Multi-agent Transfer Learning in Reinforcement Learning-based Ride-sharing Systems

Alberto Castagna and Ivana Dusparic
School of Computer Science and Statistics, Trinity College Dublin, Ireland

Keywords: Reinforcement Learning, Ride-sharing, Transfer Learning, Multi-agent.

Abstract: Reinforcement learning (RL) has been used in a range of simulated real-world tasks, e.g., sensor coordination, traffic light control, and on-demand mobility services. However, real world deployments are rare, as RL struggles with dynamic nature of real world environments, requiring time for learning a task and adapting to changes in the environment. Transfer Learning (TL) can help lower these adaptation times. In particular, there is a significant potential of applying TL in multi-agent RL systems, where multiple agents can share knowledge with each other, as well as with new agents that join the system. To obtain the most from inter-agent transfer, transfer roles (i.e., determining which agents act as sources and which as targets), as well as relevant transfer content parameters (e.g., transfer size) should be selected dynamically in each particular situation. As a first step towards fully dynamic transfers, in this paper we investigate the impact of TL transfer parameters with fixed source and target roles. Specifically, we label every agent-environment interaction with agent's epistemic confidence, and we filter the shared examples using varying threshold levels and sample sizes. We investigate impact of these parameters in two scenarios, a standard predator-prey RL benchmark and a simulation of a ride-sharing system with 200 vehicle agents and 10,000 ride-requests.

1 INTRODUCTION

Reinforcement learning (RL) has shown good performance in addressing a variety of tasks, from naive games to more complicated problems that require synchronization. RL enables an intelligent agent optimize its performance in a specific task, however, when a change in the underlying environment or a task occurs, the performance of an agent often sharply decrease. Thus, RL often performs very well in a simulated environment but struggles in real world evolving environments since it requires a constant updating of the knowledge.

To discuss issues of applying RL in such real world applications, in this work we apply multi-agent RL for the ride-request assignment in a Mobility On-Demand (MoD) system. Ride-request assignment is a widely studied task, addressed as global optimization problem (Fagnant and Kockelman, 2016; Wen et al., 2017; Liu and Samaranayake, 2019) or locally by distributing the control at vehicles level (Castagna et al., 2020; Castagna et al., 2021; Guériau et al., 2020; Al-Abbasi et al., 2019). Recent works adopt the use of RL where an agent controls a single vehicle, as we do in this paper. The type of changes that RL might need to adapt to in an MoD system, but which have not been addressed in literature, can, for example, be that road layouts change, e.g., due to closure, or that demand load or pattern changes over time, temporarily or permanently.

A possible solution to these issues is to apply transfer learning (TL) in conjunction with RL. Once encounters a new situation, an agent can reuse knowledge from other agents or its previous collected experiences in order to boost its initial performance in the new situation. For example, MoD system in (Castagna et al., 2021) replicates full knowledge from one agent to others. When a new agent joins the fleet, it can inherit knowledge good enough to be productive with no delay. Another approach to transferring knowledge across similarly defined tasks is by fine tuning previous performance; this process replicates the knowledge from one agent to the other and afterwards, the receiving agent performs a refinement step to tune the received knowledge in its own settings and environment. As an example, (Wang et al., 2018b) proposes fine-tuning in rider-driver assignment problem across scenarios with different underlying dynamics, e.g., demand patterns and road structure. However, fine-tuning protocols are usually ap-

Castagna, A. and Dusparic, I.
Multi-agent Transfer Learning in Reinforcement Learning-based Ride-sharing Systems.
DOI: 10.5220/0010785200003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 2*, pages 120-130
ISBN: 978-989-758-547-0; ISSN: 2184-433X

plied when the differences are relatively small. The time required for the receiving agent to adapt the received information is related to the gap magnitude between source and destination conditions. As this become wider, as more time the agent needs to refine the knowledge.

Transferring from a source to a very different target task, could lead to so-called negative transfer, as result the second agent requires longer time to solve the task compared to an agent that is learning from scratch. When negative transfer happens, on top of the time needed to solve the task, receiving agent need additional time to forget the wrong injected knowledge. Therefore, while TL can significantly improve the performance of an RL system, the knowledge transferred needs to be carefully selected and integrated, to prevent negative transfer resulting in even worse performance than without deploying TL.

TL can also be applicable when multiple agents are learning a task simultaneously. By exploiting transfer techniques, agents can collaborate to achieve a faster convergence rate. When applied in real-time, further challenges need to be addressed, such as selecting the best agent as source of transfer among those available.

As the first step towards this vision in which agents in a multi-agent system continually share the knowledge with evolving parameters, in this paper we study the impact of parameters used when selecting the knowledge to be transferred, but in an offline transfer scenario, where the roles of sender and receiver are well defined. Specifically, we bootstrap the agent with agent-environment interactions collected from other agents across similar scenarios. Transferred knowledge is annotated with a value depicting agent's confidence in that particular experience, as in (Ilhan et al., 2019; Silva et al., 2020b). We study TL performance using different batches of experiences, varying quantity of information shared as well as its ”quality”, as determined by adjusting the confidence threshold level.

We perform the transfer across agents performing the same task, but in completely different underlying environment dynamics. The target agent has to perform a full training cycle in the novel environment, but prior to that it inherits knowledge from another trained agent in order to bootstrap the performance achieved.

In our initial investigations presented in this paper, we focus on a simple case where just two agents are involved and the underlying learning model is represented by a neural network for both. Transfer goal focuses on improving the performance on receiving side and therefore, the communication is one way.

