Journal of Computer Science

Original Research Paper

Transfer Learning in Attack Avoidance Games

Edwin Torres and Fernando Lozano

Department of Electrical and Electronic Engineering, University of Los Andes, Bogota, Colombia

Article history
Received: 07-07-2019
Revised: 08-09-2020
Accepted: 29-10-2020

Corresponding Author:
Edwin Torres
Department of Electrical and Electronic Engineering,
University of Los Andes,
Bogota, Colombia
Email: ed.torres20@uniandes.edu.co

Abstract: Transfer knowledge is a human characteristic that has been replicated in machine learning algorithms to improve learning performance measures. However, little success has been accomplished in reinforcement learning tasks when a function approximation is needed to estimate the value functions. In this study, we present a new strategy to facilitate knowledge transfer when an agent is learning to solve a sequence of increasing difficulty tasks. We show that the tasks sequence is an effective scenario to segment the function approximation hypothesis space allowing a faster learning especially in the last task of the sequence. Moreover, the sequence allows the design of a similarity function that helps the agent to determine in which moment is more appropriated to use the transfer autonomously. We empirically show the importance of the presence of all the tasks in the established ordering to accomplish the best improvement in the learning time for the last task.

Keywords: Reinforcement Learning, Neural Networks, Transfer Learning

Introduction

Reinforcement learning algorithms require a large amount of data collected through the agent-environment interaction. This need in the volume of exploration to find adequate solutions restricts the applicability of the methods, a situation that becomes more critical when a function approximator is required to estimate the value functions. In this context, knowledge transfer is used to facilitate learning in new tasks based on previously acquired knowledge. In this way, it is possible to guide the exploration through mechanisms that make it possible to relate past experiences with those found in new tasks and provide advice to the agent when selecting actions in specific states. In this study, we propose a strategy for the task sequence construction that facilitates the function approximation and the use of transfer learning. This strategy produces a better learning rate for the agent in the final task of the sequence. We start from a fact that is generally found in human learning processes (Piaget, 1963; Vygotsky and Kozulin, 1962), in which tasks must be organized according to their level of difficulty and each new task must expand the knowledge of the previous task. Under this scheme, it is necessary to consider the capacity of representation of the function approximators used. Since, depending on its structure, the amount of exploration required to converge to an acceptable solution will also depend. In our method, we propose the use of structures whose complexity increases depending on the difficulty of the task to be solved. Thus, easy tasks use simple structures and difficult tasks use more complex structures. For example, in the use of neural networks, the increase in complexity is done by adding more neurons to the structure of the network. We build an experimental framework to let an agent learn in a sequence of related tasks ordered by its increasing difficult (Madden and Howley, 2004; Taylor *et al*., 2007a) and with the help of a similarity measure ensure when is most beneficial for the agent use transfer in an autonomous way. The rest of this document is organized as follows: Section 2 (Background) briefly presents the reinforcement learning framework, section 3 (Related work) we give a brief summary of the previous work in transfer learning applied to reinforcement learning, section 4 (proposed method) we give a detailed description of the proposed transfer strategy, in section 5 (Experimental results) we present the results of the experiments on attack avoidance tasks. Finally, section 6 (Conclusion) we resume the principal findings of this work.

Background

The tasks studied in this study can be formally described as a series of sequential decision problems. Each problem consists of a series of decisions that will lead to a final state in which the agent will evaluate if the decisions taken were appropriate: Whether some goal was achieved or not. Mathematically this process can be modeled as a Markov Decision Process (MDP),

DOI: 10.3844/jcssp.2020.1465.1476

(Puterman, 1994). An MDP is specified by the 5-tuple $\langle S, A, P, R, \gamma \rangle$. S is the set of possible states. A is the set of actions. P is a (possibly stochastic) transition function P: $S \times A \times S \rightarrow \mathbb{R}$, which indicates the probability that taking action a in state s will lead to state s'. R is the reward function P: $S \times A \times S \rightarrow \mathbb{R}$, which maps each state-action pair and the resulting state s' to a real number, the instantaneous reward. $\gamma \in [0, 1)$ is the discount factor for future rewards. At each time step, an action an is taken according to the current policy π: $S \rightarrow A$ which maps states to actions. If the MDP is episodic (as will be the assumption considered in this study), it will begin in a start state, then a series of actions will be taken until it reaches a terminal state, referred to as a goal state. Given the MDP specification, the problem is to maximize the expected sum of discounted rewards. The reward R_t represents a one-step measure of performance, that is, how good it was to take action a_t when the agent was in state s_t. The return G_t is the sum of discounted rewards and it is defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (1)$$

Maximizing the expected value of G_t (long-term measure of performance) implies finding an optimal policy $\pi*$. This policy allows the agent to select the best possible action for each state according to the maximization. Given the stochastic nature of the MDP, we are interested in the expected value of G_t, which can be characterized in two ways:

- The state value function: Expected sum of discounted rewards given the initial state s following the policy π:

$$V^{\pi}(s) = E\{G_t \mid s_t = s\} \quad (2)$$

- The state-action value function: Expected sum of discounted rewards given the initial state s and action a and following the policy π thereafter:

$$Q^{\pi}(s,a) = E\{G_t \mid s_t = s, a_t = a\} \quad (3)$$

The problem of finding an optimal policy $\pi*$ is solved through the maximization of one of these functions.

An MPD can be solved through Dynamic Programming (DP) (Bellman, 1957), but a complete and correct knowledge of the transitions and rewards is required. DP iteratively computes approximations for the true value function, improving them over time. However, the full knowledge requirement is not always possible; especially in MDPs with large high-dimensional state and action spaces where it is unfeasible to determine the dynamics P. Additionally, as the number of states increases, the computational requirements make DP untractable.

Reinforcement Learning-RL (Sutton and Barto, 1998) (Also known as Approximate Dynamic Programming-ADP) offers a powerful set of tools for sequential decision tasks with large state-action spaces. Most of them are based on Temporal Difference (TD) methods, such as Q-learning (Watkins, 1989) and SARSA (Rummery and Niranjan, 1994), in which the solution is learned by backing up experienced rewards through time, resulting in an estimated state-action value function. Updating of the current best policy is generated from Q by selecting the action that maximizes value for the current state:

$$\pi*(s) \leftarrow \underset{a \in A}{argmax}\, Q^{\pi}(s,a) \quad (4)$$

The agent taking the sequential decisions (actions) must balance between exploration, where the agent chooses a random action to observe different states and learn more about the environment and exploitation, where the agent selects actions according to the current policy (the current best action). A basic strategy that balances these two options is ε-greedy action selection: The agent selects a random action with probability ε and the current best action is selected with probability 1-ε (where ε is in [0, 1]).

