Contextual Pre-Planning on Reward Machine Abstractions for Enhanced Transfer in Deep Reinforcement Learning

Guy Azran[1], Mohamad H. Danesh[2], Stefano V. Albrecht[3], Sarah Keren[1]

[1]Faculty of Computer Science, Technion - Israel institute of technology
[2]School of Computer Science, McGill University
[3]School of Informatics, University of Edinburgh

Abstract

Recent studies show that deep reinforcement learning (DRL) agents tend to overfit to the task on which they were trained and fail to adapt to minor environment changes. To expedite learning when transferring to unseen tasks, we propose a novel approach to representing the current task using *reward machines* (RMs), state machine abstractions that induce subtasks based on the current task's rewards and dynamics. Our method provides agents with symbolic representations of optimal transitions from their current abstract state and rewards them for achieving these transitions. These representations are shared across tasks, allowing agents to exploit knowledge of previously encountered symbols and transitions, thus enhancing transfer. Empirical results show that our representations improve sample efficiency and few-shot transfer in a variety of domains.

1 Introduction

Reinforcement learning (RL) methods, especially Deep RL (DRL) methods, have shown impressive capabilities in a wide variety of problems (Chen, Xu, and Agrawal 2021; Schrittwieser et al. 2020). However, recent studies show that these algorithms have difficulty adapting to even the slightest variations in the agent's objective or environment dynamics (Danesh and Fern 2022; Agarwal et al. 2021a; Zhang et al. 2018; Leike et al. 2017). Adapting quickly to new tasks is imperative in real-world scenarios, such as robotics (Ngo et al. 2018) and healthcare (Tseng et al. 2017), where agents reside in a dynamic world with ever-changing objectives and constraints. Consequently, agents require many interactions with the environment to learn to perform new tasks despite having mastered similar ones. The problem is exacerbated for tasks with sparse reward signals (Gupta et al. 2022) and long-term dependencies between actions (Langford 2018).

Example 1 *A housekeeper robot learns to do multiple tasks, one of which is to make coffee in a mug. Next, the robot is tasked with making coffee in a glass, something it has never attempted. The two tasks are similar in that they interact with many of the same objects (e.g. coffee, spoon, etc.) and perform identical subtasks (e.g. boil water, fill cup, etc.). The robot is expected to use its experience in making coffee in a mug to learn to achieve the new task more quickly.*

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

A *Contextual MDP* (CMDP) (Langford 2017; Hallak, Di Castro, and Mannor 2015) models settings such as Example 1 as a collection of tasks within the same environment, where each task is represented by the current *context*. CMDPs have been used in recent work that aims to improve *zero-shot* transfer capabilities, i.e., solving new tasks after training on a subset of them (Benjamins et al. 2022; Hallak, Di Castro, and Mannor 2015). In contrast, we aim to improve *few-shot* transfer, in which the agent may continue training on previously unseen tasks with the objective of minimizing the additional training required to achieve desirable performance.

One of the key challenges when using a CMDP to model transfer learning settings is finding a concise way to represent the current context while maximizing transfer capabilities. For this, we take advantage of *Reward Machines* (RMs) (Toro Icarte et al. 2018), state machine based abstractions that represent the structure of the reward function and the dynamics of a task and its subtasks. Transitions between abstract states in the RM occur when certain facts, represented as binary symbols, hold true. As the agent traverses the environment, it keeps track of these facts and its current RM state. Camacho et al. (2021) used RMs by providing the agent with the current abstract state and showed that this can expedite learning on a single task. In contrast, we leverage RMs to improve transfer.

Our novel technique, called *Contextual PRE-Planning* (C-PREP), takes as input a CMDP and an RM generator function that represents contextual information through task-specific RM abstractions with shared symbolic representations. Given a task, C-PREP finds an optimal policy in the corresponding RM abstraction and gives the agent the next desired abstract transition according to that policy as additional input. Furthermore, C-PREP uses the RM by reshaping the reward function according to abstract state transitions within the RM, thus highlighting important transitions throughout learning. When transferred to a new task, the agent can exploit abstract transitions that it has encountered during training and needs only to adapt to symbols with which it has not previously interacted.

We empirically evaluate C-PREP in various environments with sparse rewards and varying difficulties. In our experiments, a DQN agent (Mnih et al. 2015) is initially trained on a collection of source contexts. Subsequently, we transfer

the policy network to a different set of target contexts, where it undergoes further training and continuous evaluation. We observe an improvement in few-shot as well as zero-shot transfer performance when using C-PREP compared to various other context representation methods. The performance gap grows as the problem difficulty increases, with an improvement of 22.84% to 42.31% in time-to-threshold (few-shot transfer), and from 11.86% to 36.5% in jumpstart (zero-shot transfer) for the most complex tasks compared to the next best baseline.

2 Background

Reinforcement Learning (RL) is a method for agent learning through experiencing the world, acting within it, and receiving rewards (both positive and negative) for achieving certain states or state transitions. RL problems commonly model the world as a **Markov Decision Process** (MDP) (Bellman 1957) $M = \langle S, A, T, R, \gamma \rangle$ where S is a set of possible states, A is a set of agent actions, $T : S \times A \times S \rightarrow [0, 1]$ is the state transition function, $R : S \times A \times S \rightarrow \mathbb{R}$ is the transition reward function, and γ is the temporal reward discount factor. The objective is to find a policy π^* such that: $\pi^* \in \arg\max_\pi \mathbb{E}[J(\pi)]$, where $J(\pi) = \mathbb{E}_{s_t, s_{t+1} \sim T; a_t \sim \pi} [\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$ is the expected return of policy π in MDP M.

In this work we focus on **Transfer learning** (TL), which is the improvement of learning a new task through the transfer of knowledge from a related task that has already been learned (Torrey and Shavlik 2009). We model a collection of MDPs using a **Contextual MDP** (CMDP) (Hallak, Di Castro, and Mannor 2015), a 4-tuple $\langle C, S, A, \mathcal{M} \rangle$ where C is the context space, S and A are state and action spaces, and $\mathcal{M}$ is a mapping from a context $c \in C$ to an MDP $\mathcal{M}_c$ consisting of S and A but with distinct transition and reward functions, that is, $\mathcal{M}_c = \langle S, A, T_c, R_c, \gamma \rangle$. We sometimes refer to context-induced MDPs as "tasks", and to the shared S and A as the "environment". In Example 1, C is the set of all house chores, S and A are the state of the house and the agent's capabilities, and $\mathcal{M}$ maps a chore c to an MDP $\mathcal{M}_c$ that corresponds to completing the chore.

Fig. 1a depicts the general flow of transfer learning over a CMDP (Benjamins et al. 2022; Hallak, Di Castro, and Mannor 2015). The input consists of the observed MDP state and the current context. The state representation is processed by a (optional) feature extractor to be represented as a vector. The context is represented via a *context representation function* that maps a context to a vector representation. The state representation is merged with the context representation (usually by concatenation), and the new representation is fed into a policy network that will determine the next action.

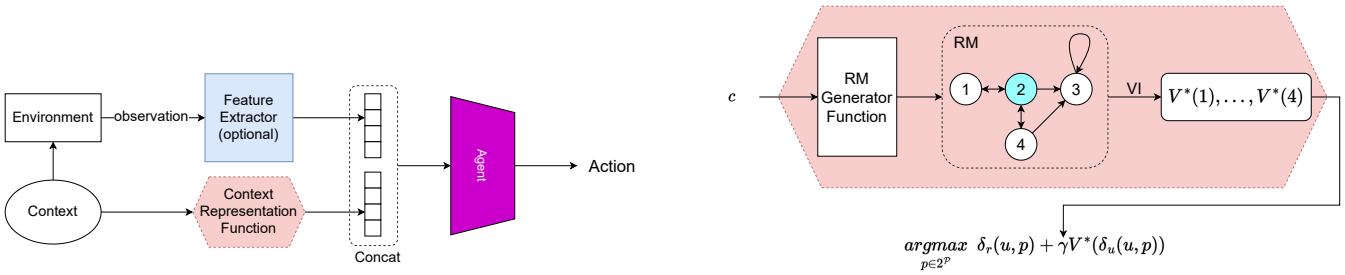
Kirk et al. (2021) distinguish between two categories of context representations. The first type, known as *Controllable* (CTL) context representations, includes the necessary information to generate the MDP, which can be thought of as a transparent implementation of the environment generation process (implemented in $\mathcal{M}$). The second type, *Procedural Content Generation* (PCG) context representations, conceal the MDP variables and only reveal information about the context identity, operating as a black-box with no insight into the generation process.

Given a CMDP $\langle C, S, A, \mathcal{M} \rangle$, transfer learning algorithms attempt to leverage knowledge from interactions with a set of *source contexts* $C_{\text{src}} \subset C$ to improve learning in a set of *target contexts* $C_{\text{tgt}} \subset C$ such that $C_{\text{src}} \cap C_{\text{tgt}} = \emptyset$. In Example 1, C_{src} is the set of contexts representing the chores it learns to do, including making coffee in a mug, and making coffee in a glass is a context in C_{tgt}. Policies learned after training in C_{src} and C_{tgt} from scratch are the *source policy* and the *target policy*, respectively. The policy learned on C_{tgt} after training in C_{src} is the *transferred policy*. Given a distribution Ψ over C, the objective is to optimize a chosen *transfer utility* $\mathcal{U}$ in expectation over sampled source and target context sets. Transfer utilities of interest in this work, suggested by Taylor and Stone (2009), are *jumpstart* (JS), *time to threshold* (TT), and *transfer ratio* (TR). JS measures (zero-shot transfer) performance on the target contexts without additional training. TT measures the number of training timesteps taken until convergence to a policy of acceptable performance threshold (few-shot transfer). TR measures the ratio of rewards accumulated over time by the agent using knowledge transfer against the agent that is trained from scratch, that is, how much the agent benefits from transfer (transfer relevance). Calculations of these utilities are available in Appendix F.

Reward Machines (RMs) (Toro Icarte et al. 2022) are state machine abstractions of MDPs. Given a set of propositional symbols $\mathcal{P}$, an RM is a 3-tuple $\mathfrak{R} = \langle U, \delta_u, \delta_r \rangle$ where U is a set of abstract states, and $\delta_u : U \times 2^{\mathcal{P}} \rightarrow U$ and $\delta_r : U \times 2^{\mathcal{P}} \rightarrow \mathbb{R}$ are the abstract transition and reward functions, respectively. Given the current abstract state $u \in U$ and a subset of propositional symbols $l \subseteq \mathcal{P}$ that hold true, $\delta_u(u, l)$ is the next abstract state and $\delta_r(u, l)$ is the reward received for this transition. When $\delta_u(u, l) = u'$, l is called the abstract *transition label* from u to u'. To connect between the abstraction and the underlying MDP, $\mathcal{P}$ is coupled with a *transition labeling function* $L : S \times A \times S \rightarrow 2^{\mathcal{P}}$ that maps state-action-state transitions in the MDP to abstract transition labels in the RM.

Fig. 2a textually describes an RM for the task of making coffee in Example 1. It defines abstract states u_0 to u_3 that each represent a high-level stage within the task of making a cup of coffee. The RM dictates that the agent must first boil some water, then put the coffee in the cup, and finally pour boiling water into the cup. These relationships are graphically visualized in Fig. 2b.