For our implementation, we combine Proximal Policy Optimization (PPO) (Schulman et al., 2017), with external knowledge in form of agent-environment interactions. Unlike off-policy algorithms that use two different policies to sample and optimize, on-policy PPO assumes that the experience used to optimize the policy is drawn by the very same policy. Therefore, we extend PPO to bootstrap external information to perform few steps of PPO gradient descent optimization before exploration begins.

Evaluation is performed in two application areas: benchmark environment predator-prey and ride sharing simulation. The latter experiments are carried out on a simulated MoD environment with ride-sharing enabled vehicles. Requests are obtained from New York City dataset (NYC Taxi and Limousine Commission, 2020) and simulation performed in the Simulator of Urban MObility (SUMO). Finally, we compare the performance against baselines: (1) policy transfer and (2) learning from scratch.

Contribution of this paper is therefore twofold: 1) we evaluate the impact of transferring confidence labelled agent-environment interactions by varying the transfer batch size and filtering using different threshold level; 2) we study applicability of TL in a Mobility-on-Demand system with ride-sharing (RS) enabled vehicles under different demand patterns.

The rest of this paper is organised as follows. Section 2 reviews relevant work related to experience reusing in transfer learning applied to RL. Section 3 describes in detail the process used to share experience within this research, Section 4 presents the scenarios, Section 5 discusses the results and discuss the findings while Section 6 concludes this manuscript by giving further research directions.

2 BACKGROUND

This section introduces the reader to previous related work on transfer learning and on the rider-driver matching problem for mobility on-demand system, finally discusses available methodologies to estimate agent's confidence.

2.1 Rider-driver Matching Problem

Rider-driver matching problem is addressed using a wide range of both centralized and decentralized techniques, with different inputs and assumptions. Generally, centralized aims to directly optimize the global performance, i.e., minimising the travelled distance (Liu and Samaranayake, 2019; Fagnant and Kockelman, 2016; Wen et al., 2017) while decentralized addresses the problem from a single vehicle perspective where each vehicle is independent and unaware of others doing. Decentralized approaches often rely on RL where each vehicle is controlled by a single agent (Castagna et al., 2020; Castagna et al., 2021; Gueriau and Dusparic, 2018; Guériau et al., 2020; Al-Abbasi et al., 2019).

Transfer Learning in Rider-driver Assignment. Within rider-driver dispatch problem there are not many works that bootstrap external knowledge applying TL. Previously, (Castagna et al., 2021) designed a multi-agent collaborative algorithm to obtain a main knowledge that is then replicated to other agents. In detail, multiple agents were learning to satisfy ride-requests in a iterative way where the set of available requests was composed by unserved requests from previous vehicles. Finally, obtained knowledge is replicated to other agents and no further refinement steps are performed.

On the other hand, (Wang et al., 2018a; Wang et al., 2018b), bootstrap previous acquired knowledge with a fine-tuning process to leverage a pre-trained learning model. Here, authors use a deep learning model and transfer a pre-trained neural network to target environments with different demand pattern and road infrastructure. This technique limits the range of applicable algorithms to those with a neural network underlying as it is feasible to transfer just the layers weights, i.e., is not possible to transfer knowledge to tabular model. Furthermore, any change in the neural network architecture require further steps to adapt the pre-trained layers.

(Wan et al., 2021) proposes an approach to transfer from source to a target environment. The goal is to fine-tune knowledge by adapting the source value-function to a target task with a concordance penalty that expresses the similar patterns between the two tasks, i.e. demand pattern. Thus, it can be applied to a range of RL techniques regardless the underlying representation. Nevertheless, a strong limitation of this work is the need of prior knowledge over the relation between source and target tasks.

2.2 Transfer Learning

Transfer learning is a widely used technique in a variety of fields, (Zhuang et al., 2020) provides an overview of the state of the art approaches across many domains. In this work we apply TL from agent to agent. We analyse the two agents case and from now on, we refer to the agent that is providing information as source agent, while the other as target agent. The latter processes external knowledge to boost the performance within its task.

When transferring across agents, tasks are normally equally defined, therefore state representations, action sets and reward models are the same. Despite these similarities, there can be some intrinsic characteristics of the environment that may differ, e.g., within MoD scenario, different configurations of the road network, different traffic loads or different demand pattern.

Existing applications of TL applied to RL are many, varying by type of transferred knowledge and technique used. For example, *experience sharing* where agents share experience in form of agent-environment interactions (de la Cruz Jr et al., 2019; Lazaric et al., 2008) as we do within this work, *policy transfer* where a trained model is replicated to a novel agent as in (Wang et al., 2018b), *advice-based* where an expert or another agent is available on-demand to support the learning phase of an agent (Fachantidis et al., 2017; Silva et al., 2020a), and finally by more sophisticated and task-related techniques as in (Wan et al., 2021). Previous work also investigated the impact of transfer parameter selection in the context of tabular RL (Taylor et al., 2019), and concluded that frequency and size of the transfer, as well as how is the knowledge incorporated on the target agent, has significant impact on the performance of the target agents.

