Personalized Reinforcement Learning with a Budget of Policies

Dmitry Ivanov, Omer Ben-Porat

Technion, Israel
divanov@campus.technion.ac.il, omerbp@technion.ac.il

Abstract

Personalization in machine learning (ML) tailors models' decisions to the individual characteristics of users. While this approach has seen success in areas like recommender systems, its expansion into high-stakes fields such as healthcare and autonomous driving is hindered by the extensive regulatory approval processes involved. To address this challenge, we propose a novel framework termed represented Markov Decision Processes (r-MDPs) that is designed to balance the need for personalization with the regulatory constraints. In an r-MDP, we cater to a diverse user population, each with unique preferences, through interaction with a small set of representative policies. Our objective is twofold: efficiently match each user to an appropriate representative policy and simultaneously optimize these policies to maximize overall social welfare. We develop two deep reinforcement learning algorithms that efficiently solve r-MDPs. These algorithms draw inspiration from the principles of classic K-means clustering and are underpinned by robust theoretical foundations. Our empirical investigations, conducted across a variety of simulated environments, showcase the algorithms' ability to facilitate meaningful personalization even under constrained policy budgets. Furthermore, they demonstrate scalability, efficiently adapting to larger policy budgets.

1 Introduction

Personalization in machine learning (ML) tailors the decision-making process of a model to align with an individual's unique characteristics and preferences. This approach is typically realized either through individual-specific models or by fine-tuning a universal model with personal data. It is successfully applied in various fields such as recommender systems (Shepitsen et al. 2008; Lee, Sun, and Lebanon 2012; Yao et al. 2020), natural language processing (Wu et al. 2023), healthcare (Ayer, Alagoz, and Stout 2012), and financial services (Capponi, Olafsson, and Zariphopoulou 2022). For instance, in recommender systems, personalization enables models to suggest products or services to users based on their individual purchase histories and browsing behaviors.

Despite these successes, the integration of personalization in ML into critical sectors like healthcare and autonomous driving, where errors can lead to severe consequences, remains limited. Products driven by ML must undergo extensive regulatory review and approval processes to ensure they offer benefits that significantly outweigh potential risks for their intended user populations. The review process, as exemplified by the Artificial Pancreas that monitors and controls glucose levels (Breton et al. 2020), involves a thorough evaluation by regulatory bodies like the Food and Drug Administration (FDA) to affirm the balance of benefits and risks. The FDA's prolonged authorization of a comprehensive Artificial Pancreas solution, spanning several years (JDRF 2022), underscores the complexity and rigor of such evaluations. Similarly, autonomous vehicles employing reinforcement learning (RL) systems for navigation confront formidable challenges. Despite accumulating millions of hours in test driving, these vehicles must pass meticulous review and audit processes before entering production. The integration of personalization in these systems, necessitating the assessment of individualized policies for safety and efficacy, further complicates the regulatory landscape.

The challenges observed in the aforementioned domains reflect a broader issue: in high-risk and complex environments, the primary obstacle often lies not in data acquisition or hardware limitations, but in the protracted regulatory approval process. This bottleneck necessitates an innovative approach that balances regulatory feasibility with the benefits of personalization. Our proposed solution is to develop a limited number of tailored policies, each catering to a specific user group, thereby streamlining the review process while maintaining the personalization advantage.

In this context, we model our scenario as a Markov decision process (MDP) involving a population of n users, or agents, each characterized by a unique reward function reflecting their preferences within the MDP. Ideally, each agent would be offered a distinct personalized policy. However, given the regulatory constraints highlighted above, we propose a more practical strategy: the development of at most $k < n$ policies. Under this framework, each agent selects the most appropriate policy from this smaller set.

To formalize this concept, we introduce a novel abstraction: the represented MDP (r-MDP). In r-MDP, agents do not directly engage with the MDP. Instead, they are aligned with k representatives, each managing a single policy within the MDP. Agents associated with the same representative adhere to the same policy. The goal is to optimally match agents to

representatives and train these representative policies to maximize the overall social welfare of the agents. This approach addresses the regulatory challenges by reducing the number of policies requiring approval, thereby facilitating a more efficient review process without significantly compromising the personalization benefits.