The main benefits of RMs are that they represent transitions between abstract states using binary symbols that pertain to the state of the underlying MDP (through L) and provide dense rewards via reward shaping. As a result, an RM corresponding to some context divides its induced task into sub-tasks that each describe a stage in the process of solving the overall task, rewarding the agent upon completion of each sub-task. To employ sensible reward shaping, we use *potential-based reward shaping* (Ng, Harada, and Russell 1999) which, given a potential function ϕ, define a new abstract reward function $\delta'_r(u, l) = \delta_r(u, l) + \gamma\phi(\delta_u(u, l)) - \phi(u)$. Toro Icarte et al. (2022) prove that potential-based re-



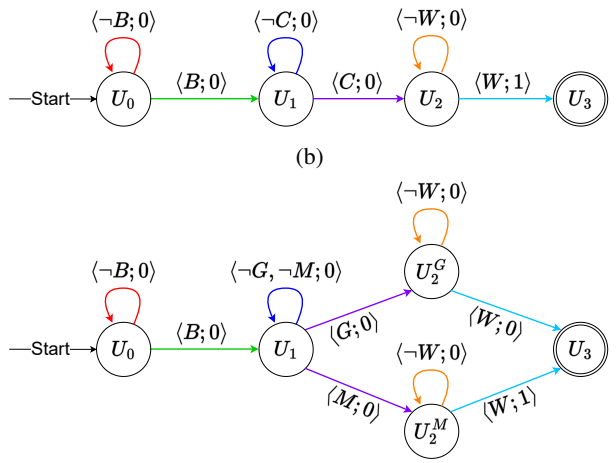
(a) (b)

Figure 1: (a) The general flow of transfer learning with a CMDP. (b) A visualization of the C-PREP context representation function. Context c is used to generate a task-specific RM.

```
Symbols:
    B - There is boiling water in the kettle.
    C - Coffee contents are in the cup.
    W - Boiling water poured into the cup.

Make-Coffee:
States - u0,u1,u2,u3
Transitions -
    (u0, not B) --> next=u0;r=0
    (u0,     B) --> next=u1;r=0
    (u1, not C) --> next=u1;r=0
    (u1,     C) --> next=u2;r=0
    (u2, not W) --> next=u2;r=0
    (u2,     W) --> next=u3;r=1
```

(a)



(b) (c)

Figure 2: (a) A textual representation of the RM in Example 1 describing the `Make-Coffee` task. (b) A graph visualization of the textually defined RM. (c) An expansion of the RM that differentiates between mug and glass receptacles, described in Section 3.

ward shaping guarantees that optimal policies in an MDP for which rewards have been replaced with RM rewards are optimal using the RM reshaped rewards. Moreover, it is empirically shown that using RM-reshaped rewards can significantly expedite policy convergence for RL agents.

3 Contextual Pre-Planning (C-PREP) for Transfer Learning

We aim to improve transfer in multi-task domains modeled as Contextual MDPs (CMDPs). Benjamins et al. (2022) proved that to guarantee optimality, it is necessary to condition the policy on the context itself. Therefore, it is crucial to represent the context in a way that allows the agent to generalize across contexts. For this, we use RMs to represent the current context and offer a novel way to enhance the agent's ability to exploit its previous experiences in new settings. Since our focus is on exploiting the structure of the RMs for transfer and not on their generation, we assume that the RM generator function is given as input, which can be based on domain knowledge, learned from demonstration (Camacho et al. 2021), or learned via discrete optimization (Toro Icarte et al. 2019).

Camacho et al. (2021) exploited RMs to expedite learning in single task domains by providing the agent with the current abstract state. We instead focus on transfer learning, and provide the next desired abstract transition from the current RM abstract state as contextual input at each timestep. Essentially, we guide the agent through optimal paths in the RM with abstract transitions represented using a set of symbols that is shared across all tasks. Upon transfer, the agent can expedite transfer learning by exploiting abstract transitions and leveraging prior knowledge of encountered symbols in the new task. This may be beneficial for learning in general but is key in transfer settings as it provides reusable representations between tasks.

C-PREP Context Representation Function. Based on the above intuition, we propose ***Contextual PRE-Planning*** (C-PREP) for leveraging information in context-specific RMs. For each task, C-PREP generates an RM $\langle U, \delta_u, \delta_r \rangle$ with abstract transitions represented using a shared symbol set, that is, we use the same symbol set $\mathcal{P}$ to represent all RM transitions. Using Value Iteration (VI) (Bellman 1957),

we find an optimal policy in the RM. We then give the agent an optimal abstract transition label in the RM from the current abstract state u (as dictated by the RM policy), i.e., a transition label l such that $\delta_u(u, l)$ is the next state on a (discounted) reward-maximizing path in the RM. Intuitively, we wish to guide the agent towards an optimal path within the RM.

C-PREP relies on providing the next desired abstract transition in the RM to the agent. However, since there is no direct representation of actions in the RM, we cannot use standard planning methods for this purpose. We therefore use a variant of value iteration, as suggested by Toro Icarte et al. (2022), with the following update rule over the abstract states of RM $\mathfrak{R}$.

$$V^k_{\mathfrak{R}}(u) = \max_{l \in 2^{\mathcal{P}}} \left[\delta_r(u, l) + \gamma V^{k-1}_{\mathfrak{R}}(\delta_u(u, l))\right] \quad (1)$$

where $V^k_{\mathfrak{R}}$ is the value of abstract state u at iteration k ($V^0_{\mathfrak{R}} = 0$), and $\delta_u(u, l)$ and $\delta_r(u, l)$ are the next abstract state and reward received for achieving transition label l at abstract state u, respectively. To show the relationship between this rule and VI for MDPs, we define $M_{\mathfrak{R}} = \langle U, 2^{\mathcal{P}}, T, R, \gamma \rangle$ where $T(u, l, \delta_u(u, l)) = 1$ and $R(u, l, u') = \delta_r(u, l)$. We observe that the VI update rule for $M_{\mathfrak{R}}$, denoted V^k, is equivalent to $V^k_{\mathfrak{R}}$. Formally,

$$\begin{aligned} V^k(u) &= \max_{l \in 2^{\mathcal{P}}} \sum_{u' \in U} T(u, l, u')(R(u, l, u') + \gamma V^{k-1}(u')) \\ &= \max_{l \in 2^{\mathcal{P}}} R(u, l, \delta_u(u, l)) + \gamma V^{k-1}(\delta_u(u, l)) \\ &= \max_{l \in 2^{\mathcal{P}}} \delta_r(u, l) + \gamma V^{k-1}(\delta_u(u, l))) = V^k_{\mathfrak{R}}(u) \end{aligned}$$

Thus, to identify optimal abstract transitions, we can find an abstract optimal policy in RM $\mathfrak{R}$ by using VI to find an optimal policy π^* in $M_{\mathfrak{R}}$.

Given the current abstract state u, from which there may be multiple optimal abstract transitions, C-PREP samples an optimal abstract transition l from $\pi^*(\cdot|u)$. Since π^* is optimal in deterministic MDP $M_{\mathfrak{R}}$:

$$supp(\pi^*(\cdot|u)) \subset \arg\max_{l \in 2^{\mathcal{P}}} (\delta_r(u, l) + \gamma V^*(\delta_u(u, l)))$$

where $supp(\pi^*(\cdot|u))$ is the support set of probability distribution $\pi^*(\cdot|u)$. Thus, any transition we sample from π^* is one that maximizes discounted return in the RM.

Based on the above formulations, the C-PREP context representation function (depicted in Fig. 1b) operates in a three-step process: (1) generate an RM $\mathfrak{R} = G(\mathcal{M}_c)$ for the current context c, (2) find an optimal policy π^* in $M_{\mathfrak{R}}$, (3) at each timestep, sample an optimal transition $l \sim \pi^*(\cdot, u)$ given the current RM abstract state u and return it.

Throughout training, the C-PREP RM generation function updates its returned representation according to the current abstract state. To notify the agent that a correct (or incorrect) abstract transition has been completed, we provide additional rewards that emphasize the executed abstract transition's quality. For this, we employ potential based reward-shaping as defined in Section 2. As it is already calculated, we use V^* as the potential function ϕ to generate the reward signal that is provided to the agent instead of the original MDP reward. In the RM described in Fig. 2a, the agent will receive a higher reward for transitioning from state u_0 to u_1 rather than loop back to itself because this brings it closer to the abstract goal state.

Transfer Learning with C-PREP The input to our setting includes a CMDP $\langle C, S, A, \mathcal{M} \rangle$ and an *RM generator function* G that maps each context-induced task $\mathcal{M}_c$ to its corresponding RM $G(\mathcal{M}_c) = \langle U^c, \delta^c_u, \delta^c_r \rangle$ which is defined over shared symbol set $\mathcal{P}$.

The C-PREP context representation function can be integrated into any algorithm following the transfer learning flow depicted in Fig. 1a. Algorithm 1 (Appendix G) demonstrates an implementation of a DQN (Mnih et al. 2015) for transfer learning settings using C-PREP as the context representation function and RM reward shaping. The key differences between this implementation and the standard DQN are that the algorithm initially generates an RM for the sampled context, calculates its state values, and reshapes the RM rewards. States encountered in the episode are augmented by the C-PREP context representation according to the RM transition. Rewards are replaced with the reshaped rewards from the RM according to the achieved abstract transition at that timestep.

We note that the ability of C-PREP to support transfer depends on the *resolution* of the generated RMs, i.e., how well the generated RMs represent the context space. If the set of propositional symbols $\mathcal{P}$ is too abstract, the generated RMs do not sufficiently distinguish between contexts. In contrast, if it is too refined, computation time may increase due to running VI in huge tables for every context.

In Example 1, when training to make coffee in a mug, the agent learns to pour water into the mug and should exploit this capability upon transferring to the task of making coffee in a glass. Fig. 2 shows two different RMs that can be used to describe this setting. The RM in Fig. 2b does not differentiate between a mug and a glass, as they are both encapsulated by the "cup" symbol $\mathtt{C}$. In contrast, the RM in Fig. 2c distinguishes between the tasks of making coffee in a mug and in a glass, rewarding the agent only for the former (when transitioning from u^M_2 to u_3).

4 Empirical Evaluation

The objective of our empirical evaluation is to examine whether agents using C-PREP exhibit improved performance on transfer utilities of interest.

4.1 Experimental Setup

Environments: We test our method in four environments with compound and long-horizon tasks and sparse reward signals[1]:

Grid Navigation (GN): An agent must reach a specified destination on a grid. The state space consists of the agent's current location and the action space includes moving in one of the four cardinal directions and a "done" action to be called upon arrival at the destination.

[1]Our code base is described in Appendix I.

Multi Points-of-interest (MP): The agent navigates to multiple destinations *in any order*. The state space consists of the agent's location and an indicator of whether a certain destination has already been visited. The action space is as in GN, but with an "arrived" action replacing the "done" action.

Pick-up and drop-off (PD): An agent picks up and drops off passengers at their destinations. The state space is as in MP with indicators for passengers that have been dropped off at their destinations. In addition to navigation actions, the action space contains "pick-up" and "drop-off" actions.

Ordered Navigation (ON): The agent must navigate to specified destinations *in a specific order*. The state and action spaces are as in MP.

All maps are 6×6. At every timestep, the agent receives a reward of 1 when achieving the environment objective and 0 otherwise, and the discount factor is $\gamma = 0.99$.

Defining the CMDP Spaces: The environments described above include pairs of state and action spaces. To define a CMDP we couple them with the following context spaces:

- **Entity Location (EL):** The context indicates the locations of core entities in the environment, e.g., passenger locations and drop-off destinations.
- **Changing Map (CM):** The context indicates the number and location of walls in the grid.
- **Point-of-interest Order (PO):** The context indicates the order of the locations to visit.

The GN, MP, and PD environments are each used with both the EL and CM context spaces. The ON environment is paired with the PO context space. Contexts are represented using *controllable* (CTL) representations (see Section 2). For full details on CTL context representations, see Appendix C.