2.3 Confidence Estimator

For estimating the agent's confidence within a state are available several possibilities that suit tabular and deep learning models. A naive approach could be to count the agent visits over states or to define a function over it as in (Zhu, 2020). However, this becomes unfeasible whenever the state space is too wide or continuous as in our MoD scenario. More sophisticated models range from the use of a Gaussian model with a confidence function defined over it (Taylor, 2018), changes on neural network structure (Silva et al., 2020a) and finally using external tool to compute it as Random Network Distillation (RND) (Burda et al., 2018). RND has been presented as a tool to regulate exploration by curiosity but has been already used as uncertainty estimator for a RL agent by (Ilhan et al., 2019). A neural network is trained to predict the output of a target rough network, while uncertainty is defined as discrepancy across the two networks output.

Algorithm 1: Process for source agent that stores agent-environment interactions.

```
Initialize experience buffer $EB$
for $e$ in episodes do
    repeat
        observe state $s$
        take an action $a$
        observe action, reward $r$ and next state $s'$
        if time to update policy then
            optimize policy for K epochs on collected interactions
            optimize RND on visited states
        end if
        if $ep >$ episode from saved experience then
            estimate agent confidence $u$ over $s$ through RND
            add $(s, a, r, s', u)$ to buffer $EB$
        end if
    until episode is complete or max steps is reached
end for
```

3 CONFIDENCE-BASED TRANSFER LEARNING

Within this section we introduce the design of transfer learning approach we have implemented to evaluate confidence and sample size impact in transfer learning.

We assume that agents explore at a different time the same or similar tasks (but performed in a different underlying environment). Thus, no knowledge mapping is required as both agents have the same set of sensors and actuators. Algorithm 1, describes the process of source agent that collects samples, represented as a tuple: $(s_t, a_t, r_t, s', u_{s_t})$, hence stores agent-environment interactions plus a confidence value computed over the visited state. The latter value depicts state-related agent epistemic confidence at time t of the visit. In detail, u stands for uncertainty and therefore, higher is the value lower is the agent's confidence within that state.

Once the first agent has accomplished the training, buffer of collected experience is transferred to a receiving agent. The latter filters the received knowledge based on interaction's confidence u and confidence threshold t and draws a certain number of samples to pre-train its learning model as in Algorithm 2.

This technique is applicable to RL algorithms regardless the underlying representation, e.g., tabular model or through a neural network. Our implementation uses policy-based deep RL algorithm called proximal policy optimization (PPO) (Schulman et al., 2017). PPO is an on-policy algorithm that optimises a policy by gradient descent, meaning that the updated policy and the one used to produce samples are the same. Therefore, to not compromise the optimisation process, we limit the bootstrapping of external knowledge before the learning phase commence. While off-policy algorithms might be naturally more suitable for reusing experience, in this paper we based it on PPO, due to faster convergence rates and its previous successes in ride-sharing application (Castagna et al., 2021).

Algorithm 2: Flow of target agent that samples interactions.

```
**Input:** experience buffer $EB$, threshold $t$, transfer budget $B$
Initialize agent
sample from $EB$ $B$ interactions where $u < t$
train agent with experience sampled
for $e$ in episodes do
    repeat
        observe state $s$
        take an action $a$
        observe action, reward $r$ and next state $s'$
        if time to update policy then
            optimize policy for K epochs on collected interactions
        end if
    until episode is complete or max steps is reached
end for
```

4 EVALUATION ENVIRONMENTS

This section introduces the scenarios used to perform evaluation, and the parameters used in these scenarios.

4.1 Predator-prey

Predator-prey environment is based on an existing version of a grid-world implementation (Chevalier-Boisvert et al., 2018). Figure 1 shows a basic configuration of the environment. Task starts with a random allocation of predator and preys around the grid, while ends whether the predator, represented as a front-oriented filled triangle, catches all the same colour-based preys, depicted as pierced triangles. The environment is partially observable as the agent knows only the 3 x 3 grid in its field of vision, which in the figure is highlighted in front of it. To fulfil the goal, a predator is enabled to move forward, turn left or right, wait and perform a catch action. With the latter, a

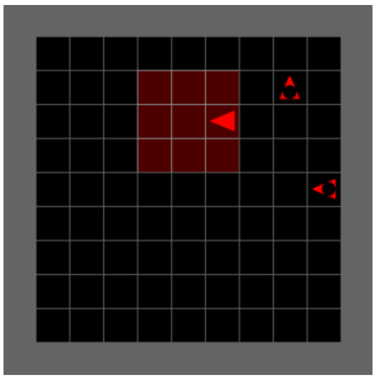


Figure 1: Instance of 9x9 grid predator prey environment with a single predator (filled triangle), and multiple preys (pierced triangles). Predator perceives the highlighted 3x3 grid in front of it and its goal is to catch the preys.

predator can catch a prey when it is in the consecutive cell.

Prey follows a random policy to escape from the predator, as follows: stay in the same position with 10% of probability, turn left or right with 25% chance each, and move forward with 40% probability.

The reward model is designed to encourage predator to catch the prey faster; the agent receives a living penalty according to the action taken, -0.01 for turn and step actions, -0.25 when staying still and finally -0.5 for a missed catch. However, whether the catch succeeds agent receives a positive reward of 1. The configuration used within this work is reported in Table 1.

For evaluation purpose, we use this benchmark scenario to study the impact of sharing agent-environment interactions from a pre-trained agent to a new "blank" agent. Source of transfer is an agent that performs well enough to accomplish the given task. During exploration, this agent labels interactions with its epistemic confidence over the visited state. Thus, confidence is estimated on the number of state visits updated to the timestep of the sampling. Afterward, knowledge is filtered by a specified confidence threshold and transferred to the experience buffer of the new agent. The new agent samples a number of interactions from the buffer and pre-trains its learning model before exploring.