The agent interacts with the environment and a value function actualization is carried out for each new sample (a sample: s_t, a_t, r_t, s_{t+1}) one at a time. For this reason, a large amount of experience is needed to obtain a near-optimal value function, to solve the task and produce an optimal policy.

Function Approximation in Reinforcement Learning

In problems where the number of states and available actions is large there is an exponential growth on the computational requirements needed to solve an RL problem (computation time and storage). In these cases, it is necessary to make use of approximation techniques to construct a compact representation of the value functions, which could be parametric or nonparametric. A parametric approximator for Q could be:

$$Q(s_t, a_t) = \Phi^T(s_t, a_t) w \quad (5)$$

where, the Q function approximator is parameterized by a n-dimensional vector w and a set of n basis functions $\Phi(s_t, a_t)$ that are used to extract pre-defined characteristics from the state-action pair. The approximation is built using samples collected from the interaction between the agent and the MDP (environment). With these samples, a regression is carried out to optimize the function approximator over the state-action space using an error measure over the value differences of the value function in each iteration. In RL algorithms a common error measure used is the

Least Mean Square error -LMS- and the optimization is usually done through a gradient descent method.

The function approximator can be linear or nonlinear in the parameters. Typically, neural networks (nonlinear) and radial basis (linear) functions are used to implement approximators. The selection of a specific method depends on the problem and the generalization capabilities of the function approximator. Algorithms for problems with continuous RL problem include: TD Learning with Function Approximation, Policy Gradient (Sutton *et al*., 2000), LSTD (Boyan, 1999), LSPI (Lagoudakis and Parr, 2003), Batch methods FQI (Ernst *et al*., 2005) and Deep Q-learning (Mnih *et al*., 2013).

Transfer Difficulties in RL

As mentioned before, to solve tasks with large state or action spaces it is needed the use of function approximation techniques. In each MDP, the state-action value function is unknown and it has to be estimated and incrementally build from rewards samples and previous estimations of the same function. This situation becomes more critical in harder tasks but can be easily managed in easier tasks as we will show later. Additionally, this estimation procedure has an important influence on the stability of the function approximation process and for the final performance of the learning, especially when a nonlinear function approximator is used. Some works (Boyan and Moore, 1995; Baird, 1995; Tsitsiklis and Van Roy, 1997) have reported low performance and divergence when function approximation is used. The exploration strategy (e.g., ε-greedy) also has an impact on the function approximation stability. The value function estimation used to derive the sampling policy affects the way the task samples are taken and these samples will affect the next value function estimation. This alternation between sample and learning can result in larger learning times and in the worst case to a divergence on the function approximation.

Related Work

The lifelong learning framework proposed by (Thrun, 1996) describes a scenario in which an agent interacts with a sequence of tasks. This scenario includes all possible future tasks that an agent may encounter over its lifetime. In RL, the lifelong learning setting focuses on problems in which an agent moves from one environment to another, or when the agent is in a changing environment. Additionally, it is assumed that exists some relation between the tasks MDPs. In our case, we need this relationship to construct an arrangement of the tasks. Tanaka and Yamamura (1997) proposed a method to pre-train a neural network using the knowledge from the previous task, using it to bias the weights of the neural network that is used in subsequent tasks. Their experiments show the importance of the relation between the tasks and how this impacts the agent's learning in future tasks. White *et al*. (2012) investigated the use of policies from past tasks to construct general value functions that are used in an off-policy setting to improve the agent's performance in a new task. The general value functions can capture a wide variety of characteristics from the environment dynamics, which allows the agent to develop and preserve multiple capabilities. Another learning strategy concerned with the use of previous knowledge is Transfer Learning (TL), which is primarily focused on the task to task transfer of knowledge. TL has been successfully applied to several problems in machine learning (Pan and Yang, 2009). TL assumes that there exists a clear identification of the boundary where a task ends and a new one begins. Taylor and Stone (2009) presented a survey of research done in transfer learning related to RL and introduced the most used metrics to measure the efficiency of the transfer. These works describe methods like calculating prior probabilities, transfer of samples, value function and policy transfer and value function structure. Konidaris *et al*. (2012) defined a new value function, which considers only the common characteristics across all the tasks and computes an approximation of the *Q* values. In this way the transfer process can be done with this function, using it to determine a value that has relevance for the action selection in every task. For this reason, additional training is necessary to approximate this function to the *Q* values. This work shows an alternative source of knowledge and validates the idea of searching for other knowledge sources. Taylor *et al*. (2007b) proposed Intertask mappings to allow previous *Q* values to be used in new tasks. The intertask mapping does a transformation of the state an action spaces from the target task to the source task, then a *Q* value is computed and used the determine which action to take in the target task. However, the authors do not mention which is the process to design or select the intertask mappings. A completely different approach is proposed by (Lazaric, 2008), the authors determined the sample relevance from the source tasks to determine which ones to use in the agent's training when learning in the target task. This framework allows a different transition model or a different reward function from the source tasks and the target task, but the state-action spaces must be the same. The objective of the sample relevance analysis is to avoid negative transfer, which worsens the agent's performance. A related approach called Multitask learning (Caruana, 1997) has proved to be effective when transferring from multiple source tasks simultaneously. Other relevant works (Drummond,

2002; Torrey *et al*., 2005; Liu and Stone, 2006; Wilson *et al*., 2012; Fernández and Veloso, 2006) have applied transfer learning strategies to RL, using different strategies: Policy advice, value function transfer, model estimation. Each one of these strategies finds knowledge in a different location and based on that implements a method to transfer it. A method related to the learning in a task sequence in the context of supervised learning, curriculum learning, is presented in (Bengio *et al*., 2009) and (Kumar *et al*., 2010). These works show experimentally the effect in the learning rate produced by the task ordering inside the sequence. Weinshall *et al.* (2018) presents an empirical evaluation of a curriculum in where the tasks are ordered by difficulty, itis experimentally shown that the convergence rate of a deep neural network improves as a result of transfer using the proposed curriculum. An extensive survey in curriculum learning applied to RL is found in (Narvekar *et al*., 2020). This work summarizes a series of approaches that focus on the transfer of knowledge when an agent is given a series of tasks to solve. The work analysis is carried out based on task generation, sequencing and transfer learning.