Transfer Session: Each transfer learning training session begins by randomly sampling two disjoint context sets from the CMDPs described above; the source set C_{src} and the target set C_{tgt}. We adopt "training protocol **B**" of Kirk et al. (2021) such that the size of C_{src} is much smaller than the size of the context space. The agent initially trains on tasks induced by C_{src} for N_{src} steps and then continues its training in C_{tgt} for additional N_{tgt} steps. We record performance progress during and after training. For full details see Appendix E.

Context Representations: We vary the RM information exposed to the agent. We examine the following representations:

- CTL- Controllable context representation without RMs (same baseline in (Toro Icarte et al. 2018)).
- CTL+RS (Toro Icarte et al. 2018) - Adds dense reshaped RM rewards to the current context's reward functions.
- CTL+LTL+RS (Camacho et al. 2021) - Adds the *Last Transition Label* (LTL) as an additional context representation that is the current truth assignment of all propositional labels.
- CTL+C-PREP (**ours**) - Adds the C-PREP context representation, i.e., the *Desired Transition Label* (DTL), with RS.
- C-PREP (**ours**) - The C-PREP context representation without a CTL context representation

In Appendix A we show additional experiments using PCG context representations in lieu of CTL.

Reported Metrics: During each training session, we evaluate the source, target, and transferred policies on the context set on which it is trained at 100 uniformly spaced evaluation points. At each evaluation point, we record the policy's average return on 50 sampled contexts. Each training session is repeated 5 times, using different random seeds. From the computed average returns, we calculated the transfer utilities defined in Section 2: JS, TT, and TR (see Appendix F for the formula used to compute these measures). We aggregate the results using interquartile mean (IQM) and calculated the standard deviation and stratified bootstrap 95% confidence intervals (Agarwal et al. 2021b). To report results for different performance thresholds, we plot the TT as a function of the threshold. We measure the IQM area under the curve (AUC) of this function, denoted TT_{AUC}.

4.2 Results

First, to examine the performance over the entire transfer session, Table 1 shows the interquartile mean (IQM) and standard deviation of the measured transfer utilities (TT_{AUC}, JS, TR) for all tested configurations using a CTL context representation. The best results for each CMDP (row) are marked in bold. Negative TR values that indicate non-beneficial transfer are colored red.

Our method performed best in terms of TT_{AUC} and JS in all but two CMDPs: (1) in GN (shortest horizon) with both context spaces (EL and CM), CTL+LTL+RS performs best in terms of TT_{AUC}; (2) in GN with EL context space, using CTL alone performs best in terms of JS. Notably, in PD, which is the longest horizon task, our method outperforms all other configurations. Compared to the highest performing baseline, we see TT_{AUC} improvements of 22.84% in the context space CM and 42.31% in the context space EL, and JS improvements of 11.86% in CM and 36.5% in EL. TR results show low and negative TR values for most configurations. Our method is the only one with positive TR throughout all tasks. In the PO environment (longest horizon), we see a performance improvement of over 300% when using C-PREP compared to the next best configuration.

Next, we examine the achieved threshold performance throughout training. Fig. 3 visualizes the IQM TT results, measured in training progress (percentage) as a function of the threshold, i.e., the curve from which we derive TT_{AUC}. The shaded areas are stratified bootstrap 95% confidence intervals. Each row corresponds to a context space. Each column corresponds to an environment. Agents using RS in environments tested with the CM and EL context show similar performance in GN, but a performance gap (in favor of our method) widens as the task horizon grows. In the PO environment, our method is the only one that can be seen converging to a high performing policy.

Table 1: IQM and standard deviation transfer utilities of configurations with CTL.

Utility	Context Space	Environment	CTL	CTL+RS	CTL+LTL+RS	CTL+C-PREP (ours)	C-PREP (ours)
TT_{AUC}	EL	GN	25.41 ± 7.27	6.37 ± 0.71	**6.21 ± 0.42**	6.42 ± 0.44	42.15 ± 3.95
		MP	94.77 ± 6.02	44.78 ± 12.06	19.88 ± 9.36	**18.32 ± 9.27**	86.43 ± 1.89
		PD	97.39 ± 1.57	95.18 ± 3.98	37.58 ± 23.38	**21.68 ± 6.55**	88.35 ± 1.59
	CM	GN	42.18 ± 1.58	16.30 ± 2.35	**7.14 ± 1.41**	7.64 ± 1.72	32.98 ± 0.68
		MP	71.90 ± 10.62	26.48 ± 15.51	14.24 ± 8.85	**13.64 ± 8.60**	47.74 ± 11.46
		PD	86.30 ± 18.81	54.19 ± 21.64	37.52 ± 25.46	**28.95 ± 24.15**	51.02 ± 15.46
	PO	ON	98.04 ± 0.00	95.15 ± 7.66	97.50 ± 5.59	32.93 ± 10.21	**22.54 ± 1.57**
JS	EL	GN	**0.28 ± 0.11**	0.06 ± 0.06	0.07 ± 0.09	0.05 ± 0.05	0.01 ± 0.05
		MP	0.03 ± 0.06	0.26 ± 0.15	0.48 ± 0.26	**0.49 ± 0.17**	0.01 ± 0.02
		PD	0.02 ± 0.02	0.03 ± 0.01	0.34 ± 0.20	**0.38 ± 0.12**	0.00 ± 0.01
	CM	GN	0.42 ± 0.03	0.55 ± 0.06	0.73 ± 0.05	**0.75 ± 0.08**	0.49 ± 0.06
		MP	0.23 ± 0.09	0.49 ± 0.14	0.66 ± 0.11	**0.68 ± 0.10**	0.42 ± 0.10
		PD	0.08 ± 0.19	0.28 ± 0.19	0.38 ± 0.27	**0.52 ± 0.25**	0.31 ± 0.19
	PO	ON	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.14 ± 0.26	**0.75 ± 0.02**
TR	EL	GN	-0.11 ± 0.10	**0.13 ± 0.04**	0.08 ± 0.03	0.07 ± 0.03	0.04 ± 0.02
		MP	-0.99 ± 0.01	**0.24 ± 0.08**	0.24 ± 0.18	0.16 ± 0.12	0.06 ± 0.11
		PD	-1.00 ± 0.00	-0.99 ± 0.21	-0.14 ± 0.38	**0.14 ± 0.09**	-0.05 ± 0.12
	CM	GN	-0.11 ± 0.04	0.08 ± 0.04	**0.11 ± 0.01**	**0.11 ± 0.01**	0.08 ± 0.04
		MP	-0.85 ± 0.32	**0.29 ± 0.13**	0.26 ± 0.06	0.21 ± 0.06	0.07 ± 0.05
		PD	-0.94 ± 0.25	-0.06 ± 0.29	-0.05 ± 0.20	0.06 ± 0.19	**0.07 ± 0.07**
	PO	ON	0.00 ± 0.00	**2.04 ± 6.94**[2]	-0.87 ± 0.90	0.69 ± 0.20	0.31 ± 0.03

Appendix A shows results of experiments wherein we replace the CTL representations with uninformative PCG representations. TT and JS results are similar to those presented with CTL representations. TR results show that for all configurations, it is non-beneficial to use PCG representations for transfer due to severe overfitting.

Appendix B presents results for an ablation study that shows that all components of C-PREP are required to achieve the best results. Appendix H shows results for additional experiments on sample efficiency (in terms of number of contexts) and generalization capabilities of C-PREP using PCG.

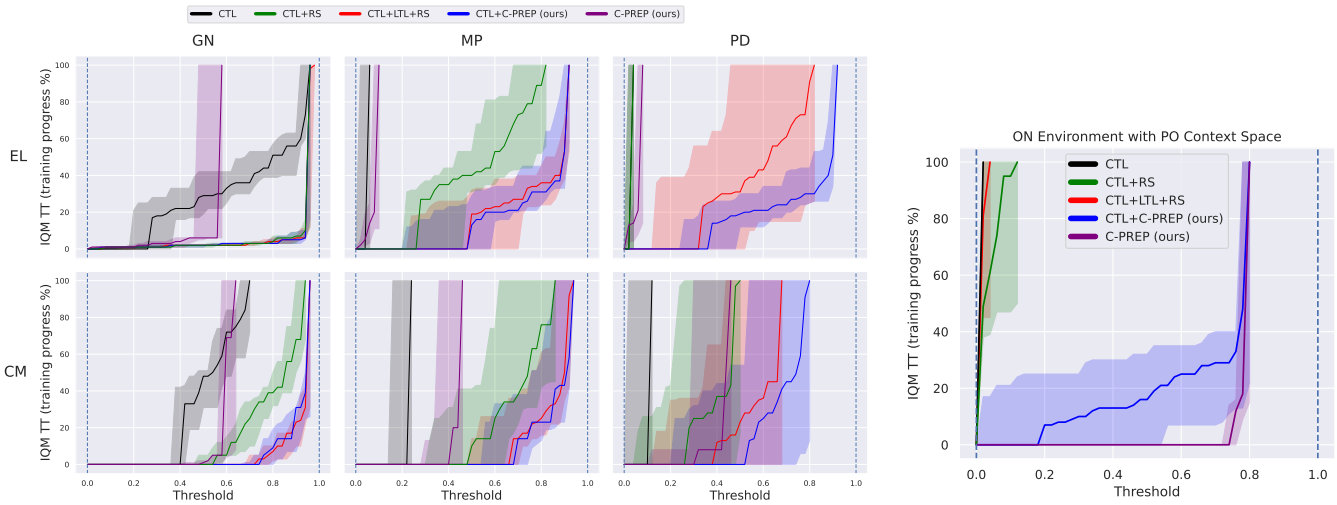
4.3 Discussion

Results demonstrate that C-PREP improves transfer performance in more complex tasks without hindering performance on simpler tasks. As visualized in Fig. 3a, all methods perform similarly in the GN environment (short horizon), but C-PREP opens a performance gap in TT that increases with the difficulty of the environment. The TR results show that only our method is beneficial for transfer in all tasks, as is evidenced by the negative TR values reported for all other configurations. In the PO environment, only agents using C-PREP are able to achieve threshold greater than 0.2. Furthermore, since the RM in this case differentiates all tasks, it is preferable to use C-PREP without CTL. We observe that the JS performance is approximately 93% of the maximum achieved performance threshold, which is reached in less than 20% of the training progress.

We examine the performance of C-PREP using partial resolution RMs, i.e., some tasks may be represented with the same RM. For this we remove CTL and use C-PREP alone. In the GN, MP, and PD environments, the agent will achieve a threshold performance of no more than 50% of C-PREP's performance *with* CTL. Fig. 3a shows that C-PREP without CTL achieves medium to low performance depending on the environment and context space. We attribute this to the low coverage of tasks with the partial RM resolution. These RMs (describe in detail in Appendix D) cover approximately 25% of the tasks in GN and approximately 6% of the MP and PD tasks. We conclude from this that C-PREP with partial resolution RMs cannot compensate for missing contextual information. However, Fig. 3b shows that in the ON environment, where the RMs are of full resolution, it is preferable to use C-PREP without additional context. We hypothesize that this is due to the large overlap in contextual data between the C-PREP context representation and CTL, making the information in CTL irrelevant. C-PREP's zero-shot results in this setting compared to other baselines. This use of local context illustrate the advantage of solving the context as a series of smaller, simpler contexts.

We show additional results in experiments using PCG in place of CTL in Appendix A to examine the case of uninformative global context representations. Here, we notice that in 60% of the runs, it was more beneficial to train from scratch in C_{tgt} than to transfer from C_{src}. Fig. 4 in the appendix visualizes TT performance of PCG configurations and shows hindered performance compared to those in Fig. 3a that use CTL representations.