Our proposed pipeline can be summarized in three stages:

1. **Manufacturing**. The k policies are trained in a simulator to jointly maximize the welfare of n agents in an r-MDP. Taking self-driving cars as an example, this stage involves developing k driving policies based on aggregated user preferences (e.g., gathered from surveys). *Our primary focus is on this stage.*
2. **Assessment**. At this stage, regulatory authorities evaluate the developed policies. The costs incurred here stem from the extensive review process and potential requests for policy modifications. The number of policies, k, naturally balances the degree of personalization offered by each policy against the assessment costs.
3. **Deployment**. Following successful assessment, the policies are authorized for real-world deployment.

As we discuss later, solving r-MDP directly is intractable. However, we can simplify the problem by separating it into two more manageable sub-problems: optimizing policies given fixed assignments and optimizing assignments given fixed policies. Drawing inspiration from the classic K-means (MacQueen 1967; Lloyd 1982) and Expectation-Maximization (EM) (Dempster, Laird, and Rubin 1977) clustering algorithms, we introduce our first algorithm, which iteratively updates policies and assignments. Moreover, recognizing the differentiability of policy objectives with respect to assignments, we propose our second algorithm employing gradient descent for end-to-end training. We provide theoretical guarantees of monotonic improvement and convergence to a local maximum of social welfare using our algorithms.

Our empirical analysis encompasses Resource Gathering environment (Barrett and Narayanan 2008) and four MuJoCo (Todorov, Erez, and Tassa 2012) tasks, adapted as r-MDPs. The results consistently demonstrate that our algorithms surpass existing baselines in performance. Notably, we observe that even a limited number of policies can provide significant personalization, highlighting the efficacy of our approach.

Our contributions

1. **Problem Formulation**: We introduce a nuanced problem formulation in the realm of personalized RL, emphasizing the challenge posed by the resource-intensive review and authorization process for personalized policies.
2. **Novel Setting**: We propose the r-MDP framework that addresses the need for a practical compromise between the desire for high personalization and the constraints of expedited regulatory review processes.
3. **Efficient Algorithms**: We present two deep RL algorithms, backed by robust theoretical justifications, to approximately solve r-MDPs. These algorithms demonstrate superior performance in achieving personalized outcomes compared to approaches from existing literature.

Limitations This study primarily addresses the challenge of training a limited number of policies for a large user base in the Manufacturing stage of our pipeline, leaving the complexities of the Assessment and Deployment stages, such as policy revisions and real-world performance stability, for future exploration. Additionally, our focus on utilitarian social welfare may inadvertently lead to uneven reward distributions between agents. We discuss potential alternatives in Section 2.2. Finally, while we use parameter sharing to enhance sample efficiency, the possibility of further improvements through advanced techniques remains. Nevertheless, given the controlled nature of simulator training, sample efficiency is a secondary concern in our study.

1.1 Related Work

Related works in personalization, multiple objectives, and multi-agent systems provide valuable context for our research, yet none directly address the unique challenge of operating within an explicitly constrained policy budget.

RL for personalization This field aims at creating tailored RL solutions for individuals or groups. For an in-depth review, see (den Hengst et al. 2020). A key challenge is personalizing RL policies in real-world applications, especially in healthcare (Hassouni et al. 2018; Zhu et al. 2018; Grua and Hoogendoorn 2018; El Hassouni et al. 2019; el Hassouni et al. 2022). While offering a single policy to all users can be suboptimal, training a policy per user can be an inefficient use of collected samples. A common strategy involves clustering users by their behavior to train cluster-specific policies. Unlike these approaches where clustering is driven by sample efficiency, our approach addresses real-world policy implementation costs, with training conducted in a simulator. Still, the trajectory-clustering concept is relevant to our framework and serves as a baseline in our experiments. We also acknowledge works that use external data for clustering (Martin and Arroyo 2004; Goindani and Neville 2020), but our methods do not require such data.