We study the impact of leveraging collected experience by varying threshold confidence level and number of interactions sampled. By doing so, we aim to gain insights into how is the second agent behaviour influenced by the amount of samples passed and by the quality of injected knowledge.

4.2 Ride-sharing-enabled Mobility on Demand Scenario

For the real world case study we use the system presented in (Castagna et al., 2021). However, we port the implementation into a widely used traffic simulator of urban mobility, SUMO (Lopez et al., 2018), to increase realistic traffic movement and conditions.

Agent's goal is to maximise the amount of ride-request served. To serve all the requests, a fleet of 5-seater RS-enabled vehicles is dispatched. Control is distributed at vehicle level and therefore, each car is controlled by a single agent with no communication nor coordination with others.

At each timestep, each RL vehicle agent knows its location, number of unoccupied seats and destination of the request that can be served the quickest, among those that it has assigned. Furthermore, an agent receives information about the three closest requests in the neighbourhood, with an number of passengers that could fit on board and that could be picked up before request's expiration time. A representation of state and perception is shown in Figure 2. For each of these requests, agent knows request origin, request destination, number of passengers and minimum detour time from the current route to reach the pick-up point and destination point of the request.

After evaluating state and perception, an agent can take one of the 5 actions available, (1) *being parked*, vehicle stays parked for a predefined amount of time, (2) *drop-off*, vehicle drives towards its destination and finally, (3) *pick-up*, where agent drives to pick-up point of the selected request. Note that agent can decide to pick one of the three available requests and as results, agent has three different pick-up options, resulting in total of 5 available actions.

Vehicle position is updated in real-time, however decision process is triggered on the conclusion of an action, i.e, vehicle arrives to pick-up or drop-off location. On top of that, a vehicle can evaluate to pick further requests while is serving others. This evaluation step is allowed whenever the vehicle is further from arrival point and has at least a free seat. All vehicles within the fleet are synchronized under the same clock, therefore evaluation process is performed following a FIFO queue. Given the task's nature, an episode begins with the vehicle spawn and terminates when there are no further request to be served.

Reward scheme is designed to minimise the cumulated delay for each of the action taken by the agent. Therefore, is defined as the elapsed time from the agent's decision to the end of the action. First, when an action is not possible to accomplish, agent receives a penalty of -1. Second, when agent decides of being

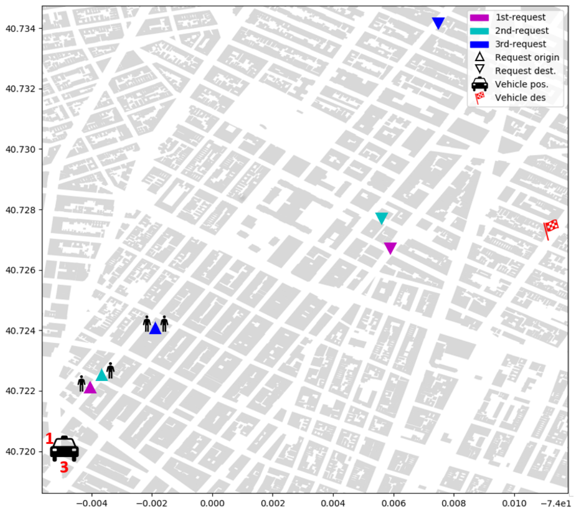


Figure 2: Representation of vehicle status with the three requests perceived in the neighbourhood.

parked, receives a penalty defined over the elapsed time: $\frac{-x}{x+delay}$, where x is set to 5 whether agent is not serving any request, else to 1. Third, while dropping off, reward is defined as follow, $\exp\left(\frac{1}{1+delay}\right)$. Last, on pick-up reward given to an agent is defined as $\frac{x}{x+delay}$,where x is fixed to 1 whether the request picked is the first, 2 otherwise. Reward model has been designed in that way to normalize delay into (0,1) interval.

To emulate a real-world scenario, we use ride-requests data from (Guériau et al., 2020), where authors have aggregated NYC taxi trips from 50 consecutive Mondays between July 2015 and June 2016 in Manhattan zone (NYC Taxi and Limousine Commission, 2020). Trips are then organized on a time-base schedule. Finally, we obtained 4 datasets representing peak times, morning, afternoon evening and night. Within this research, we use the morning peak slot (from 7 to 10 am) and the evening (from 6 to 9 pm).

We use these different datasets to evaluate the impact of sharing experience among same tasks but varying the underlying demand pattern. We let agents explore and collect interactions by serving requests from the morning set and afterwards we leverage part of the collected experience to pre-train agents that operate in the evening shift. In detail, 200 vehicles are dispatched during simulation to serve the demand. To evaluate our work, we use two demand loads and we execute the system in the following conditions, 1) train and test on morning peak; 2) train and test on evening peak; 3) train on morning peak and test on evening peak; lastly, 4) train on morning peak, transfer knowledge to a new agent that will then be further trained and tested on evening peak.

5 EVALUATION RESULTS AND ANALYSIS

This section presents and discuss the results achieved through simulations in the two evaluation environments and is organized in two parts. First, we discuss the benefit of reusing experience in predator-prey scenario and second, in MoD scenario.

5.1 Predator-prey

To evaluate the performance obtained within predator-prey scenario we studied agent's learning curve, by analysing the reward.

For the following set of experiments, Table 1 summarises the configuration of the environment, while Table 3 reports the simulation parameters for the task. We set RND size to 1024 as we discovered to be a good trade off between performance and network size.