Ablation results (found in Appendix B) show the importance of every component of C-PREP. We notice that both TT_{AUC} and JS can be improved by up to 12% in five out



(a) GN, MP, and PD environments with EL and CM context spaces

(b) ON task with PO context space

Figure 3: The IQM TT of configurations using CTL as a function of the threshold.

of seven tasks by adding the LTL modification to CTL+C-PREP in the most complex task. We hypothesize that this will yield an even greater benefit in scenarios where it is harder to infer the current abstraction label.

Additional results in Appendix H reveal two interesting capabilities of C-PREP. First, C-PREP is more sample efficient in terms of the number of source contexts it is trained on, that is, C-PREP needs to train on less source contexts to achieve similar or better transfer performance than other tested configurations. Second, adding RM information when using PCG significantly improves generalization capabilities in the beginning of training. We see a spike in performance in the first 1M training steps, hinting at the potential of using RMs for learning generalized state representations.

5 Related Work

DRL agents are extremely susceptible to overfitting to the context in which they were trained. Leike et al. (2017) show that small changes to a single detail or obstacle could result in performance degradation. Danesh et al. (2021) demonstrate that simple RL agents overfit to the training settings such that they completely ignore observations. One solution is to train the agent on a distribution of contexts, rather than a single one (Zhang et al. 2018). However, once the context distribution departs from the training distribution, the performance drops despite the knowledge obtained during training (Agarwal et al. 2021a). We focus on transferring knowledge to expedite training in novel tasks.

There are several approaches to improve transfer learning in DRL. Some meta-learning methods (Eghbal-zadeh, Henkel, and Widmer 2021; Papoudakis, Christianos, and Albrecht 2021; Zintgraf et al. 2020; Wang et al. 2017; Duan et al. 2016) attempt to learn to estimate a context representation based on accumulated experiences while exploring. The model-agnostic approach (Finn, Abbeel, and Levine 2017) directly optimizes its parameters to minimize the number of gradient steps required to adapt the parameters to the current context. Model-based methods (Shrestha et al. 2020; Tamar et al. 2017) attempt to learn an approximate model of the world, use classical planning on them, and utilize the plan either explicitly or implicitly through another learning component. In this case, different contexts will induce different plans within the model. Techniques for improving exploration (Dorfman, Shenfeld, and Tamar 2021; Zintgraf et al. 2020) use Bayesian-adaptive RL to learn how to best explore the environment based on past episodes. All of the above rely on additional exploration to determine the context before learning to solve it. In contrast, we use contextual information to understand the task a priori to reduce exploration. Using CMDPs, we view the context as additional input to the agent (Langford 2017; Hallak, Di Castro, and Mannor 2015).

To improve few-shot transfer, our method represents different contexts as RMs (Toro Icarte et al. 2022). Previous work shows that RMs can be used to expedite learning of a single context (Toro Icarte et al. 2022; Camacho et al. 2021). We utilize RMs to represent contextual information, resulting in better sample efficiency and few-shot transfer learning capabilities in multi-context settings.

6 Conclusion

We presented ***Contextual PRE-Planning*** (C-PREP) as a novel context representation function and showed how it can enhance zero-shot and few-shot transfer capabilities of DRL agents. C-PREP exploits RMs by planning on them and providing the agent with a representation of the next desired transition. Our empirical evaluation demonstrates C-PREP's ability to improve sample efficiency and different transfer utilities, with an increased benefit for tasks of increasing difficulty.

To focus on the effect our RM-based representation has on transfer, we assumed the RM generation function is given as

input. Future work will include a theoretical analysis of the conditions under which a context representation is guaranteed to enhance transfer and the development of methods for learning the appropriate RMs through experience. As a second extension, we intend to examine alternative symbolic representations beyond RMs for enhancing learning and transfer, as well as consider the effect our suggested context representation function has in setting in which *options* (Sutton, Precup, and Singh 1999; Illanes et al. 2020) are used to distinguish between sub-tasks. Finally, we plan to examine our representation in real-world settings in which transfer may be beneficial.

References

Agarwal, R.; Machado, M. C.; Castro, P. S.; and Bellemare, M. G. 2021a. Contrastive Behavioral Similarity Embeddings for Generalization in Reinforcement Learning.

Agarwal, R.; Schwarzer, M.; Castro, P. S.; Courville, A. C.; and Bellemare, M. 2021b. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34, 29304–29320. Curran Associates, Inc.

Bellman, R. 1957. A Markovian Decision Process. *Indiana University Mathematics Journal*, 6(4): 679–684.

Benjamins, C.; Eimer, T.; Schubert, F.; Mohan, A.; Biedenkapp, A.; Rosenhahn, B.; Hutter, F.; and Lindauer, M. 2022. Contextualize Me – The Case for Context in Reinforcement Learning.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym.

Camacho, A.; Varley, J.; Zeng, A.; Jain, D.; Iscen, A.; and Kalashnikov, D. 2021. Reward Machines for Vision-Based Robotic Manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 14284–14290. Xi'an, China: IEEE. ISBN 978-1-72819-077-8.

Chen, T.; Xu, J.; and Agrawal, P. 2021. A System for General In-Hand Object Re-Orientation. In *Conference on Robot Learning*.

Danesh, M. H.; and Fern, A. 2022. Out-of-Distribution Dynamics Detection: RL-Relevant Benchmarks and Results. ArXiv:2107.04982 [cs].

Danesh, M. H.; Koul, A.; Fern, A.; and Khorram, S. 2021. Re-understanding Finite-State Representations of Recurrent Policy Networks. In *Proceedings of the 38th International Conference on Machine Learning*, 2388–2397. PMLR.

Dorfman, R.; Shenfeld, I.; and Tamar, A. 2021. Offline Meta Reinforcement Learning – Identifiability Challenges and Effective Data Collection Strategies. In *Advances in Neural Information Processing Systems*, volume 34, 4607–4618. Curran Associates, Inc.

Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P. L.; Sutskever, I.; and Abbeel, P. 2016. RL\^2\: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv:1611.02779 [cs, stat]*.

Eghbal-zadeh, H.; Henkel, F.; and Widmer, G. 2021. Context-Adaptive Reinforcement Learning using Unsupervised Learning of Context Variables. In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning*, 236–254. PMLR.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]*.

Gupta, A.; Pacchiano, A.; Zhai, Y.; Kakade, S. M.; and Levine, S. 2022. Unpacking Reward Shaping: Understanding the Benefits of Reward Engineering on Sample Complexity.

Hallak, A.; Di Castro, D.; and Mannor, S. 2015. Contextual Markov Decision Processes. *arXiv:1502.02259 [cs, stat]*.

Illanes, L.; Yan, X.; Icarte, R. T.; and McIlraith, S. A. 2020. Symbolic Plans as High-Level Instructions for Reinforcement Learning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30: 540–550.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kirk, R.; Zhang, A.; Grefenstette, E.; and Rocktäschel, T. 2021. A Survey of Generalisation in Deep Reinforcement Learning. *arXiv:2111.09794 [cs]*.

Langford, J. 2017. Contextual reinforcement learning. In *2017 IEEE International Conference on Big Data (Big Data)*, 3–3.

Langford, J. 2018. Real World Reinforcement Learning. Place: MLiTRW 2018, Criteo Paris.

Leike, J.; Martic, M.; Krakovna, V.; Ortega, P. A.; Everitt, T.; Lefrancq, A.; Orseau, L.; and Legg, S. 2017. AI Safety Gridworlds.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, 278–287. Morgan Kaufmann.

Ngo, P. D.; Wei, S.; Holubová, A.; Muzik, J.; and Godtliebsen, F. 2018. Reinforcement-learning optimal control for type-1 diabetes. In *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, 333–336.

Papoudakis, G.; Christianos, F.; and Albrecht, S. V. 2021. Agent Modelling under Partial Observability for Deep Reinforcement Learning.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; Lillicrap, T.; and Silver, D. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Shrestha, A.; Lee, S.; Tadepalli, P.; and Fern, A. 2020. DeepAveragers: Offline Reinforcement Learning by Solving Derived Non-Parametric MDPs.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; and Abbeel, P. 2017. Value Iteration Networks.

Taylor, M. E.; and Stone, P. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(56): 1633–1685.

Terry, J. K.; Black, B.; Grammel, N.; Jayakumar, M.; Hari, A.; Sullivan, R.; Santos, L.; Perez, R.; Horsch, C.; Dieffendahl, C.; Williams, N. L.; Lokesh, Y.; and Ravi, P. 2021. PettingZoo: Gym for Multi-Agent Reinforcement Learning.

Toro Icarte, R.; Klassen, T.; Valenzano, R.; and McIlraith, S. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2107–2116. PMLR.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2022. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *Journal of Artificial Intelligence Research*, 73: 173–208.

Toro Icarte, R.; Waldie, E.; Klassen, T.; Valenzano, R.; Castro, M.; and McIlraith, S. 2019. Learning Reward Machines for Partially Observable Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Torrey, L. A.; and Shavlik, J. 2009. Chapter 11 Transfer Learning. *undefined*.

Tseng, H.-H.; Luo, Y.; Cui, S.; Chien, J.-T.; Ten Haken, R. K.; and Naqa, I. E. 2017. Deep reinforcement learning for automated radiation adaptation in lung cancer. *Medical Physics*, 44(12): 6690–6705.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2017. Learning to reinforcement learn. *arXiv:1611.05763 [cs, stat]*.

Zhang, C.; Vinyals, O.; Munos, R.; and Bengio, S. 2018. A Study on Overfitting in Deep Reinforcement Learning.

Zintgraf, L.; Shiarlis, K.; Igl, M.; Schulze, S.; Gal, Y.; Hofmann, K.; and Whiteson, S. 2020. VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning.

A PCG Experiments

Table 2 shows that among configurations that use PCG based context representation, our CTL+C-PREP performed best in all CMDPs of the GN, MP, and PD environments in the TT_{AUC} utility. The near-zero JS values and small (¡0.1) to negative TR values indicate overfitting and nonbeneficial transfer in all CMDPs for all configurations except PCG+RS, which is able to show positive transfer in CM for MP and PD. In the PO environment, using C-PREP without the PCG context representation shows an improvement of at least 120% in all CM tasks.

B Ablation Tests

Table 3 shows the results of our ablation tests based on the C-PREP components. In the GN, MP, and PD environments, we used CTL as the base context representation. The results show that all components are necessary to achieve the highest transfer performance. As can be seen in Fig. 5a, configurations that do not use RS show no performance improvement during transfer. We further observe that adding LTL to our representation (CTL+LTL+C-PREP) outperforms CTL+C-PREP in the TT_{AUC} utility in five out of six of the CMDPs in these environments. In the ON environment, due to the success of C-PREP without CTL, we additionally tested ablations in this environment without CTL. From the ablation analysis in Fig. 5b we see that without all C-PREP components, the agent cannot complete the tasks after transfer, which shows the significance of each component. In this setting (ON) LTL+C-PREP performs on par with C-PREP, showing that our representation is sufficient.

C Context Representations

In our experiments, we use both CTL and PCG context representations. CTL representations contain all the information required to generate the MDP induced by the context. In practice, this is part of the observation that is constant throughout the task. PCG representations are random indices assigned to each context as an identifier. These indices are provided to the agent as one-hot-encoding vectors. Fig. 6 illustrates the implementation of the corresponding context representation functions.

The EL contexts are represented by the location of the entities (passengers and their destinations) in the environment. This is a pair of row-column coordinates for each entity. The CM contexts are represented as a binary vector where each value indicates the existence of a wall at a certain position in the environment map. The PO contexts are represented by a single index for each passenger that is the position of the passenger in the pickup order. Fig. 7a shows two different contexts in the PD environment coupled with the EL context space.