Other aspects in RL for Personalization include privacy-respecting data sharing (Tabatabaei, Hoogendoorn, and van Halteren 2018; Baucum et al. 2022), which can be addressed with Federated RL (Nadiger, Kumar, and Abdelhak 2019), and exploration under safety constraints (Perkins and Barto 2002; Hans et al. 2008; Moldovan and Abbeel 2012; Junges et al. 2016). Though significant, these challenges do not align closely with our specific research focus.

Meta-RL While Meta-RL aims for policy adaptability to an unlimited number of tasks (Finn, Abbeel, and Levine 2017), r-MDP imposes a strict constraint on the number of policies. A capable meta-policy could offer a personalized solution to each user in the absence of such a constraint, but optimally choosing a limited subset of policies to meet the needs of all users is a unique challenge of our framework. Note that there exists a potential for synergy: Meta-RL could provide a versatile policy that our algorithms would deploy strategically within the explicit policy budget. This synergy emphasizes that the two frameworks address distinct but potentially complementary aspects of the RL problem space.

Multi-Objective RL (MORL) Similarly to our setting, MORL involves optimizing multiple rewards. However, MORL typically focuses on either developing a single policy that balances various objectives or approximating the Pareto front with a potentially large set of policies (Hayes et al. 2022). While the latter algorithms could technically be adapted to our setting, for instance by selecting k policies from the Pareto set, they only apply to problems with a few reward functions. In contrast, we tackle problems with as many as a thousand reward functions, which underscores the scalability of our framework.

Multi-Agent RL (MARL) While superficially similar, MARL differs fundamentally from our framework. MARL involves multiple agents acting and interacting within a shared environment, often formalized as a Markov game (Littman 1994). In contrast, our framework trains policies that operate in a single-agent environment independently, without inter-policy interaction. This key distinction sets our work apart from the interactive dynamics central to MARL, emphasizing our focus on individual preference optimization.

2 Background and Problem Setup

2.1 Markov Decision Process

A Markov Decision Process (MDP) is a tuple $\mathcal{M} = (S, A, \mathcal{T}, \mathcal{T}_0, r, \gamma)$, where: S is the set of all states s; A is the set of all actions a available to the agent; $\mathcal{T} : S \times A \to \Delta(S)$ is the transition function that specifies the distribution of next states, where Δ denotes a set of discrete probability distributions; $\mathcal{T}_0 = \Delta(S)$ specifies the distribution of initial states s_0; $r : S \times A \to \mathcal{P}(\mathbb{R})$ is the reward function that specifies the distribution of rewards, where $\mathcal{P}$ is a set of continuous probability distributions; $\gamma \in (0, 1)$ is the discounting factor.

Let $\pi : S \to \Delta(A)$ be a policy. Given $s \in S$, $\pi(s, a)$ denotes the probability assigned to action a. A transition is a tuple $(s, a, \tilde{r}, s')$, where $a \sim \pi(s)$, $\tilde{r} \sim r(s, a)$, and $s' \sim \mathcal{T}(s, a)$. An episode is a sequence of transitions, in which each transition corresponds to a time step $t = 0, 1, \ldots, T$. The episode starts at time step $t = 0$ and progresses until the terminal time step T, which marks the horizon.