Materials and Proposed Method

Based on the following characteristics of the human learning processes we proposed a technique to overcome some of the problems mentioned before:

- The tasks faced by humans appears in a sequentially increasing difficulty order (Scaffolding (Vygotsky and Kozulin, 1962)
- In the learning of new tasks humans make use of previously acquired knowledge finding similarities with the old tasks (Zone of Proximal Development (Vygotsky and Kozulin, 1962)
- Each new task expands the previous task state and knowledge representation

Our method is related to the curriculum learning framework from a RL perspective that replicates characteristics of the human learning process: An agent is learning to solve a sequence of tasks, the curriculum, that are related and ordered by increasing difficulty. Each task constitutes an episodic MDP with a large state space. The agent must learn to solve each MDP starting with the easiest one, the first task in the sequence. At the end of the first task, the agent will have knowledge that can be used (transferred) to solve the next task, which we assume to be related to the previous task, as shown in Fig. 1. The task decomposition into a sequence of ordered tasks can help to a function approximation hypothesis space segmentation as shown in Fig. 2. Easier tasks can use a small function approximator and thus have a smaller hypothesis space in where the value function optimization and the reinforcement learning problem can benefit from: (1) Less prone to function approximation divergence, (2) Better task exploration (sampling policy), need to explore only in the new part of the space (i.e., state space). There is already a knowledge of some part of the space. (3) Less time and samples needed to find a near optimal policy.

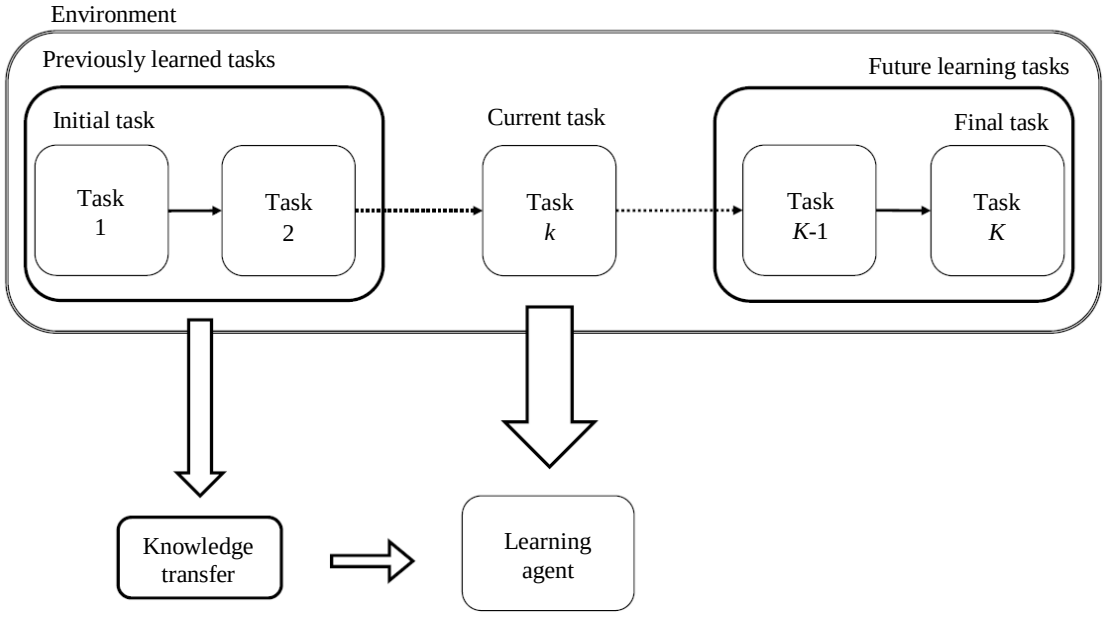

Fig. 1: Sequence of N MDPs

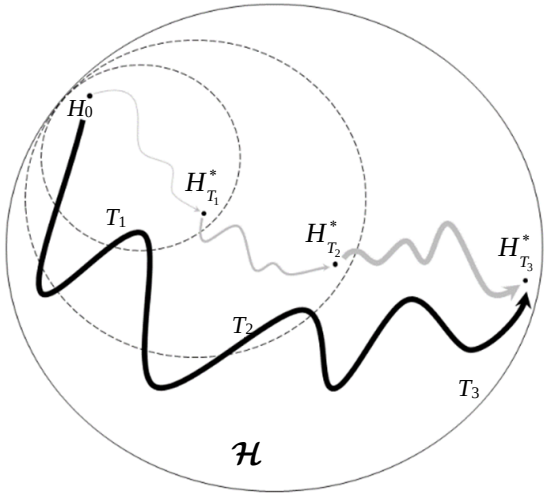

Fig. 2: Function approximation hypothesis space. In this case, the hypothesis space segmentation improves the function approximation estimation process for the target task T_3. The learning of less complex tasks results in a better initial point for the learning of the target task

Task Sequence Generation: The Curriculum

The objective of learning in a sequence of RL tasks is to improve some performance measures in the learning of the last task T_K. For this reason, we adopt a top-down strategy. This means starting from the last one which is the most difficult or complex to solve. Then, a decomposition step must be applied to T_K in order to obtain an easier T_{k-1} and continue until some desired initial task. The decomposition step consists in the application of a rule f to a given task T: $f \times T_k \rightarrow T_{k-1}$

There are a variety of rules f and each one can generate a different sequence $\mathbb{T}$. Even more, a different rule can be applied in each decomposition step. We call $\mathbb{F}$ the set of all possible rules applicable to T_K.

From the application $\mathbb{F}$ to T_K we can generate the sequences set T, which is the set of all possible sequences $\mathbb{T}$ whose final task is T_K.

Definition 1. Task sequence $\mathbb{T}$. Given a target task T_K and rule f. We can obtain easier related tasks T_{K-n}, with $n < K$, by applying f repeatedly to every task. With these tasks, we can construct the sequence of tasks $\mathbb{T}$ ordered by increasing difficulty.

This ruled top-down strategy guaranties a relation between the tasks in $\mathbb{T}$ and knowledge preservation since every task in $\mathbb{T}$ contains all the information from the previous easier tasks in $\mathbb{T}$. In this context easier means that an agent can learn a policy that solves T_{k-1} in a shorter time (less experience needed) than the time needed to solve T_k. Using this time measure we can say that a task is easier or harder than another task.