D Experiment Reward Machines

In the PO context space, there is an RM generator function of full resolution that generates simple RMs where the number of propositional symbols abstract states is equal to the number of passengers. The RM generator function template with variables `i1,..., i2` that defines the passenger pickup order is as follows:

```
Symbols:
    P1 - Passenger 1 has been picked up
    ...
    P5 - Passenger 5 has been picked up

Order-i1,i2,i3,i4,i5:
States - u0, ..., u5
Transitions -
    (u0, not Pi1) --> next=u0;r=0
    (u0,     Pi1) --> next=u1;r=0
    ...
    (u4, not Pi5) --> next=u4;r=0
    (u2,     Pi5) --> next=u5;r=1
```

For the EL context space, the entities may be anywhere on the map. This means that an RM generator function that differentiate between all contexts needs to have use at least one symbol for every possible position. Because the agent can potentially visit any of these positions during an episode, we must have an abstract state for each position in the map and passenger status (waiting, picked up, delivered). Such an RM is about the same size as the entire MDP, and so it would be more beneficial to run VI on the MDP itself and guarantee a near-optimal policy rather than the RM. Similarly, the CM contexts determine the locations of obstacles, and so the RM must account for every possible transition in the map (which is also about the size of the MDP).

To overcome this issue, we use an RM generator function that groups adjacent cells into sectors and use symbols that indicate the agent's presence in a specific sector. A transition between sectors is possible if a transition between cells of those sectors is possible. Fig. 7b illustrates this in the PD environment. This significantly reduces the size of the generated RMs, but the RM can only account for transitions between sectors (partial resolution).

E Training Technical Details

We trained each DQN agent on a single CPU core for four million steps on C_{src} and C_{tgt}. For the EL context space, we train on 100 source contexts and transfer to 200 target contexts. For the CM context space, we train 250 source contexts and transfer to 500 contexts. The GN environment contains only 36 possible entity position and so with the EL context space there are only 36 contexts. Thus, we train on 8 source contexts and transfer to 16 target contexts in this environment, leaving room for the sampled contexts to change between runs.

DQN hyperparameters were selected via grid search of potential candidates, followed by a manual search in areas of interest. To demonstrate C-PREP's resilience to hyperparameters, hyperparameters were chosen to optimize the CTL configuration and used globally across all other configurations. The Q-value estimator network is comprised of two hidden linear layers with Rectified Linear Unit (ReLU) activations, ending with a linear output layer of width equal to the number of agent actions. The parameters are optimized

Table 2: IQM and standard deviation transfer utilities in tested CMDPS using PCG as the base context representation. The best results for each CMDP (row) are marked in bold. Negative TR values that indicate non-beneficial transfer are colored red.

Utility	Context Space	Environment	PCG	PCG+RS	PCG+LTL+RS	PCG+C-PREP (ours)	C-PREP (ours)
TT_{AUC}	EL	GN	84.95 +- 3.23	7.12 +- 0.66	5.89 +- 0.22	**5.82 +- 0.28**	42.15 +- 3.95
		MP	98.04 +- 0.00	96.11 +- 1.14	95.02 +- 2.03	**61.96 +- 9.37**	86.43 +- 1.89
		PD	98.04 +- 0.00	98.04 +- 0.00	96.75 +- 0.90	**58.97 +- 6.69**	88.35 +- 1.59
	CM	GN	85.61 +- 12.21	20.01 +- 3.63	16.45 +- 2.37	**14.58 +- 2.19**	32.98 +- 0.68
		MP	98.04 +- 0.00	51.29 +- 12.97	42.41 +- 12.14	**32.73 +- 13.21**	47.74 +- 11.46
		PD	98.04 +- 0.00	55.94 +- 19.14	43.39 +- 16.48	**39.95 +- 18.23**	51.02 +- 15.46
	PO	ON	98.04 +- 0.00	92.56 +- 13.58	93.01 +- 3.80	37.65 +- 22.90	**22.54 +- 1.57**
JS	EL	GN	**0.01 +- 0.03**	0.00 +- 0.01	0.00 +- 0.01	0.00 +- 0.01	0.01 +- 0.05
		MP	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	**0.01 +- 0.02**
		PD	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	0.00 +- 0.01
	CM	GN	0.00 +- 0.05	0.00 +- 0.02	0.18 +- 0.14	0.10 +- 0.18	**0.49 +- 0.06**
		MP	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.02	0.00 +- 0.00	**0.42 +- 0.10**
		PD	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.13	**0.31 +- 0.19**
	PO	ON	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	**0.75 +- 0.02**
TR	EL	GN	-0.86 +- 0.05	0.05 +- 0.01	**0.05 +- 0.02**	0.04 +- 0.02	0.04 +- 0.02
		MP	-1.00 +- 0.00	-0.85 +- 0.03	-0.93 +- 0.13	-0.31 +- 0.21	**0.06 +- 0.11**
		PD	-1.00 +- 0.00	-0.84 +- 0.09	-0.94 +- 0.07	-0.33 +- 0.16	**-0.05 +- 0.12**
	CM	GN	-0.77 +- 0.37	-0.02 +- 0.06	-0.02 +- 0.04	0.00 +- 0.02	**0.08 +- 0.04**
		MP	-1.00 +- 0.00	**0.11 +- 0.18**	-0.03 +- 0.08	-0.02 +- 0.03	0.07 +- 0.05
		PD	-0.60 +- 0.49	**0.21 +- 0.21**	0.06 +- 0.19	0.06 +- 0.06	0.07 +- 0.07
	PO	ON	0.00 +- 0.00	inf[3]	-0.71 +- 1.79	0.31 +- 0.55	**0.31 +- 0.03**

with Adam (Kingma and Ba 2015). Table 4 lists all hyperparameters.

F Evaluation

Agent policies are evaluated as deterministic policies every 1% of training completed. An additional evaluation occurs before training to account for zero-shot transfer. Each evaluation records the return acquired from running 50 episodes in randomly sampled contexts from the context set on which we are training. The source policy is also evaluated in C_{tgt} to allow analysis of generalization throughout training. The returns are averaged to estimate the expected discounted return, with which we calculate the transfer utilities.

Let h_π be the training history of policy π, that is, a mapping from number of timesteps trained to the estimated expected discounted return of the policy at that time. For some training configuration (environment, context space, hyperparameters, etc.), denote the target and transferred policies by $\pi_{C_{\text{tgt}}}$ and $\pi_{C_{\text{src}},C_{\text{tgt}}}$, respectively. Denote some predetermined threshold by τ. We calculate our transfer utilities as follows:

$$\mathcal{U}_{TT}(\pi,\tau) = \min\{t | h_\pi(t) \geq \tau\} \quad \text{(Time to Threshold)}$$

$$\mathcal{U}_{JS}(\pi_{C_{\text{src}},C_{\text{tgt}}}, \pi_{C_{\text{tgt}}}) = h_{\pi_{C_{\text{src}},C_{\text{tgt}}}}(0) \quad \text{(Jumpstart)}$$

$$\mathcal{U}_{TR}(\pi_{C_{\text{src}},C_{\text{tgt}}}, \pi_{C_{\text{tgt}}}) = \frac{AUC(h_{\pi_{C_{\text{src}},C_{\text{tgt}}}}) - AUC(h_{\pi_{C_{\text{tgt}}}})}{AUC(h_{\pi_{C_{\text{tgt}}}})} \quad \text{(Transfer Ratio)}$$

[2]Target policy scored close to 0 on all seeds.

[3]Target policy scored 0 on all seeds.

where AUC is the area under the curve, estimated by the average of all recorded values on the curve. To provide a single TT value that considers all thresholds, we also calculate utility $\mathcal{U}_{TT_{AUC}}(\pi) = \underset{\tau}{AUC}(\mathcal{U}_{TT}(\pi,\tau))$.

G C-PREP with DQN

Algorithm 1 shows the DQN algorithm with C-PREP integration. A textual description of the algorithm is available in Section 3.

H Additional Experiments

We perform two additional experiments. The first aims to show C-PREP's sample efficiency in terms of the number of source contexts on which it is trained. For this, we doubled the size of the source context set and rerun the experiments. IQM transfer utlities for this experiment are available in Table 5. Fig. 8 shows the TT performance over the achieved threshold for these settings in the GN, MP, and PD environments. As we can see, the performance gap between our method and LTL decreased when we doubled the size of C_{src}. We conclude from this that C-PREP is able to maintain its transfer performance even when the size of the source context set is shrinks.

The second experiment tested the generalization capabilities of C-PREP using PCG representations. As we saw in Section 4.2 and analyzed in Section 4.3, we witness severe overfitting when using PCG. However, we find that using RM information is not completely futile. Figure Fig. 9 shows the performance of the source policy during training in C_{src},

Table 3: IQM and standard deviation transfer utilities for tested ablation configurations (environment-context space pairs) using both base context representation (CTL and PCG), aggregated over all seeds. The best results for each CMDP (row) and base context representation (left-right split) are marked in bold. Negative TR values that indicate non-beneficial transfer are colored red.