Within this work, we let the source agent to collect experience from the last 20% episodes and we vary the number of drawn interactions from the transfer buffer (5,000 and 10,000) and confidence threshold for filtering interactions. Figure 3 depicts the achieved results by varying both parameters. As threshold, we used few hardcoded task-related values empirically obtained (0.015, 0.02, and 0.05) and two others computed over the transfer buffer (mean and median confidence values).

In addition, we compare achieved results against four different baselines and we show the results through Figure 4. As first baseline, we propose a no-transfer agent, i.e., an agent that addresses the task for the first time with no external support. Other baselines included an additional agent that can support the learning phase by providing advice when needed. As transfer framework for these baselines, we opted for the teacher-student paradigm (Torrey and Taylor, 2013), where student is represented by the learning agent while teacher is the additional agent that already accomplished the task obtaining good performance. Among the transfer-enabled baselines we can distinguish three cases: first *advice at beginning*, as we noticed that the initial episodes are the hardest for an agent to exhibit good behaviour, second *mistake correction*, where teacher supervises student by providing advice when the latter misbehaves according to teacher's expectation and last, *confidence based* $\varepsilon-$*decay* (Norouzi et al., 2020), where student asks for advice following an $\varepsilon-$decay probability when it has lower confidence than teacher. In all teacher-supported baselines, as is the standard in teacher-student TL related work, we set a maximum budget for exchanging advice in order to match the number of sampled interactions.

Figure 3 shows the results achieved by varying filter threshold on receiving side organized by the amount of sampled interactions, 5,000 (*a*) and 10,000 (*b*). Impact of threshold is mild compared to transfer buffer size; contrary to what we intuitively expected, using less samples (5,000) has shown a greater improvement of the performance against a no transfer enabled agent. Regardless of the transfer settings, agent performance is overall better compared to a no transfer agent. In a few of the cases we observe a jumpstart, i.e., an improved performance since the very beginning. However, we cannot do a linear comparison between transfer and no-transfer results because the former exploits external knowledge resulting in a few additional episodes. Nevertheless, most of transfer-enabled instances are able to outperform a no transfer agent since the beginning as we can easily notice from Figure 3a.

Tuning the right threshold could be very expensive in term of time as it is task-dependent and might not results in the expected outcome (i.e., resulting in a negative transfer). Nevertheless, Figure 4 shows a significant gap between our proposal and the evaluated baselines. In particular, we can notice from Figure 4*a* that when the transfer budget is low, our proposal stands out from the others and enables the agent to reach very soon good performance that are kept over time.

Table 1: Parameters in predator-prey environment.

Parameter	Value
grid size	9
#teams	1
#predators	1
#preys	2
grid-view size	3
rewards	
successful catch	1
missed catch	-.5
hold position	-.25
turn or step penalty	-.01

5.2 Mobility on Demand

For this scenario we adopted a MoD previously evaluated against multiple baselines with and without ride-sharing and rebalancing (Castagna et al., 2021), therefore in this work we solely focus on evaluating the impact of TL. Table 2 presents the environment related settings while, Table 3 reports the learning model configuration.

We rely on common metrics used to evaluate per-

Table 2: Parameters in Mobility-on-Demand scenario.

Parameter	Value
fleet size	200
#seats	5
#available requests 7-10am	9663
#available requests 6-9pm	9662
training rounds	6
training vehicle per round	10

Table 3: Simulation settings for predator-prey (PP) and Mobility-on-Demand (MoD) scenarios.

Parameter	PP	MoD
# episodes	3000	N/A
max #steps	500	N/A
#hidden layer(s)	1	4
size hidden layer(s)	64	128
gamma	.99	.999
learning rate	.001	1e-4
update_iter	500	32
epochs	10	10
$\varepsilon-clip$	.2	.2
RND size	1024	1024

formances of MoD system. On passenger side, we evaluate waiting time, defined as time elapsed from moment that request is being made until the passenger pick-up by the ride-sharing vehicle. Furthermore, as Table 4 reports, we take into account detour ratio ($\overline{D_r}$) that captures the additional distance driven by the vehicle with the passenger over the expected distance without ride-sharing enabled. At the overall fleet level we evaluate percentage of requests being served in ride-sharing(*%RS*) and demand satisfaction rate (*% Served Reqs.*). Finally, on vehicle level we evaluate the passenger distribution across the available vehicles, reporting the variance of passengers distribution (σ *pass*) in Table 4, alongside the average travelled distance covered by vehicles during the whole simulation ($\overline{d_t}$).

We compare our TL-enabled MoD, *TL-enabled test 6-9pm* against two baselines:

1. *Train 7-10am test 6-9pm*, where we transfer the learnt model from an agent trained on the morning demand set (7-10am) to evening set (6-9pm)
2. *Train & test 6-9pm* where an agent learns over the evening set and is evaluated over the same dataset.

We also present the results achieved on the morning peak by agent trained on the same set of requests (*Train & test 7-10am*). The latter provides experience that we have used within our proposal where a novel agent, following Algorithm 2, preprocesses received knowledge and then performs a training on evening demand set (6-9pm).