Is this context easier means that an agent can learn a policy that solves T_{k-1} in a shorter time (less experience needed) than the time needed to solve T_k and the opposite applies to harder. Using this time measure we can say that a task is easier or harder than another task.

Task Similarity

In contrast to previous works (Carroll and Seppi, 2005; Ferns *et al*., 2012; Bou Ammar *et al*., 2014) where it is necessary to determine which tasks are relevant to transfer from, in our approach the task similarity measure is used to decide when is the best moment to transfer knowledge from the previous task. The agent will calculate this measure through a similarity function which evaluates the relatedness of the present states from T_K with the previous task available information (e.g., Q-value function, policy, state space information). The idea behind the similarity function is intuitive from the human learning perspective, when the agent is learning a new task T_k and face a new situation the obvious first reaction is to relate the actual state to previous experience from T_{k-1} to decide which could be the best action to take. In the same way, we used a rule set $\mathbb{F}$ to generate the sequence tasks, it is necessary a similarity function for knowledge transfer from T_{k-1} to T_k.

Definition 2. Given a state s from T_k and the state representation $I(S_{k-1})$ used in T_{k-1}, the sample similarity of s is defined as:

$$\rho\left(s, S_{k-1}\right) = \frac{1}{1+\exp\left(-\left\|s - I\left(S_{k-1}\right)\right\|\right)} \tag{6}$$

where, I is a function that extract quantitative characteristics that represent a given space (action or state space).

Algorithm

The RL agent's learning algorithm for the learning in each task is shown in Algorithm 1, it is based on (Riedmiller, 2005). A Q value Function Approximator (FA) is initialized and an actualization of this function is done in batch after N interactions between the agent and the environment. The samples are collected using SARSA method (Rummery and Niranjan, 1994). This process is repeated M times (batches) to guarantee and appropriate learning of the task. This approach is called Batch algorithm (Lagoudakis and Parr, 2003; Bradtke and Barto, 1996; Ernst *et al*., 2005).

The function *simulate* (line 5) is used to observe the environment evolution to next state s' when an action a is performed by the agent when it is in state s. The *actionSelection* function (line 6) is used by the agent to select the next action a' when it is in state s' and this function is where the transfer takes place.

DOI: 10.3844/jcssp.2020.1465.1476

Algorithm 1: RL agent learning T_k

Require: N: num samples, M: num batches, $\hat{Q}_{k-1}$: state-action value function from T_{k-1}

1: $Q_k \leftarrow$ initialize FA()
2: **repeat**
3: s, a, q initial Sate()
4: **while** i < N **do**
5: s', $r \leftarrow simulate(s, a)$
6: a', $q' \leftarrow actionSelection\left(Q_k, Q_{k-1}, s'\right)$
7: $X[i] \leftarrow s, a, r, q, q'$
8: $a, s, q \leftarrow s', a', q'$
9: **end while**
10: Q_k train FA(X)
11: **until** M

Algorithm 2 shows the pseudocode that implements the action selection through a transfer strategy from T_{k-1} or using and ε-greedy strategy. The transfer strategy depends on the similarity measure ρ and the transfer rate parameter ϕ which is used to control the amount of transfer. The parameter ρ_0 is used as a threshold to determine when a state s is similar enough to the previous experience. The transfer uses a policy advice method to select the action a through the function approximator $\hat{Q}_{k-1}$ estimated from the task T_{k-1}. The function filter (line 3) adapts the actual state s, since it comes from S_k, to be a state in the form required by Q_{k-1}.

Algorithm 2: Action selection with similarity function

Require: Q_k , $\hat{Q}_{k-1}$, ρ_0, s: state
1: random uniform variables β
2: **if** $\rho(s, I(S_{k-1})) > \rho_0$ and $\beta > \phi$ **then**
3: $\hat{s} \leftarrow$filter(s)
4: Choose a from ŝ using policy derived from $\hat{Q}_{k-1}$
5: **else**
6: Choose a from s using ε-greedy policy derived from Q_k
7: **end if**
8: **return** a, q

Experimental Results and Discussion

In this section, we describe the set of experiments conducted to test our proposed strategy. We introduce the attack avoidance task on which the task sequence and the similarity function were generated.

Attack Avoidance

The attack avoidance game is a discrete version of the pursuit-evasion (Ho *et al*., 1965; Parsons, 1978) and differential games (Isaacs, 1999). These games classes are related to the analysis and modeling of dynamical systems in which a set of variables evolve following a differential equation system.

Our attack avoidance game consists of an agent who must reach a goal zone and an attacker(s) who is(are) pursuing the agent. If the attacker touches the agent before it reaches the goal zone, the agent loses the game, the agent wins otherwise. Additionally, the agent is not allowed to stay in the forbidden zone, which corresponds to zones to both sides of the goal zone. The game board and the actions for the agent and the attacker are shown in Fig. 3.

Using the attack-avoidance game, we designed a set of four tasks shown in Fig. 4. In task T_0 the agent has the same dynamics but there is no attacker. In this case, the agent's objective is to find the path to the goal zone. The rewards are defined as: 100 for reaching the goal zone, −100 for reaching the forbidden zone and 0 otherwise. In the second task T_1 there is one attacker. In the third task T_2 there are two attackers and in the third one T_3 there are three attackers. This increase in the number of attackers makes each game more difficult than the previous one. In these last three tasks, the rewards changed: 1 for reaching the goal zone, −1 for reaching the forbidden zone, or when the attacker touches the agent and 0 otherwise.

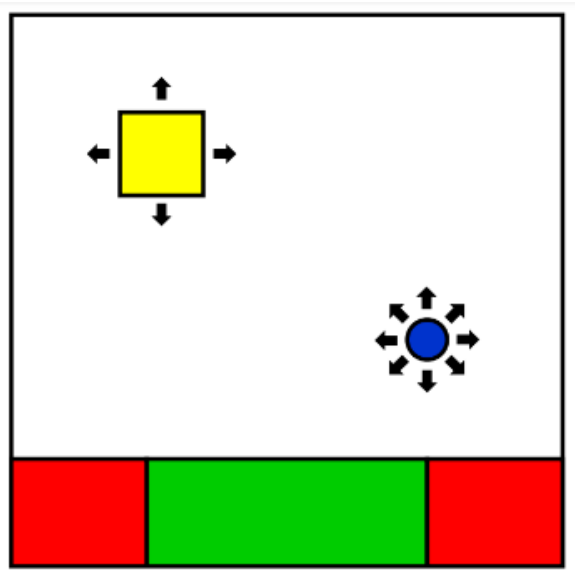

Fig. 3: Attack-avoidance board for task T_1. The agent (gold square) actions: Stay in the same position, move up, left, right or down. Each agent action moves it a distance 1/12 of the board size. Attacker actions: King's moves. The attacker action probabilities are: 0.1 For a random action and 0.9 for an action that minimize the distance to the agent. Each attacker action moves it a distance 1/36 of the board size