Utility	Context Space	Environment	CTL+LTL	CTL+DTL	CTL+RS	CTL+LTL+DTL	CTL+LTL+RS	CTL+C-PREP (ours)	CTL+LTL+C-PREP (ours)	PCG+LTL	PCG+DTL	PCG+RS	PCG+LTL+DTL	PCG+LTL+RS	PCG+C-PREP (ours)	PCG+LTL+C-PREP (ours)
TT_{AUC}	EL	GN	35.25 +- 6.97	48.41 +- 10.33	7.12 +- 0.66	53.00 +- 10.25	6.21 +- 0.42	6.42 +- 0.44	**5.69 +- 0.24**	80.30 +- 8.29	72.21 +- 15.60	7.12 +- 0.66	**5.56 +- 0.15**	5.89 +- 0.22	5.82 +- 0.28	84.37 +- 9.77
		MP	62.09 +- 15.17	84.31 +- 8.59	96.11 +- 1.14	94.12 +- 12.98	19.88 +- 9.36	**18.32 +- 9.27**	24.93 +- 6.72	98.04 +- 0.00	98.04 +- 0.00	96.11 +- 1.14	**54.92 +- 9.86**	95.02 +- 2.03	61.96 +- 9.37	98.04 +- 0.00
		PD	85.62 +- 6.37	95.42 +- 2.08	98.04 +- 0.00	96.08 +- 1.75	37.58 +- 23.38	21.68 +- 6.55	**17.07 +- 15.58**	98.04 +- 0.00	98.04 +- 0.00	98.04 +- 0.00	**43.21 +- 12.76**	96.75 +- 0.90	58.97 +- 6.69	98.04 +- 0.00
	CM	GN	30.94 +- 4.47	31.11 +- 2.80	20.01 +- 3.63	29.73 +- 3.06	7.14 +- 1.41	7.64 +- 1.72	**6.46 +- 1.33**	80.90 +- 5.40	91.53 +- 3.80	20.01 +- 3.63	**13.75 +- 2.15**	16.45 +- 2.37	14.58 +- 2.19	85.39 +- 7.17
		MP	59.05 +- 13.28	53.03 +- 19.86	51.29 +- 12.97	52.37 +- 20.12	14.24 +- 8.85	13.64 +- 8.60	**11.58 +- 11.46**	93.42 +- 4.32	98.04 +- 0.00	51.29 +- 12.97	33.52 +- 12.62	42.41 +- 12.14	**32.73 +- 13.21**	97.11 +- 3.67
		PD	82.42 +- 21.06	86.27 +- 15.43	55.94 +- 19.14	64.64 +- 24.99	37.52 +- 25.46	28.95 +- 24.15	**25.22 +- 20.73**	98.04 +- 2.83	98.04 +- 0.00	55.94 +- 19.14	40.65 +- 18.64	43.39 +- 16.48	**39.95 +- 18.23**	98.01 +- 2.92
	PO	ON	98.04 +- 0.00	98.04 +- 0.00	92.56 +- 13.58	98.04 +- 0.00	97.50 +- 5.59	**32.93 +- 10.21**	52.05 +- 33.84	98.04 +- 0.00	98.04 +- 0.00	92.56 +- 13.58	**34.35 +- 6.34**	93.01 +- 3.80	37.65 +- 22.90	98.04 +- 0.00
JS	EL	GN	0.05 +- 0.03	0.06 +- 0.02	0.00 +- 0.01	0.04 +- 0.03	0.07 +- 0.09	0.05 +- 0.05	**0.10 +- 0.06**	0.01 +- 0.01	0.01 +- 0.02	0.00 +- 0.01	**0.03 +- 0.03**	0.00 +- 0.01	0.00 +- 0.01	0.02 +- 0.02
		MP	0.38 +- 0.15	0.15 +- 0.08	0.00 +- 0.00	0.04 +- 0.13	0.48 +- 0.26	**0.49 +- 0.17**	0.31 +- 0.19	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	0.00 +- 0.01	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**
		PD	0.13 +- 0.07	0.04 +- 0.02	0.00 +- 0.00	0.04 +- 0.02	0.34 +- 0.20	0.38 +- 0.12	**0.53 +- 0.19**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**
	CM	GN	0.59 +- 0.05	0.62 +- 0.05	0.00 +- 0.02	0.61 +- 0.04	0.73 +- 0.05	0.75 +- 0.08	**0.76 +- 0.09**	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.02	**0.25 +- 0.14**	0.18 +- 0.14	0.10 +- 0.18	0.00 +- 0.06
		MP	0.39 +- 0.13	0.41 +- 0.20	0.00 +- 0.00	0.46 +- 0.20	0.66 +- 0.11	0.68 +- 0.10	**0.74 +- 0.17**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	0.00 +- 0.06	0.00 +- 0.02	**0.00 +- 0.00**	**0.00 +- 0.00**
		PD	0.13 +- 0.19	0.13 +- 0.14	0.00 +- 0.00	0.32 +- 0.23	0.38 +- 0.27	0.52 +- 0.25	**0.55 +- 0.25**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	0.00 +- 0.12	**0.00 +- 0.00**	0.00 +- 0.13	**0.00 +- 0.00**
	PO	ON	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	0.00 +- 0.00	**0.14 +- 0.26**	0.01 +- 0.20	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**	**0.00 +- 0.00**
TR	EL	GN	-0.24 +- 0.09	-0.44 +- 0.14	0.05 +- 0.01	-0.48 +- 0.13	**0.08 +- 0.03**	0.07 +- 0.03	0.06 +- 0.02	-0.80 +- 0.11	-0.69 +- 0.20	0.05 +- 0.01	0.05 +- 0.03	**0.05 +- 0.02**	0.04 +- 0.02	-0.86 +- 0.12
		MP	-0.96 +- 0.06	-0.95 +- 0.04	-0.85 +- 0.03	-1.00 +- 0.01	**0.24 +- 0.18**	0.16 +- 0.12	0.14 +- 0.07	-1.00 +- 0.00	-1.00 +- 0.00	-0.85 +- 0.03	-0.32 +- 0.20	-0.93 +- 0.13	**-0.31 +- 0.21**	-1.00 +- 0.00
		PD	-0.98 +- 0.01	-0.99 +- 0.03	-0.84 +- 0.09	-1.00 +- 0.00	-0.14 +- 0.38	0.14 +- 0.09	**0.20 +- 0.25**	-1.00 +- 0.00	-1.00 +- 0.00	-0.84 +- 0.09	**-0.10 +- 0.35**	-0.94 +- 0.07	-0.33 +- 0.16	-1.00 +- 0.00
	CM	GN	-0.07 +- 0.07	-0.09 +- 0.05	-0.02 +- 0.06	-0.07 +- 0.05	**0.11 +- 0.01**	0.11 +- 0.01	0.10 +- 0.02	-0.58 +- 0.13	-0.82 +- 0.09	-0.02 +- 0.06	-0.03 +- 0.04	-0.02 +- 0.04	**0.00 +- 0.02**	-0.69 +- 0.14
		MP	-0.68 +- 0.28	-0.71 +- 0.38	0.11 +- 0.18	-0.55 +- 0.34	**0.26 +- 0.06**	0.21 +- 0.06	0.15 +- 0.05	-0.78 +- 0.30	-1.00 +- 0.00	**0.11 +- 0.18**	-0.11 +- 0.04	-0.03 +- 0.08	-0.02 +- 0.03	-0.92 +- 0.23
		PD	-0.77 +- 0.33	-0.99 +- 0.19	**0.21 +- 0.21**	-0.46 +- 0.42	-0.05 +- 0.20	0.06 +- 0.19	0.03 +- 0.22	-1.00 +- 0.39	-0.99 +- 0.40	**0.21 +- 0.21**	-0.10 +- 0.07	0.06 +- 0.19	0.06 +- 0.06	-0.94 +- 0.48
	PO	ON	0.00 +- 0.00	0.00 +- 0.00	inf	0.00 +- 0.00	-0.87 +- 0.90	**0.69 +- 0.20**	-0.08 +- 0.73	0.00 +- 0.00	0.00 +- 0.00	inf	**0.69 +- 1.19**	-0.71 +- 1.79	0.31 +- 0.55	0.00 +- 0.00

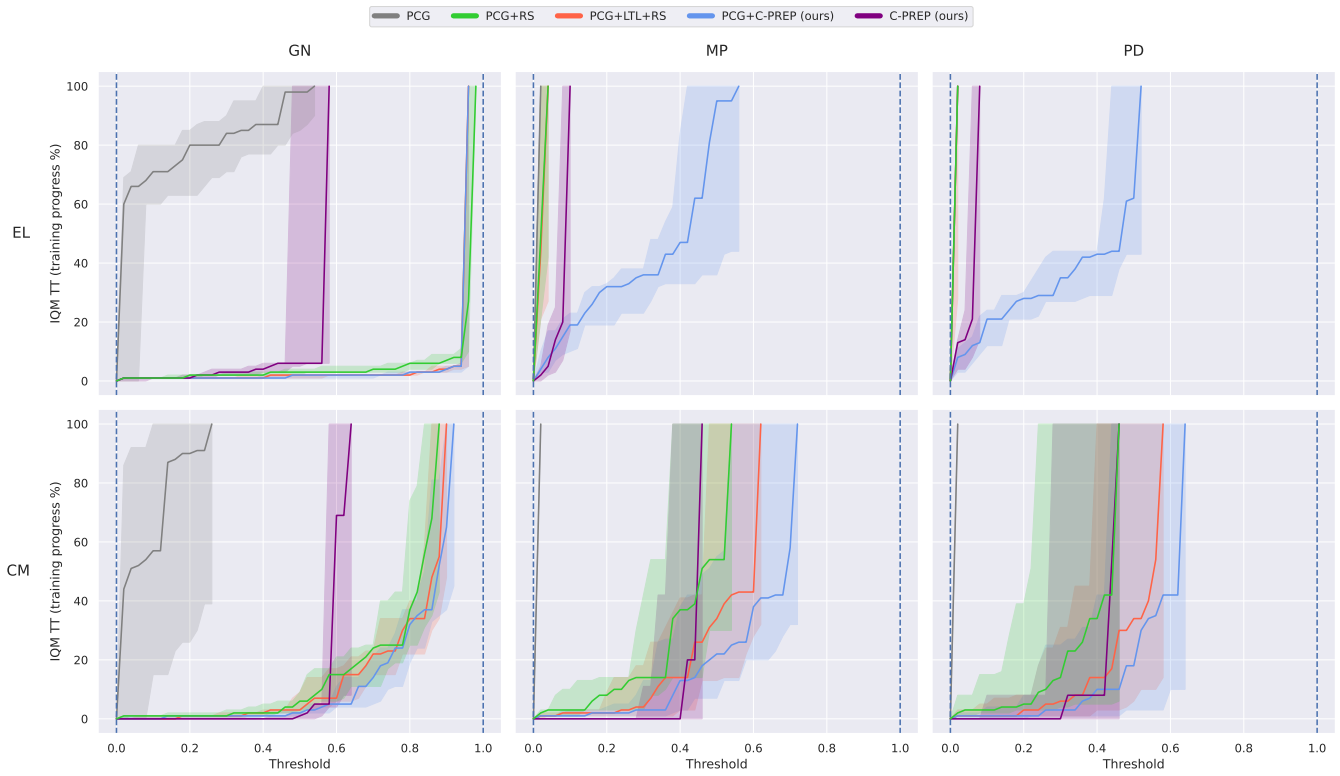


Figure 4: The IQM TT of configurations with PCG base context representations as a function of the threshold.

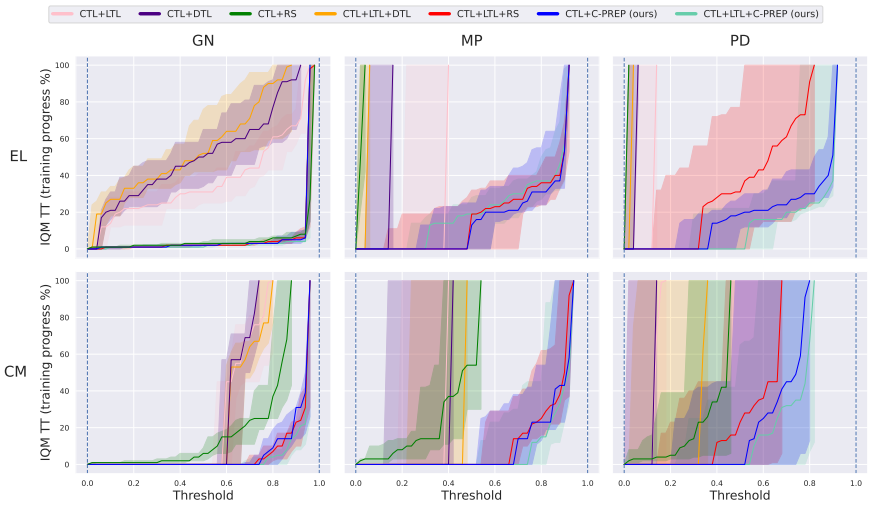
Table 4: DQN hyperparameters for all baseline configurations.

Parameter	Value
Target Q-network update interval	10,000 steps
Exploration time	50% training duration
replay buffer size	1M
Discount factor	0.99
training frequency	every 4 steps
gradient steps per training	4
sample batch size	32
Number of hidden layers	2
Hidden layer width	64
Hidden activations	ReLU
learning rate	0.0001
Adam β_1	0.9
Adam β_2	0.999

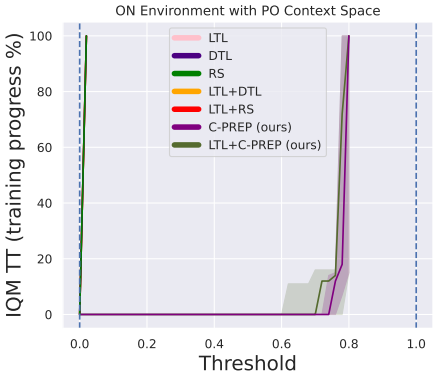
evaluated in C_{tgt} in the PD environment with the CM context space. We observe that in the first 30% of training there is a spike in performance for LTL +RS or C-PREP. These are the only configurations that use additional context representations from RM information to augment the contextual PCG input. The performance spike is far too high to be "luck" since otherwise other configurations should also display this phenomenon. This begs the question "What is learned before overfitting occurs and how can we preserve this knowledge"? Furthermore, we still do not have a clear explanation as to why specifically RM information causes this spike. We believe this has implications for representation learning, where agents learn latent representations of the state space in a disentangled manner from the policy.