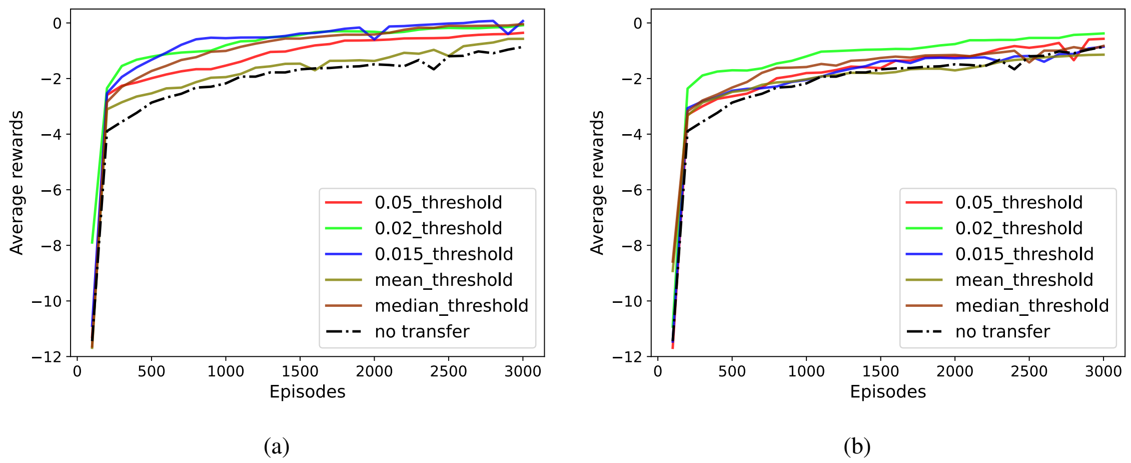


Figure 3: Comparison of averaged reward with different number of interactions sampled by the transfer buffer, 5.000 *(a)* and 10.000 *(b)*. Results are obtained as average over 50 simulations in predator prey scenario enabling transfer learning.

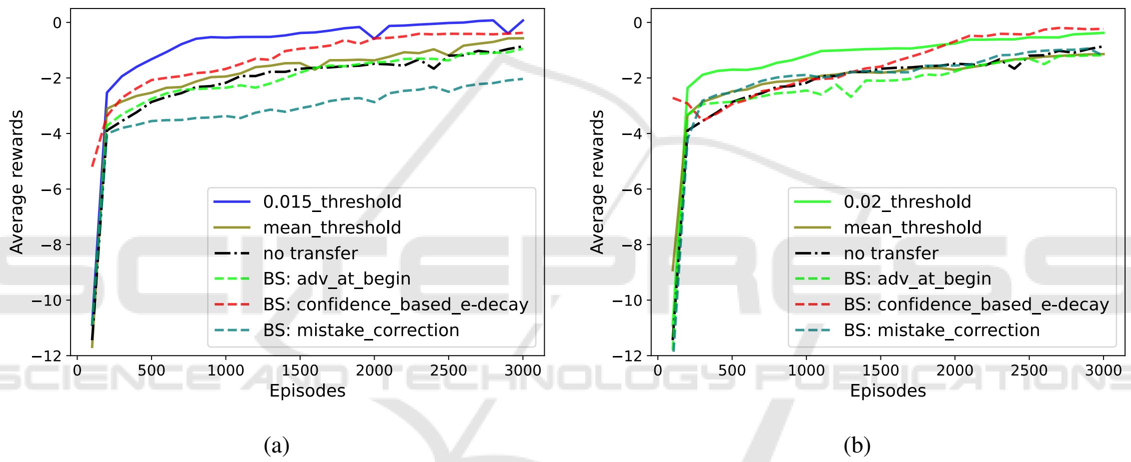


Figure 4: Comparison of best and worst instance against proposed baselines with different number of interactions sampled by the transfer buffer, 5.000 *(a)* and 10.000 *(b)*. Results are obtained as average over 50 simulations in predator prey scenario enabling transfer learning.

The details of how we conducted experience in our TL-based scenario *TL-enabled test 6-9pm* are as follows. To collect enough samples, agent training in *Train & test 7-10am* required a larger set of requests. Hence we used 30,000 requests maintaining the scheme of 6 rounds with 10 vehicles, forming a 60 episodes training. In detail, each vehicle selects requests to be served from the pool of requests that previous vehicles could not accomplish. Once a round is completed, request set is restored and a new cycle can begin. All vehicles contribute to the same model, which is later replicated to all the 200 vehicles operating in morning and evening shifts.

Agent was enabled to store knowledge only over the final 40% episodes and it collected approximately 14,000 interactions. Uncertainty interval for the collect interactions was really broad, 17 - 500,000, with average being 50,000 and with approximately 6,000 interactions having a state with no requests in agent neighbourhood and an associated uncertainty lower than 450.

Initially, we performed the tests with multiple thresholds applied to uncertainty level of interactions reused. First, we provided to target agent the whole set (approximately 14,000 interactions) without applying any filter. Second, we discarded all interactions composed by a state with no requests in proximity of the vehicle and from those we sampled 5,000 interactions. Third, we set the threshold to 50,000, utilizing only samples with uncertainty under that number, resulting in around 10,000 available interactions.

However, all of these initial tests failed to run to completion because they overloaded the SUMO engine with requests leading to a crash. Nevertheless, the early stages of the experiments were useful in order to tune the threshold. For the final experiment, results of which we present here we halve the available interactions for target agent by setting the threshold to 10,000 taking all the interactions with lower uncertainty. Agent randomly samples 5,000 interactions and pre-train its learning model. Finally, it performs a 6 rounds with 10 vehicles training.

We present the simulation results in Table 4 and Figure 5. We show the waiting time for requests in (*a*), the distribution of passengers served per vehicle in (*b*) and the travelled distance per vehicle in (*c*).