DOI: 10.3844/jcssp.2020.1465.1476

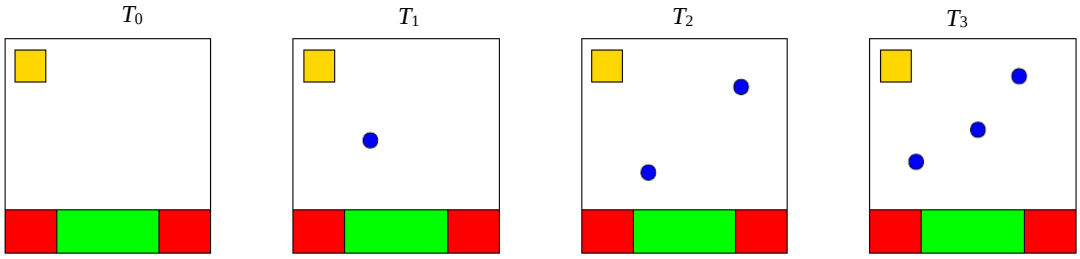

Fig. 4: Task sequence

The sequence containing the four tasks was generated by applying a simplification rule that consisted in eliminating one attacker from every task starting with T_3. In this case, it ended when there were no more attackers on the board.

The state variables are the agent position (x, y), the inverse distance from the agent to the attacker and the inverse distance from the attacker to a fixed point in the goal zone. This state representation is increased in the tasks where there is a new attacker.

To solve each task we took N samples from the environment before making a value function update, as shown in Algorithm 1. In this sampling phase, a ε-greedy strategy collects samples from the interaction with the environment (game simulation), ε was set to 0.1. Next, in the learning phase, an approximation of the value function is calculated using the samples on the SARSA algorithm with γ set to 1.0.

At the beginning of each episode, the agent and the attackers are initialized at random positions. 300 batches of 600 samples were performed to train the agent in T_0 and 6000 batches of 600 samples were performed to train the agent in tasks T_1, T_2 and T_3.

We designed 15 experiments to tests different sequence configurations using the four tasks from the complete attack avoidance sequence. Table 1 shows the tasks included in each curriculum experiment.

Function Approximation

A neural network was used to approximate the Q-value function. We started with a simple model for T_0 an let the model grew as needed along the sequence. A representation of this process is shown in Fig. 5 and the structure of each network is shown in Table 2. The increase process is described as follows: After the training in T_{k-1}, a new neural network is created for T_k using the neural network structure of T_{k-1} and adding nodes in the input layer to be able to receive the state variables for T_k. Additionally, we added more neurons to the hidden layer in order to increase the network approximation space for the more complex T_k. The weights of the new neural network were initialized randomly.

Similarity and Policy Advice

Transfer was done using a policy advice strategy inside the action Selection function, Algorithm 1 line 6. For this knowledge source, we used thee Q-value approximator obtained in T_{k-1} to derive the policy π_{k-1}. This policy was used to advise the agent in the T_k training using a ε-greedy ρ-advice strategy. Algorithm 2 shows the action selection function where the agent used a similarity function ρ to determine in which states the agent could use the previous policy. For the attack avoidance game, a similarity function for each T_{k-1} to T_k transfer was designed. In Fig. 6 are shown the similarity functions used in the transfer to T_3. To be able to measure the influence of the amount of transfer used we added a transfer control rate variable ϕ. This variable was set to ten different values ϕ = [0.0,0.1,0.2, ...,0.9] and for each value it was fixed during the training.

Table 1: Experiment sequences

Experiment	T_0	T_1	T_2	T_3	Target task
E_1	X				T_0
E_2		X			T_1
E_3	X	X			T_1
E_4			X		T_2
E_5	X		X		T_2
E_6		X	X		T_2
E_7	X	X	X		T_2
E_8				X	T_3
E_9	X			X	T_3
E_{10}		X		X	T_3
E_{11}	X	X		X	T_3
E_{12}			X	X	T_3
E_{13}	X		X	X	T_3
E_{14}		X	X	X	T_3
E_{15}	X	X	X	X	T_3

DOI: 10.3844/jcssp.2020.1465.1476

Table 2: Neural net structure

Task	Attackers	NN structure	NN parameters	State space size
T_0	0	7-7-1	64	144
T_1	1	9-10-1	111	171072
T_2	2	11-14-1	183	203233536
T_3	3	13-19-1	286	2,41441E11