I Multi-Taxi Environment

Multi-Taxi is a highly configurable multi-agent environment, based on the OpenAI gym taxi environment (Brockman et al. 2016), which adheres to the PettingZoo API (Terry et al. 2021). Fig. 10 shows a visualization of the environment. The environment's configurable features allow the user to set the number of passengers and taxis, the taxi's capacity and fuel requirements, the actions' stochasticity, the sensor function, and more. We note that while multi-taxi is natively a multi-agent environment, we explore it as a single-agent setting. By leveraging the domain's customizability we define seven very different environment settings of varying complexity levels based on three context spaces changing different aspects of the environment between tasks. The code is provided in the supplementary materials.



(a) Ablation configurations using CTL as the base context representation

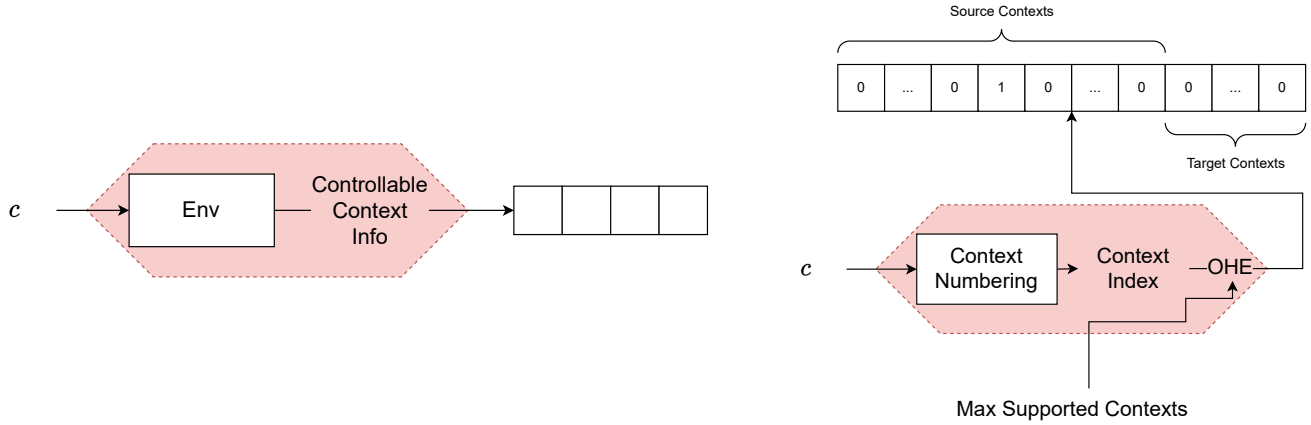


(b) Ablation configurations using CTL as the base context representation

Figure 5: The IQM TT of ablation test configurations as a function of the threshold.

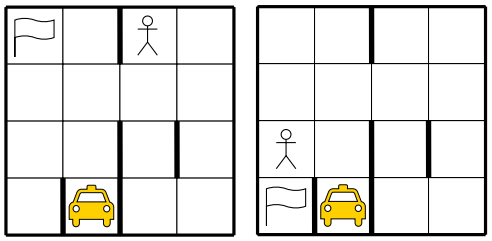
Algorithm 1: Training DQN with C-PREP

Input: $\pi = Q_\theta$ - initial DQN policy
Input: C - contexts set
Input: N- number of episodes to train
Input: G - RM generator function
Input: L - transition labeling function
$\mathcal{D} \leftarrow$ empty experience replay buffer
for $i \in [N]$ **do**
  $c \leftarrow sampleUniform(C)$
  $\langle U, \delta_u, \delta_r \rangle \leftarrow$ generated RM for c
  $V \leftarrow$ value iteration on RM
  $\delta_r(u,l) \leftarrow \delta_r(u,l) + \gamma V(\delta_u(u,l)) - V(u)$ {reward shaping}
  $u \leftarrow u_0$
  for each step in episode of $\mathcal{M}_c$ **do**
    $l^* \leftarrow \arg\max_{l \in 2^{\mathcal{P}}} \delta_r(u,l) + \gamma V(\delta_u(u,l))$
    $\hat{s} \leftarrow \langle s, l^* \rangle$ {using augmented state space}
    $q \leftarrow Q_\theta(\hat{s})$
    $a \leftarrow \epsilon$-greedy(q)
    $s' \leftarrow \mathcal{M}_c$.step$(a)$
    $u' \leftarrow \delta_u(u, L(s,a,s'))$
    $r \leftarrow \delta_r(u, L(s,a,s'))$ {using RM reward}
    $l^{*\prime} \leftarrow \arg\max_{l \in 2^{\mathcal{P}}} \delta_r(u',l) + \gamma V(\delta_u(u',l))$
    $\hat{s}' \leftarrow \langle s', l^{*\prime} \rangle$
    $\mathcal{D}$.store$(\hat{s}, a, r, \hat{s}')$
    $\hat{s}, a, r, \hat{s}' \leftarrow \mathcal{D}$.sampleBatch()
    $l \leftarrow \sum_{\text{batch}} (r + \max_{a'} \{Q_\theta(\hat{s}')[a']\} - Q_\theta(\hat{s})[a])^2$
    $\theta \leftarrow \nabla_\theta l$
    $s, u \leftarrow s', u'$
  end for
end for
Return Q_θ

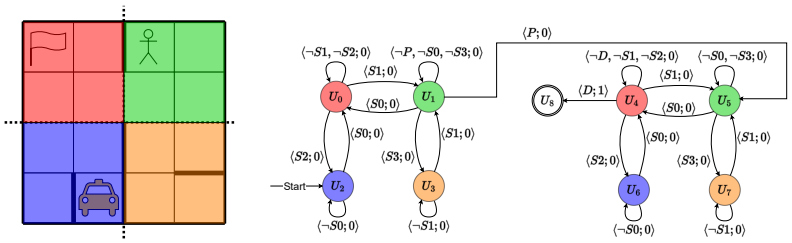


(a) (b)

Figure 6: Baseline context representation functions. (a) Controllable (CTL) environment context representation function. Each context is represented using the original informative context representation offered by the environment. (b) Procedural content generation (PCG) context representation function. Contexts have unique indices that are converted into binary indicator vectors (one-hot-encoding) for the source and target contexts together. One-hot encoding context representations are limited to a maximum number of supported contexts, determined ahead of time.



(a) Two different EL contexts in the PD environment. The taxi icon is the acting agent, the person icon is the passenger to be picked up, and the flag icon is passenger's destination. The agent can navigate to any adjacent cell, but cannot cross thick walls.



(b) An example abstraction of a task in the PD environment on a 4×4 grid. The environment is environment is split into four sectors of 2×2 cells. We say that the agent is in sector i if it is currently located in one of the cells within the sector. Symbol `Si` indicates the agent is in sector `i`. Symbols `P` and `D` indicate the passenger has been picked up or dropped off, respectively. The colors match abstract states to their corresponding sectors.

Figure 7

Table 5: IQM and standard deviation transfer utilities for all tested configurations (environment-context space pairs) using CTL as the base context representation trained on doubly sized source context set before transfer, aggregated over all seeds. The best results for each CMDP (row) are marked in bold. Negative TR values that indicate non-beneficial transfer are colored red.

Utility	Context Space	Environment	CTL	CTL+RS	CTL+LTL+RS	CTL+C-PREP (ours)	C-PREP (ours)
TT_{AUC}	EL	GN	19.07 ± 3.10	6.26 ± 0.51	$\mathbf{5.92 \pm 0.27}$	6.13 ± 0.70	43.95 ± 4.74
		MP	86.27 ± 7.70	18.63 ± 6.06	$\mathbf{10.03 \pm 0.69}$	10.24 ± 0.92	87.03 ± 1.74
		PD	94.12 ± 15.42	68.17 ± 24.76	12.96 ± 3.58	$\mathbf{11.70 \pm 0.58}$	87.92 ± 1.39
	CM	GN	37.99 ± 1.16	10.22 ± 2.15	$\mathbf{5.40 \pm 0.88}$	5.52 ± 1.09	34.74 ± 4.14
		MP	67.83 ± 13.45	20.97 ± 16.25	10.96 ± 6.24	$\mathbf{10.85 \pm 4.82}$	49.75 ± 9.38
		PD	78.22 ± 22.00	44.16 ± 23.82	27.30 ± 27.32	$\mathbf{22.16 \pm 19.23}$	48.33 ± 15.64
	PO	ON	98.04 ± 0.00	97.10 ± 16.40	96.31 ± 6.65	23.29 ± 12.84	$\mathbf{22.69 \pm 1.48}$
JS	EL	GN	$\mathbf{0.42 \pm 0.15}$	0.21 ± 0.06	0.08 ± 0.15	0.26 ± 0.12	0.00 ± 0.02
		MP	0.13 ± 0.08	0.69 ± 0.04	$\mathbf{0.87 \pm 0.02}$	0.86 ± 0.04	0.03 ± 0.02
		PD	0.06 ± 0.16	0.19 ± 0.23	0.78 ± 0.09	$\mathbf{0.80 \pm 0.03}$	0.00 ± 0.02
	CM	GN	0.52 ± 0.06	0.76 ± 0.05	$\mathbf{0.93 \pm 0.01}$	0.91 ± 0.04	0.53 ± 0.06
		MP	0.31 ± 0.13	0.66 ± 0.19	0.83 ± 0.11	$\mathbf{0.84 \pm 0.09}$	0.29 ± 0.14
		PD	0.12 ± 0.23	0.43 ± 0.23	$\mathbf{0.67 \pm 0.29}$	0.66 ± 0.28	0.36 ± 0.17
	PO	ON	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.74 ± 0.31	$\mathbf{0.78 \pm 0.01}$
TR	EL	GN	-0.02 ± 0.07	$\mathbf{0.14 \pm 0.04}$	0.08 ± 0.03	0.08 ± 0.03	0.08 ± 0.04
		MP	-0.98 ± 0.12	$\mathbf{0.56 \pm 0.36}$	0.33 ± 0.07	0.27 ± 0.07	0.11 ± 0.15
		PD	-0.98 ± 0.03	$\mathbf{0.61 \pm 0.64}$	0.43 ± 0.48	0.32 ± 0.05	-0.00 ± 0.14
	EL	GN	-0.05 ± 0.07	$\mathbf{0.23 \pm 0.02}$	0.13 ± 0.01	0.14 ± 0.01	0.09 ± 0.04
		MP	-0.75 ± 0.30	$\mathbf{0.36 \pm 0.07}$	0.30 ± 0.09	0.28 ± 0.05	0.10 ± 0.04
		PD	-0.29 ± 0.37	$\mathbf{0.36 \pm 0.31}$	0.11 ± 0.24	0.26 ± 0.06	0.08 ± 0.04
	PO	ON	0.00 ± 0.00	0.45 ± 0.88	-0.80 ± 7.12	$\mathbf{0.83 \pm 0.34}$	0.33 ± 0.08

