusion

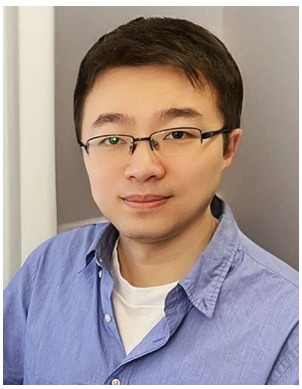
In this paper, we investigated the feasibility to transfer knowledge between tasks that are interrelated and utilizing the FACL algorithm. To enable this process of knowledge transfer, we proposed the FRT method which maps the consequent parameters between the source and target task based on rule similarity. We showed the FRT method and the concept of the similarity measurement between fuzzy rules in Sect. 3. We demonstrate these ideas through simulations in two sets of differential games; the pursuit-evasion game and the guarding-a-territory game in Sect. 4. In all simulations, the results exhibit a consistent pattern that the FRT method is able to speed up the learning process of the FACL agents within the target tasks.

One pivotal direction of our future work will center around how fuzzy rules from the source tasks can be selectively transferred. At present, our methodology entails the indiscriminate transfer of knowledge containing all fuzzy rules from a source task to a target task. This approach, although sufficient in general cases, may have certain limitations. While some rules prove to be remarkably relevant and beneficial, others may not yield similar levels of efficacy or might even introduce disruptions within a new task. Can we potentially improve the existing solutions if we advance towards selective knowledge transfer within fuzzy rules? Another direction of our future work is combining multiple source tasks to produce a richer transfer of experience. Our current work is primarily constrained to the use of a singular source task for the transfer process. However, it prompts the question: could the integration of knowledge from multiple source tasks lead to a more robust generalization for learning within a target task? These are some interesting questions that need to be investigated in subsequent studies.

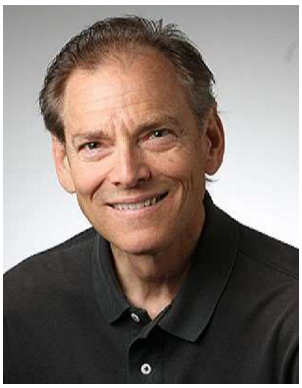
References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT, Cambridge (1998))
2. Schwartz, H.M.: Multi-Agent Machine Learning: A Reinforcement Approach. Wiley, Hoboken (2014)
3. Jouffe, L.: Actor-critic learning based on fuzzy inference system. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 339–344 (1996)
4. Wang, L.-X.: A Course in Fuzzy Systems and Control. Prentice Hall, Upper Saddle River (1997)
5. Andrecut, M., Ali, M.K.: Fuzzy reinforcement learning. Int. J. Mod. Phys. C **13**(05), 659–674 (2002)
6. Kuo, Ping-Huan., Jun, Hu., Lin, Ssu-Ting., Hsu, Po-Wei.: Fuzzy deep deterministic policy gradient-based motion controller for humanoid robot. Int. J. Fuzzy Syst. **24**(5), 2476–2492 (2022)