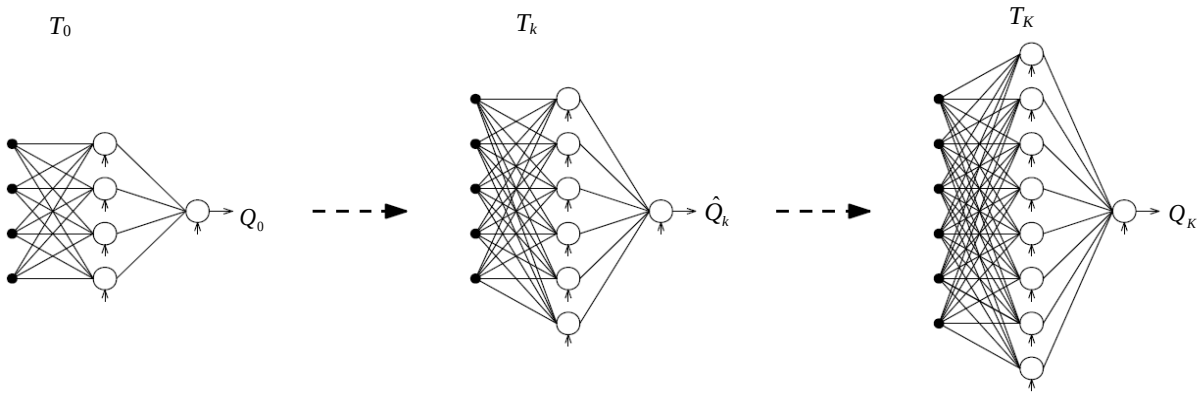

Fig. 5: FA structure evolution

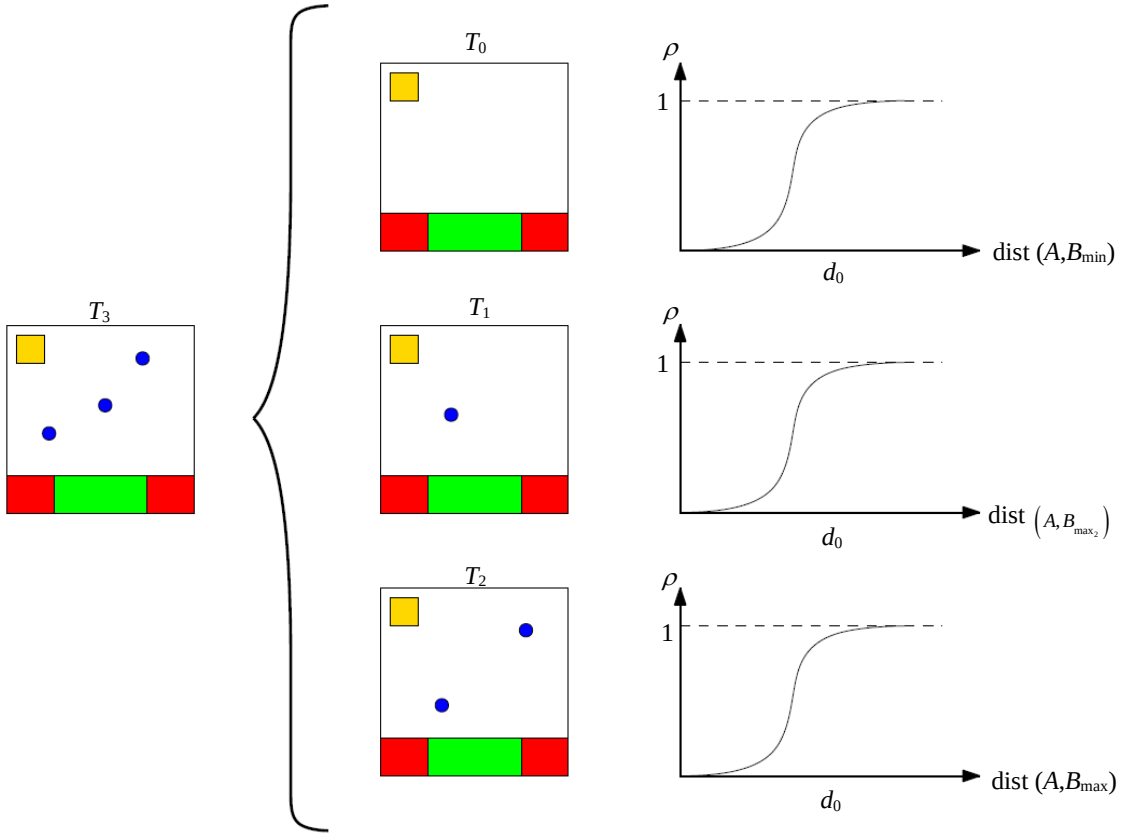

Fig. 6: Task similarity functions used in the transfer to T_3. Depending on the source tasks, the similarity function measures how similar two tasks are. In the case where T_0 is the source task the similarity is evaluated through the distance from the agent to the closer attacker in T_3 called dist (A, B_{min}). In the case where T_1 is the source task the similarity is evaluated through the distance from the agent to the second closer attacker in T_3 called dist (A, B_{max2}). For each case, if this distance is larger the two tasks are similar and the similarity value r is closer to one

DOI: 10.3844/jcssp.2020.1465.1476

Analysis

We focus our analysis in the experiments where T_3 is the target tasks. Each transfer experiment T_{k-1} to T_k was run 15 times. The Q value function transferred corresponds to an agent whose performance was the average of the 15 runs. From this agent, we obtain also the transfer rate ϕ. Table 3 shows the detailed performance for the transfer in the curriculums.

Figure 7 shows the agent winning probability at the end of the training for the experiments in which T_3 was the target task. All the experiments show an increase in their performance for the transfer rate ϕ = 0.1. Particularly important at lower transfer rates is the performance for experiment E_9 which contains T_0 and T_3, which reveals that a single task can improve the performance considerable if this task contains relevant knowledge that can be shared across all the curriculum. In this case, T_0 is the source for the knowledge related to the goal location. This is more evident in Table 3, where can be seen that the best performances for T_3 are for the experiments that included T_0 in the curriculum. Although E_{10} and E_{12} also contain only two tasks, the performance in these experiments is not as good as the E_9. The knowledge in E_{10} and E_{12} is more complex due to the presence of the attackers and is not shared over all the tasks.

The agent with the best performance corresponds to the E_{15} whose curriculum contains all the tasks.

Figure 8 shows the mean reinforcement in the final part of the curriculum for the experiments in which T_3 was the target task. As expected, these results show a similar behavior to the winning probabilities. However, for the transfer rate ϕ = 0.9 the agent is not learning, since 90% of the time is using the previous policy and is exploring in the other 10%. In this case, the performance is highly dependent in the previous knowledge transferred to the agent and the experiment performances are perfectly ordered according to Table 1, which indicates that as more complete is the curriculum the better the performance will be.

In Fig. 7 and 8 we see a decrease in the agent performance as the transfer rate increase. This decrease is a signal that negative transfer is occurring and that the selection of transfer must be done carefully to avoid it. Also, in these figures the agent performance in E_8 was plotted to visually compare the effectiveness of our strategy in an equal time setting. E_{15} takes 300 batches from T_0 and 6000 batches for T_1, T_2 and T_3 for a total of 18300 batches. The curriculum in E_{15} was able to accomplish a better final performance than the one obtained when learning T_3 from scratch even for the transfer rate ϕ = 0.9 and for a similar number of batches.