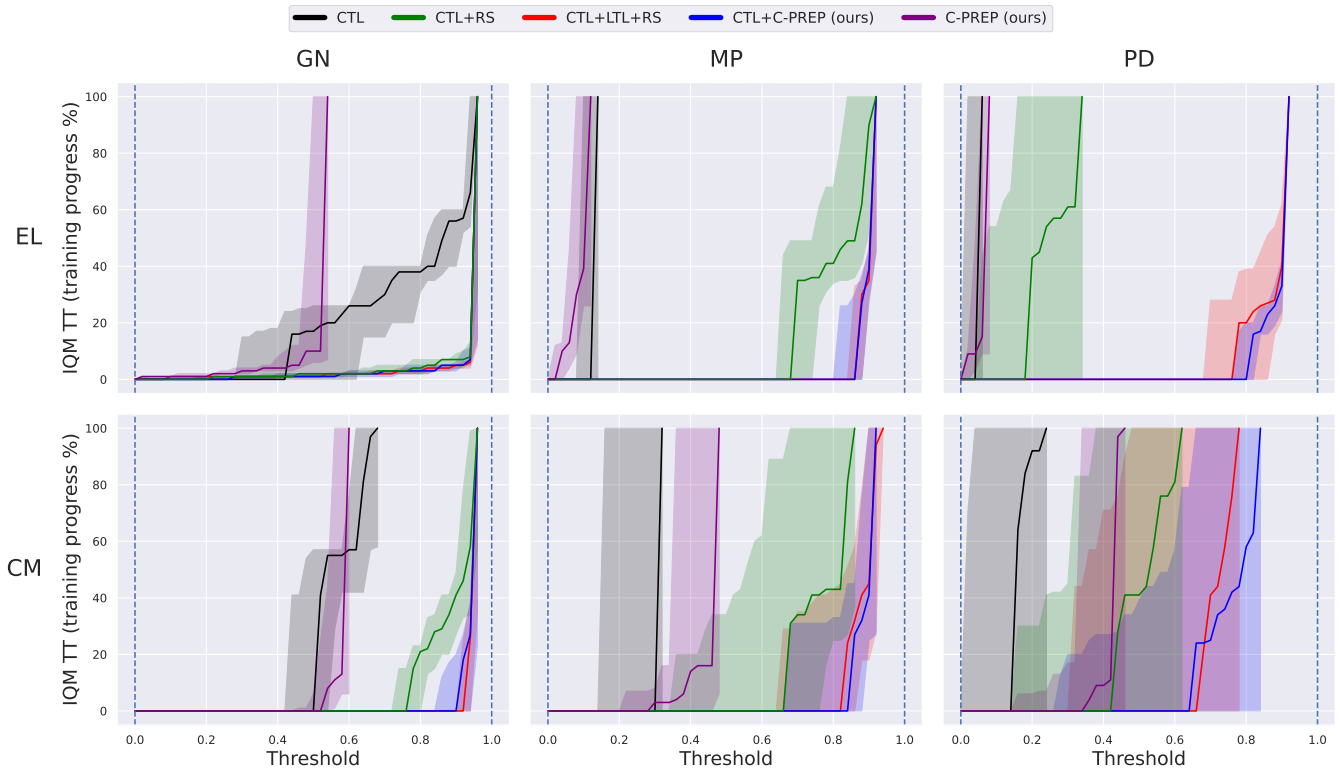


Figure 8: The IQM TT, measured in percentage of completed training, as a function of the threshold in environments GN, MP, PD, with doubley sized C_{src}. Each color represents a different configuration, specified in the legend. The shaded areas indicate stratified bootstrap 95% confidence intervals. Each row corresponds to a different context space, indicated to the left of the row. Each column corresponds to a different environment, indicated above the column.

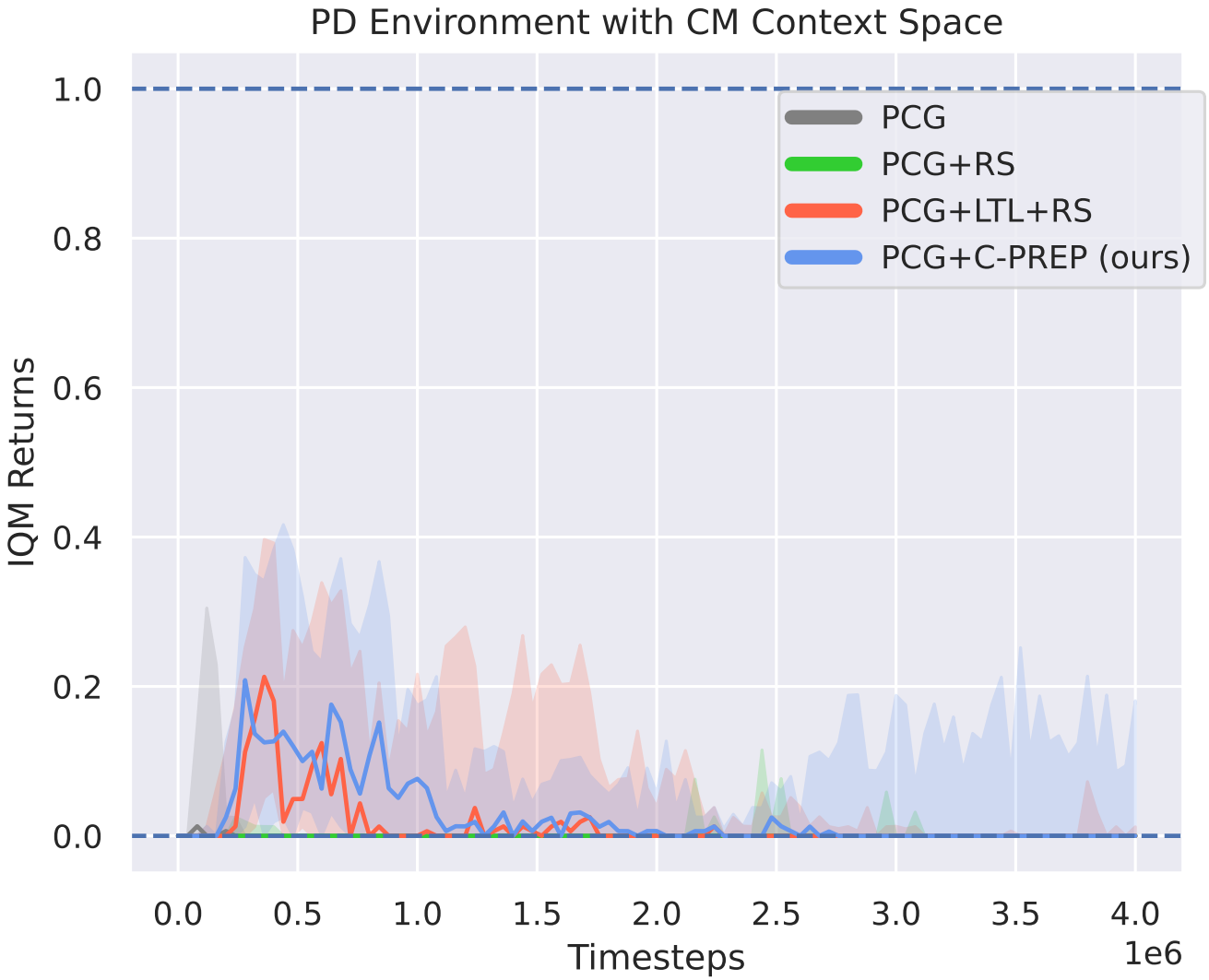


Figure 9: The IQM performance of the source policy, evaluated on the target context set. in environment PD and context space CM with PCG configurations. Each color represents a different configuration, specified in the legend. The shaded areas indicate stratified bootstrap 95% confidence intervals. Each row corresponds to a different context space, indicated to the left of the row. Each column corresponds to a different environment, indicated above the column.

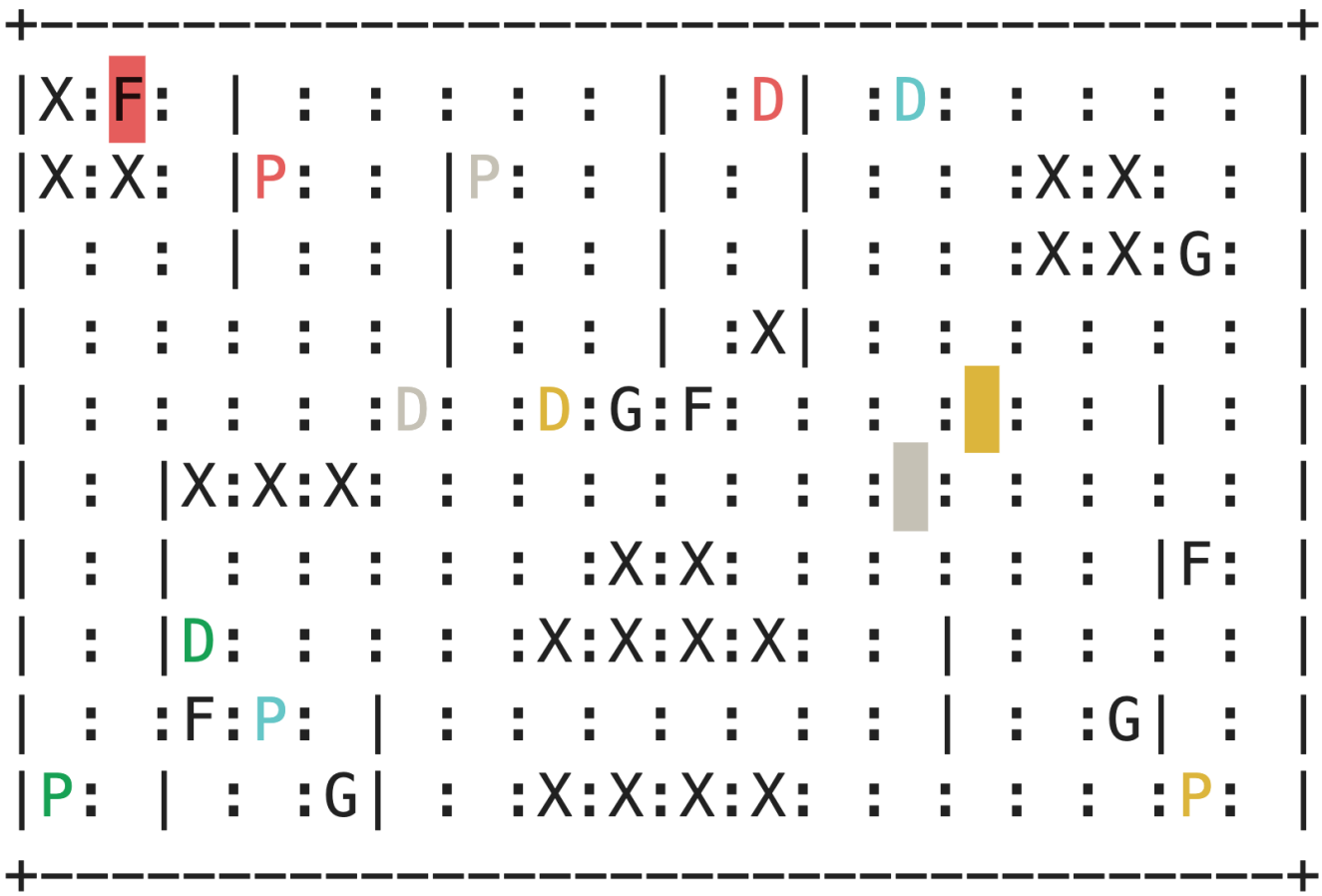


Figure 10: A visualization of the Multi-Taxi environment. Colored rectangles are taxi agents, P and D symbols indicate the location of a passenger and its corresponding destination (respectively). 'X' and '—' values indicate different kinds of obstacles. 'F' and 'G' are two different kinds of fuel stations.