Note that green scenario, (*Train & test 7-10am*), reports a longer distance covered by vehicles due to a greater number of requests satisfied. Our scenario *TL-enabled test 6-9pm*, enable the system to cover further requests when compared to baselines on same demand set. As Table 4 reports, it enables the fleet to cover 79% of demand while *Train 7-10 test 6-9pm* and *Train & test 6-9pm* stop at 77% and 76%. Furthermore, despite the few additional requests served, average vehicles mileage is lowered by 5 km per vehicle (82 km) against (87km). Although, passengers distribution is unbalanced compared to baselines, as variance is almost doubled (41.12) against (21.65 and 19.35)

Eventually, while maintaining a fair RS rate (93) compared to baselines (99), our scenario almost halves the average request detour ratio (5.23) compared to baselines (9 and 9.11), resulting in less kilometres travelled onboard for passengers before reaching their destination.

When leveraging experience across equal-defined task as within our predator-prey simulations, target agent is enabled to outperform a no-transfer agent and baselines. In detail, TL-enabled agent overcomes no-transfer enabled agent performance in less than a half episodes.

On the other hand, when applying TL to a more complicated scenario, as our real-world MoD simulator, it demands a not trivial evaluation over transfer parameters. However, in our experiment, agents are able to satisfy a greater number of passengers while reducing travelled distance and improving rider experience. In fact, detour ratio, defined as distance travelled by passengers onboard of MoD vehicle against expected travelled distance on a solo trip, is sharply decreased when compared against other scenarios.

6 CONCLUSION AND FUTURE WORK

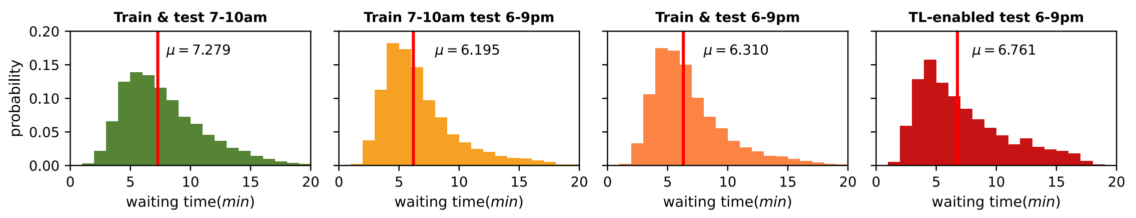
In this paper we presented an experience-sharing (transfer learning) strategy to leverage collected experience across same tasks executed in different environment dynamics. A first agent (source), collects samples throughout exploration while learning a task, that are then transferred to a second agent (target). The latter, preprocesses its learning model by sampling a certain amount of interactions from receiving buffer. We have analysed the performance varying transfer parameters in two scenarios, predator-prey, where target and source agent address the very same task, and a mobility on demand scenario where source and target are evaluated on a same task but under different underlying ride request demand.

We generally noticed that within the predator-prey scenario, by enabling TL, agent which receives transferred data is able to outperform a standard agent which learns from scratch.

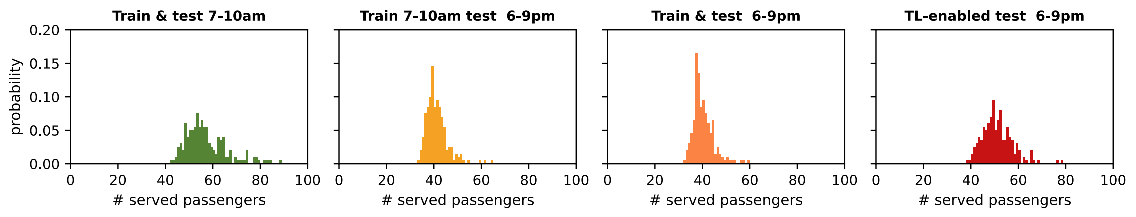
These improvements are more difficult to notice in a simulated real world environment since we do not track the reward achieved by the agents but we rely on domain-specific performance measures dependent on our application area. However, we can still notice an improvement in the ride-requests satisfied and in the distance travelled by the vehicles. When we compare TL-enabled agent against agent learning from scratch on the same set of demand, we can notice that ride-sharing ratio is slightly decreased in favour of a lower detour ratio. As a result, passengers lower the time spent onboard by reaching faster their final destination.

This paper presents a first step in a wider work on multi-agent experience sharing in real-time. In this research, we evaluated the impact and the feasibility of defining parameters and threshold in experience reusing with fixed roles of information sender and receiver.

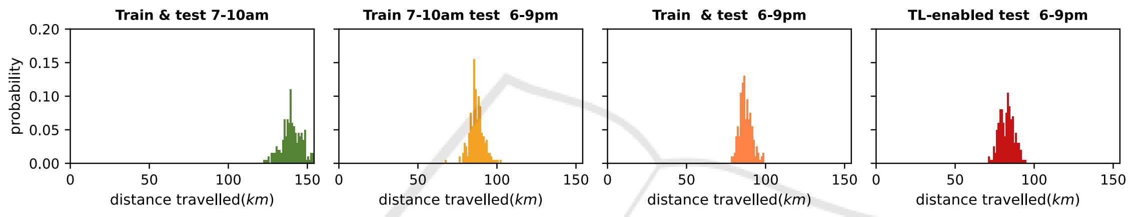
As a next step, we plan to evaluate ”importance” of the states for the target agent, in order to reduce transfers in less important states to save communication budget if needed. For example, we have observed that a source agent has high confidence in its knowledge about the state in which there are no requests to serve, but since there is less potential for ”mistake” actions when there are no requests, transferring knowledge in this situation might not be crucial. Afterward, we plan to bootstrap knowledge across scenarios with different road structure to further test the reliability of TL applied to simulation of real-world case and develop the full dynamic online multi-agent TL system.