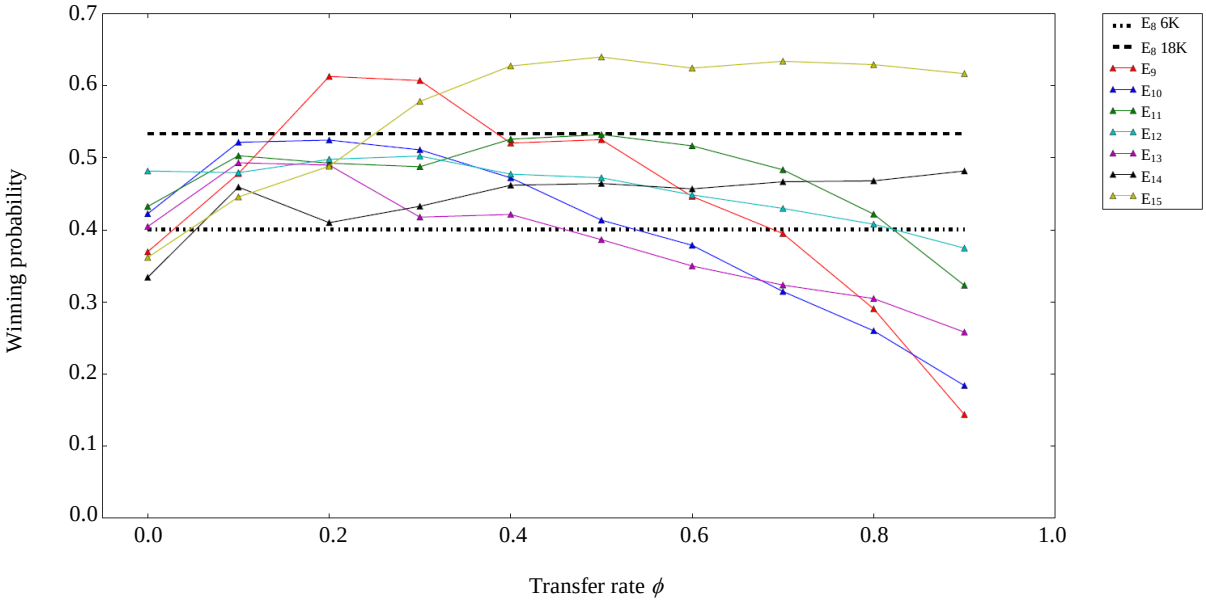

Fig. 7: T_3 agent's winning probability. The horizontal axis corresponds to the transfer rate used in the last part of the curriculum, the one used in the transfer between the last two tasks. The lines are used to highlight the trend with respect to ϕ since it only took discrete values [0: 0,0:1,0:2,…, 0: 9]. Each point was averaged over 15 runs. The black dashed lines correspond to the agent performance after training in experiment E_8 which contains only one task T_3, learning from scratch. The dot dashed line corresponds to the agent performance after a 6000 batches training and the dashed line after 18000 batches

DOI: 10.3844/jcssp.2020.1465.1476

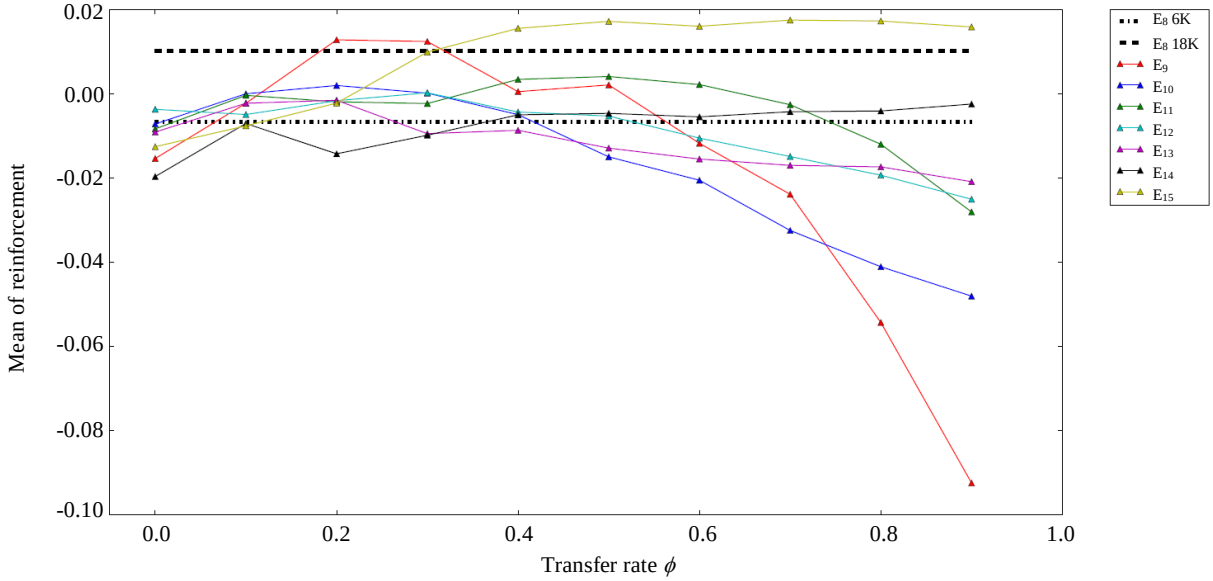

Fig. 8: T_3 agent's mean reinforcement. The horizontal axis corresponds to the transfer rate used in the last part of the curriculum, the one used in the transfer between the last two tasks. The lines are used to highlight the trend with respect to ϕ since it only took discrete values [0.0, 0.1, 0.2,…, 0.9]. Each point was averaged over 15 runs. The black dashed lines correspond to the agent mean reinforcement after training in experiment E_8 which contains only one task T_3, learning from scratch. The dot dashed line corresponds to the agent mean reinforcement after a 6000 batches training and the dashed line after 18000 batches

Table 3: Experiments results - Target task T_3

	Transfer rate			Winning probability			
				T_0	T_1	T_2	T_3
Experiment	0-1	1-2	2-3	$\varepsilon = 0.1$	$\varepsilon = 0.1$	$\varepsilon = 0.1$	$\varepsilon = 0.1$
E_8	-	-	-	-	-	-	0.4008 0.5336
E_9	-	-	0.2	0.9996	-	-	0.6130
E_{10}	-	-	0.2	-	0.4671	-	0.5246
E_{11}	0.4	-	0.5	0.9996	0.6574	-	0.5322
E_{12}	-	-	0.3	-	-	0.4593	0.5025
E_{13}	-	0.2	0.1	0.9996	-	0.5095	0.4931
E_{14}	-	0.1	0.9	-	0.4671	0.4542	0.4817
E_{15}	0.4	0.4	0.5	0.9996	0.6574	0.6002	0.6400

Conclusion

In this study, a new framework for transfer learning was presented. First, we show the importance of using a similarity function when learning in a sequence of tasks. The similarity function act as a memory unit that allows the agent to compare old experiences with new ones and exploit the acquired knowledge in similar states. Our evaluations confirmed that the use of the similarity function improves the agent's learning rate in new tasks compare to learning from scratch.