$R_t = \sum_{l=t}^{T} \left[\gamma^{l-t} \tilde{r}_l\right]$ is a return of an episode at time step t. The value function $V^\pi : S \to \mathbb{R}$ is defined as $V^\pi(s) \equiv V(s \mid \pi) = \mathbb{E}[R_t \mid s_t = s, \pi]$. The objective is to find the policy that maximizes the value function in all states:

$$\arg\max_\pi \mathbb{E}_{\mathcal{T},\mathcal{T}_0} V^\pi(s) = \arg\max_\pi \mathbb{E}_{s_0 \sim \mathcal{T}_0} V^\pi(s_0). \quad (1)$$

2.2 Represented Markov Decision Process

We define a represented MDP (r-MDP) as a tuple $\mathcal{M}_r = (S, A, \mathcal{T}, \mathcal{T}_0, \gamma, N, K, (r^i)_{i \in N})$, where: $S, A, \mathcal{T}, \mathcal{T}_0, \gamma$ are defined above; N is the set of agents, where $|N| = n$; K is the set of representatives, where $|K| = k < n$ is the budget of policies; $r^i : S \times A \to \mathcal{P}(\mathbb{R})$ is a reward function of $i \in N$.

In r-MDPs, agents do not interact with the environment directly. Instead, each agent $i \in N$ is represented by one of k representatives $j \in K$ that acts for them. Denote $\pi^j : S \to \Delta(A)$ as the j-th representative policy; and $\alpha^i \in \Delta(K)$ as the i-th agent's assignment, with $\alpha^i(j)$ denoting the probability of agent i being represented by j. The objective in r-MDP is to both match agents with representatives and train the representative policies such that the utilitarian social welfare of all agents is maximized:

$$\max_{(\alpha^i)_{i \in N}, (\pi^j)_{j \in K}} \sum_{i,j} \alpha^i(j) \mathbb{E}_{\mathcal{T}_0} V^{ij}(s_0), \quad (2)$$

where $V^{ij}(s) = V^i(s \mid \pi^j) = \mathbb{E}[R^i_t \mid s_t = s, \pi^j]$ is the value function of agent i assigned to representative j.

Note that representatives are an abstraction to distinguish the actors in the environment and the agents. In particular, representatives do not have intrinsic reward functions and maximize the assigned agents' welfare. Each representative effectively interacts with its copy of the environment with identical dynamics but different reward functions (see the definition of M^j in Section 3.1).

Applications The development of represented Markov Decision Processes (r-MDPs) primarily addresses the challenge of designing personalized ML solutions subject to rigorous regulatory assessments, as highlighted in the introduction. Beyond this primary motivation, r-MDPs hold broader applicability in scenarios where solution quantity is constrained.

Take, for instance, a financial institution formulating portfolios for multiple Exchange-Traded Funds (ETFs). The institution aims to cater to a diverse range of investor preferences, such as risk tolerance, asset types, and market exposure, informed by market data or surveys. However, offering a unique portfolio to each investor is impractical, necessitating a compromise on the number of ETFs. This scenario is algorithmically solved for one-dimensional preferences (Diana et al. 2021). In more complex, multi-dimensional cases, r-MDPs offer a viable modeling approach. By leveraging our r-MDP framework and the associated algorithms, the financial institution can optimally balance the diversity of ETF offerings with the practical limitations on the number of available portfolios, demonstrating the adaptability and utility of r-MDPs in varied contexts beyond regulatory constraints.

Limitations Optimizing utilitarian social welfare may result in unfair reward distributions between agents. Egalitarian or Nash-product social welfare may be more reasonable in applications where this is a concern. To this end, the techniques from socially fair RL (Mandal and Gan 2022) and clustering (Kar et al. 2023) could potentially be adapted. However, this direction is out of the scope of this paper.

2.3 Proximal Policy Optimization

PPO (Schulman et al. 2017) is a deep RL algorithm based on the prominent Actor-Critic framework.

The critic is a neural network ϕ that parameterizes an approximation of the agent's value function $\tilde{V}(s)$. It is trained with gradient descent to minimize the mean squared difference with a target value:

$$L(\phi) = \sum_{t \in B} \left[\tilde{A}_\phi(s_t, a_t) = (y(s_t, a_t) - \tilde{V}_\phi(s_t))\right]^2, \quad (3)$$

where L denotes a loss function, B denotes a batch of transitions, y denotes a target value, and $\tilde{A}_\phi(s_t, a_t)$ denotes an approximation of the advantage, which we estimate using generalized advantage estimation (Schulman et al. 2015b).