(a) Waiting time for request



(b) Number of passengers served per vehicle



(c) Vehicles mileage

Figure 5: Comparison of simulation results for the analysed scenarios over the following metrics: (*a*) waiting time, (*b*) passengers distribution per vehicle and (*c*) distance travelled by vehicle. The *y*-axis represents the probability that a request for (*a*) or vehicle in (*b*, *c*) assumes a certain value. Note that results are obtained from a single simulation run.

Table 4: Performance metrics across all 5 evaluated approaches in ride-sharing scenario.

Scenarios	%Served requests	%RS	σ pass	$\overline{d_t}$ (km)	$\overline{D_r}$
Train & test 7-10am	93	93	76.61	140	9.4
Train 7-10am test 6-9pm	77	99	21.65	87	9
Train&test 6-9pm	76	99	19.35	87	9.11
TL-enabled test 6-9pm	79	93	41.12	82	5.23

ACKNOWLEDGEMENTS

This work was sponsored, in part, by the Science Foundation Ireland under Grant No. 18/CRT/6223 (Centre for Research Training in Artificial Intelligence), and Grant No. 16/SP/3804 (Enable).

REFERENCES

Al-Abbasi, A. O., Ghosh, A., and Aggarwal, V. (2019). Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4714–4727.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018). Exploration by random network distillation.

Castagna, A., Guériau, M., Vizzari, G., and Dusparic, I. (2020). Demand-responsive zone generation for real-time vehicle rebalancing in ride-sharing fleets. In *11th International Workshop on Agents in Traffic and Transportation (ATT 2020) at ECAI 2020*.

Castagna, A., Guériau, M., Vizzari, G., and Dusparic, I. (2021). Demand-responsive rebalancing zone generation for reinforcement learning-based on-demand mobility. *AI Communications*, pages 1–16.

Chevalier-Boisvert, M., Willems, L., and Pal, S. (2018). Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid.

de la Cruz Jr, G. V., Du, Y., and Taylor, M. E. (2019). Pre-training neural networks with human demonstrations for deep reinforcement learning.

Fachantidis, A., Taylor, M., and Vlahavas, I. (2017). Learning to teach reinforcement learning agents. *Machine Learning and Knowledge Extraction*, 1(1):21–42.

Fagnant, D. and Kockelman, K. (2016). Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, texas. *Transportation*, 45.

Guériau, M., Cugurullo, F., Acheampong, R. A., and Dusparic, I. (2020). Shared autonomous mobility on demand: A learning-based approach and its performance in the presence of traffic congestion. *IEEE Intelligent Transportation Systems Magazine*, 12(4):208–218.

Gueriau, M. and Dusparic, I. (2018). Samod: Shared autonomous mobility-on-demand using decentralized reinforcement learning. In *The 21st IEEE International Conference on Intelligent Transportation Systems*.

Ilhan, E., Gow, J., and Perez Liebana, D. (2019). Teaching on a budget in multi-agent deep reinforcement learning. pages 1–8.

Lazaric, A., Restelli, M., and Bonarini, A. (2008). Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 544–551, New York, NY, USA. Association for Computing Machinery.

Liu, Y. and Samaranayake, S. (2019). Proactive rebalancing and speed-up techniques for on-demand high capacity vehicle pooling. *CoRR*.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wiessner, E. (2018). Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582.

Norouzi, M., Abdoos, M., and Bazzan, A. (2020). Experience classification for transfer learning in traffic signal control. *The Journal of Supercomputing*.

NYC Taxi and Limousine Commission (2020). Tlc trip record data.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Silva, F., Hernandez-Leal, P., Kartal, B., and Taylor, M. (2020a). Uncertainty-aware action advising for deep reinforcement learning agents.

Silva, F. L. D., Hernandez-Leal, P., Kartal, B., and Taylor, M. D. (2020b). Uncertainty-aware action advising for deep reinforcement learning agents. In *AAAI*.

Taylor, A., Dusparic, I., Guériau, M., and Clarke, S. (2019). Parallel transfer learning in multi-agent systems: What, when and how to transfer? In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Taylor, M. E. (2018). Improving reinforcement learning with human input. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5724–5728. International Joint Conferences on Artificial Intelligence Organization.

Torrey, L. and Taylor, M. (2013). Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060.

Wan, R., Zhang, S., Shi, C., Luo, S., and Song, R. (2021). Pattern transfer learning for reinforcement learning in order dispatching. *arXiv preprint arXiv:2105.13218*.

Wang, L., Guo, B., and Yang, Q. (2018a). Smart city development with urban transfer learning. *Computer*, 51:32–41.

Wang, Z., Qin, Z., Tang, X., Ye, J., and Zhu, H. (2018b). Deep reinforcement learning with knowledge transfer for online rides order dispatching. pages 617–626.

Wen, J., Zhao, J., and Jaillet, P. (2017). Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 220–225. IEEE.

Zhu, C. (2020). Learning by reusing previous advice in teacher-student paradigm. In *Proceedings of the 2020 International Conference on Autonomous Agents and Multi-Agent Systems, Underline Science Inc.*

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.