Moreover, we also show the importance of the presence of all the tasks into the sequence. Our experiments using different sequences, called curriculums, show that important knowledge is contained in every task. For this reason, an adequate construction of the sequence must be done to guarantee an effective transfer of the knowledge that will result in a better performance in the target task.

Additionally, by the experiments modifying the FA size for each task we observed that it is possible to devise a strategy to find an optimal FA structure for the target task (the complex or harder one) and a proper way to training it to obtain a more optimal one in less time than training it from scratch.

Finally, by using the proposed transfer strategy, which includes the sequence design and the similarity function, the performance of the agent was increased in terms of time and samples needed in the target task, hence, confirming the importance of using the similarity function to determine which a where to apply transfer during the learning.

Acknowledgement

Funding for this research was provided by the Administrative Department of Science, Technology and Innovation - Colciencias, Colombia.

Author's Contributions

Edwin Torres: Development of the main idea, experimental tests execution and text writing.

Fernando Lozano: Development of the main idea and general supervision and guidance.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Bou Ammar, H., Driessens, K., Eaton, E., Taylor, M. E., Mocanu, D. C., Weiss, G., & Tuyls, K. (2014). An automated measure of MDP similarity for transfer in reinforcement learning.

Bellman, R. (1957). Dynamic Programming. (1st Edn.), Princeton University Press, Princeton, NJ, USA.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009, June). Curriculum learning. In Proceedings of the 26th annual international conference on machine learning (pp. 41-48).

Boyan, J. A. (1999, June). Least-squares temporal difference learning. In ICML (pp. 49-56).

Bradtke, S. J., & Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. Machine learning, 22(1-3), 33-57.

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In Advances in neural information processing systems (pp. 369-376).

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In Machine Learning Proceedings 1995 (pp. 30-37). Morgan Kaufmann.

Carroll, J. L., & Seppi, K. (2005, July). Task similarity measures for transfer in reinforcement learning task libraries. In Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. (Vol. 2, pp. 803-808). IEEE.

Caruana, R. (1997). Multitask learning. Machine learning, 28(1), 41-75.

Drummond, C. (2002). Accelerating reinforcement learning by composing solutions of automatically identified subtasks. Journal of Artificial Intelligence Research, 16, 59-104.

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. Journal of Machine Learning Research, 6(Apr), 503-556.

Fernández, F., & Veloso, M. (2006, May). Probabilistic policy reuse in a reinforcement learning agent. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (pp. 720-727).

Ferns, N., Castro, P. S., Precup, D., & Panangaden, P. (2012). Methods for computing state similarity in Markov decision processes. arXiv preprint arXiv:1206.6836.

Ho, Y., Bryson, A., & Baron, S. (1965). Differential games and optimal pursuit-evasion strategies. IEEE Transactions on Automatic Control, 10(4), 385-389.

Isaacs, R. (1999). Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization. Courier Corporation.

Konidaris, G., Scheidwasser, I., & Barto, A. G. (2012). Transfer in reinforcement learning via shared features. The Journal of Machine Learning Research, 13(1), 1333-1371.

Kumar, M. P., Packer, B., & Koller, D. (2010). Self-paced learning for latent variable models. In Advances in neural information processing systems (pp. 1189-1197).

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. Journal of machine learning research, 4(Dec), 1107-1149.

Lazaric, A. (2008). Knowledge transfer in reinforcement learning (Doctoral dissertation, PhD thesis, Politecnico di Milano).

Liu, Y., & Stone, P. (2006, July). Value-function-based transfer for reinforcement learning using structure mapping. In Proceedings of the national conference on artificial intelligence (Vol. 21, No. 1, p. 415). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Madden, M. G., & Howley, T. (2004). Transfer of experience between reinforcement learning environments with progressive difficulty. Artificial Intelligence Review, 21(3-4), 375-398.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. arXiv preprint arXiv:2003.04960.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.

Parsons, T. D. (1978). Pursuit-evasion in a graph. In Theory and applications of graphs (pp. 426-441). Springer, Berlin, Heidelberg.

Piaget, J. (1963). The origins of intelligence in children. The Norton library, N202. Norton.

Puterman, M. L. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. (1st Edn.), John Wiley and Sons, Inc., New York, USA.

Riedmiller, M. (2005, October). Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In European Conference on Machine Learning (pp. 317-328). Springer, Berlin, Heidelberg.

Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems (Vol. 37, p. 20). Cambridge, UK: University of Cambridge, Department of Engineering.

Sutton, R. S., & Barto, A. G. (1998). Introduction to reinforcement learning (Vol. 135). Cambridge: MIT press.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057-1063).

Taylor, M. E., Kuhlmann, G., & Stone, P. (2007a, September). Accelerating search with transferred heuristics. In ICAPS Workshop on AI Planning and Learning.

Taylor, M. E., Stone, P., & Liu, Y. (2007b). Transfer learning via inter-task mappings for temporal difference learning. Journal of Machine Learning Research, 8(Sep), 2125-2167.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research, 10(7).

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first?. In Advances in neural information processing systems (pp. 640-646).

Torrey, L., Walker, T., Shavlik, J., & Maclin, R. (2005, October). Using advice to transfer knowledge acquired in one reinforcement learning task to another. In European Conference on Machine Learning (pp. 412-424). Springer, Berlin, Heidelberg.

Tsitsiklis, J. N., & Van Roy, B. (1997). Analysis of temporal-diffference learning with function approximation. In Advances in neural information processing systems (pp. 1075-1081).

Tanaka, F., & Yamamura, M. (1997, August). An approach to lifelong reinforcement learning through multiple environments. In 6th European Workshop on Learning Robots (pp. 93-99).

Vygotsky, L. S., & Kozulin, A. (1962). Thought and Language. (1st Edn.), MIT Press,

Watkins, C. J. C. H. (1989). Learning from delayed rewards.

Weinshall, D., Cohen, G., & Amir, D. (2018). Curriculum learning by transfer learning: Theory and experiments with deep networks. arXiv preprint arXiv:1802.03796.

White, A., Modayil, J., & Sutton, R. S. (2012, November). Scaling life-long off-policy learning. In 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL) (pp. 1-6). IEEE.

Wilson, A., Fern, A., & Tadepalli, P. (2012, June). Transfer learning in sequential decision problems: A hierarchical Bayesian approach. In Proceedings of ICML Workshop on Unsupervised and Transfer Learning (pp. 217-227).