7. Tsai, C.-C., Chen, H.-Y., Chen, S.-C., Tai, F.-C., Chen, G.-M.: Adaptive reinforcement learning formation control using ORFBLS for omnidirectional mobile multi-robots. Int. J. Fuzzy Syst. **25**(9), 1–14 (2023)
8. Wang, X., Bin, X., Guo, Y.: Fuzzy logic system-based robust adaptive control of AUV with target tracking. Int. J. Fuzzy Syst. **25**(1), 338–346 (2023)
9. Taylor, M.E., Stone, P.: Behavior transfer for value-function-based reinforcement learning. In Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 53–59 (2005)
10. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint (2015). arXiv:1511.05952
11. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: theory and application to reward shaping. In: ICML, vol. 99, pp. 278–287 (1999)
12. Awheda, M.D., Schwartz, H.M.: A residual gradient fuzzy reinforcement learning algorithm for differential games. Int. J. Fuzzy Syst. **19**, 1058–1076 (2017)
13. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: a survey. J. Mach. Learn. Res. **10**(Jul), 1633–1685 (2009)
14. Taylor, M.E., Stone, P., Liu, Y.: Transfer learning via inter-task mappings for temporal difference learning. J. Mach. Learn. Res. **8**(Sep), 2125–2167 (2007)
15. Da Silva, F.L., Glatt, R., Costa, A.H.R.: Simultaneously learning and advising in multiagent reinforcement learning. In: Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems, pp. 1100–1108 (2017)
16. Ramon, J., Driessens, K., Croonenborghs, T.: Transfer learning in reinforcement learning problems through partial policy recycling. In: European Conference on Machine Learning, pp. 699–707 (2007)
17. Taylor, M.E., Stone, P.: Cross-domain transfer for reinforcement learning. In: Proceedings of the 24th International Conference on Machine Learning, pp. 879–886 (2007)
18. Taylor, M.E., Whiteson, S., Stone, P.: Transfer via inter-task mappings in policy search reinforcement learning. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, p. 37 (2007)
19. Razavi, R., Klein, S., Claussen, H.: Self-optimization of capacity and coverage in lTE networks using a fuzzy reinforcement learning approach. In: 21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1865–1870 (2010)
20. Jouffe, L.: Fuzzy inference system learning by reinforcement methods. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **28**(3), 338–355 (1998)
21. Van Buijtenen, W.M., Schram, G., Babuska, R., Verbruggen, H.B.: Adaptive fuzzy control of satellite attitude by reinforcement learning. IEEE Trans. Fuzzy Syst. **6**(2), 185–194 (1998)
22. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **1**, 116–132 (1985)
23. Givigi, S.N., Schwartz, H.M., Lu, X.: A reinforcement learning adaptive fuzzy controller for differential games. J. Intell. Robot. Syst. **59**(1), 3–30 (2010)
24. Desouky, Sameh F., Schwartz, Howard M.: Self-learning fuzzy logic controllers for pursuit-evasion differential games. Robot. Auton. Syst. **59**(1), 22–33 (2011)
25. Dai, X., Li, C.-K., Rad, A.B.: An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. IEEE Trans. Intell. Transp. Syst. **6**(3), 285–293 (2005)
26. Awheda, M.D., Schwartz, H.: A decentralized fuzzy learning algorithm for pursuit-evasion differential games with superior evaders. J. Intell. Robot. Syst. **83**, 35–53 (2015)
27. Schwartz, H.: An object oriented approach to fuzzy actor-critic learning for multi-agent differential games. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 183–190 (2019)
28. Givigi, S.N., Schwartz, H.M., Lu, X.: An experimental adaptive fuzzy controller for differential games. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 3017–3023 (2009)
29. Randløv, J,, Alstrøm, P.: Learning to drive a bicycle using reinforcement learning and shaping. In: ICML, vol. 98, pp. 463–471 (1998)
30. Gullapalli, V., Barto, A.G.: Shaping as a method for accelerating reinforcement learning. In: Proceedings of the 1992 IEEE International Symposium on Intelligent Control, pp. 554–559 (1992)
31. Isaacs, R.: Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization. Wiley, New York (1965)
32. Analikwu, C.V., Schwartz, H.M.: Multi-agent learning in the game of guarding a territory. Int. J. Innov. Comput. Inf. Control **13**(6), 1855–1872 (2017)



Dawei Ni is a Senior Data Scientist specializing in Artificial Intelligence (AI) and Machine Learning (ML) within the telecommunications sector at Ericsson Canada. He is an integral part of the Global Artificial Intelligence Accelerator (GAIA) located in Montreal. Dawei holds a Master of Applied Science in Electrical and Computer Engineering, as well as a Bachelor of Engineering in Communications Engineering, both from Carleton University. His work primarily centers on the practical realization of AI and ML technologies in telecommunications, including LTE/5G and prospective 6G systems. His research interests include fuzzy reinforcement learning, machine learning and its adaptations in telecom, complexity reduction in ML models, as well as various other topics in deep learning and classical machine learning.



Howard M. Schwartz received his BEng. degree in Civil Engineering from McGill University, Montreal, Canada, and his MSc. and Ph.D. degrees from M.I.T., Cambridge, Massachusetts, in Aerospace Engineering and Mechanical Engineering, respectfully. He is currently Professor of Electrical and Computer Engineering at Carleton University. He served as Department Chairman from 2009 to 2013. Professor Schwartz's research is in the field of Machine Learning and Multi-Agent Machine Learning for teams of mobile robots. He is Associate Editor of the IEEE Transactions on Cybernetics and is the author of the book Multi-Agent Machine Learning: A Reinforcement Approach, Wiley, 2014.