The actor is a neural network θ that parameterizes the policy π. It is trained to minimize the negated clipped surrogate objective:

$$L(\theta) = -\sum_t \text{clip}(\rho_\theta(s_t, a_t))\tilde{A}(s_t, a_t). \tag{4}$$

where $\rho_\theta(s, a) = \frac{\pi_\theta(s,a)}{\pi_{old}(s,a)}$ is a ratio between the policy that is being optimized and the policy that collected the experience, and $\text{clip}(\cdot)$ truncates the argument according to a specific rule that we report in the Appendix. Repeatedly updating on this objective using a batch of experience divided into smaller mini-batches approximates a policy update within a trust region (Schulman et al. 2015a).

Multiple technical details can make or break an implementation of PPO. For our experiments, we relied on (Huang et al. 2022) and were able to replicate the performance reported in the original PPO paper.

3 Our Approach and Algorithms

In this section, we describe our factorized approach to r-MDPs and propose two factorized deep RL algorithms.

3.1 Factorized Approach

Directly optimizing (2) involves finding the optimal joint assignment from a set exponential in the number of agents. Even if restricted to discrete assignments, the cardinality of this set is K^n, making the problem intractable for large n.

Consider a simplification of the joint objective (2) where the policies π^j are fixed for all $j \in K$. Then, maximizing it reduces to independently solving a set of trivial problems:

$$\max_{\alpha^i} \left[V^i = \sum_j \alpha^i(j) \mathbb{E}_{\mathcal{T}_0} V^{ij}(s_0) \right], \tag{5}$$

The optimal solution is to greedily assign agent i to the best-performing representative j^* (assuming its uniqueness):

$$\alpha^i(j^*) = 1 \iff j^* = \arg\max_j Q^i(j), \tag{6}$$

where $Q^i(j) = \mathbb{E}_{\mathcal{T}_0} V^{ij}(s_0)$ is the Q-value of assigning i to j. Ties can be broken arbitrarily. This quantity can be empirically approximated with Monte-Carlo sampling for each (i, j) as an average welfare over several episodes.

Consider another simplification of (2) where the assignments α^i are fixed for all $i \in N$. Then, optimizing social welfare over the policies reduces to independently solving a set of MDPs $(\mathcal{M}^j = (S, A, \mathcal{T}, \mathcal{T}_0, r^j, \gamma))_{j \in K}$, where $r^j(s, a) = \sum_i \alpha^i(j) r^i(s, a)$. The objective in $\mathcal{M}^j$ is:

$$\max_{\pi^j} \Big[\mathbb{E}_{\mathcal{T}_0} V^j(s_0) = \sum_i \alpha^i(j) \mathbb{E}_{\mathcal{T}_0} V^{ij}(s_0) \Big], \tag{7}$$

where $V^j(s)$ is the value of policy π^j in state s.

We define the factorized approach as an independent optimization of objectives (5) and (7). That is, each assignment is myopically optimized given the current policies, and vice versa. We are interested in designing factorized algorithms that approximate optimal solutions to the joint objective (2).

3.2 Training the Representatives

Before describing our algorithms that simultaneously train assignments and policies, we focus on the latter given fixed assignments. As the backbone, we use the PPO algorithm described in Section 2.3, but note that any other Actor-Critic algorithm would suffice.

Recall that a representative optimizes the expected value function defined in (7). Estimating it requires the summation of values $V^{ij}(s)$ over all agents weighted by the assignment probabilities $\alpha^i(j)$, which change throughout training. Because of this, directly parameterizing $\tilde{V}^j(s)$ with a neural network ϕ^j (as a direct application of the Actor-Critic approach would suggest) results in a non-stationary objective for the critic. Instead, we parameterize $\tilde{V}^{ij}(s)$. Specifically, a critic parameterized with ϕ^j outputs n values $\tilde{V}^{ij}_{\phi^j}(s)$, representing the welfare of each agent when assigned to j. Each output is trained to minimize the loss function (3) given rewards sampled from $r^i(s, a)$ and actions sampled from $\pi^j(s)$:

$$L(\phi^j) = \sum_{i,t} \left(\tilde{A}^{ij}_{\phi^j}(s_t, a_t) \right)^2, \tag{8}$$

where $\tilde{A}^{ij}_{\phi^j}(s_t, a_t) = (y^{ij}(s_t, a_t) - \tilde{V}^{ij}_{\phi^j}(s_t))$. The marginal advantage $\tilde{A}^j_{\phi^j}(s, a) = \sum_i \alpha^i(j) \tilde{A}^{ij}_{\phi^j}(s, a)$ is estimated according to the current assignments. Then, an actor θ^j that parameterizes π^j can be trained on the objective (4):

$$L(\theta^j) = -\sum_t \text{clip}(\rho^j_{\theta^j}(s_t, a_t)) \tilde{A}^j_{\phi^j}(s, a). \tag{9}$$

To improve training efficiency, we share the parameters of intermediate layers between actors θ^j, as well as critics ϕ^j.

Note that training the policies requires the experiences of all representatives acting for all agents. However, since the dynamics are identical, performing one transition with a representative can be used to sample rewards for all agents.

3.3 Hard Assignment via EM-like Learning

Our EM-like algorithm is inspired by the classic K-means (MacQueen 1967; Lloyd 1982) and Expectation-Maximization (EM) (Dempster, Laird, and Rubin 1977) clustering algorithms. It alternates between two steps. At the E-step, agents are assigned to representatives in analogy to points being assigned to clusters. At the M-step, representatives' policies are improved given the assignments of agents in analogy to cluster centers being improved given the assignments of points.

We maintain an $n \times k$ table $\tilde{Q}$, each element of which $\tilde{Q}^{ij}$ approximates the corresponding Q-value $Q^i(j)$ (defined in Section 3.1). Before performing the E-step, the elements of table $\tilde{Q}$ are updated as moving averages:

$$\tilde{Q}^{ij} \leftarrow (1-\lambda)\tilde{Q}^{ij} + \lambda[R_0^i \sim \pi^j], \tag{10}$$

where $\lambda \in (0, 1]$ is a mixing coefficient. Technically, for each assignment α^i, this update rule is Q-learning with learning rate λ in a stateless environment, albeit non-stationary since policies change over time. At the E-step, each agent is greedily reassigned to a best-performing representative, approximating (6):

$$\alpha^i(j^*) = 1 \iff j^* = \arg\max_j \tilde{Q}^{ij}. \tag{11}$$

At the M-step, we update policies with PPO as described in Section 3.2. A crucial trade-off is that of the frequency of E-steps and the magnitude of M-steps. We found it best to perform an E-step as frequently as possible, resulting in an M-step that corresponds to a single PPO update per policy.

Similarly to K-means, our algorithm can be proven to converge to a local optimum. We formulate this as a theorem:

Theorem 1. *Given an r-MDP, the EM-like algorithm converges to a local maximum of utilitarian social welfare.*

The proof is provided in the Appendix. Assuming that the M-step is performed until convergence with an (RL) algorithm with global convergence guarantees, we show that both the E-step and M-step monotonically improve social welfare.

3.4 Soft Assignment via End-to-End Learning

Our second algorithm is based on an observation that the loss function of a representative policy (9) is differentiable with respect to the assignment probabilities $\alpha^i(j)$. We leverage this by parameterizing assignments α^i for all agents with ψ and updating the parameters to minimize the same loss function as the policies. The resulting loss function for ψ is:

$$L(\psi) = -\sum_{j,t} \text{clip}(\rho^j_{\theta^j}(s_t, a_t)) \sum_i \alpha^i_\psi(j) \tilde{A}^{ij}_{\phi^j}(s_t, a_t). \tag{12}$$

This parameterization is implemented as an $n \times k$ table ψ of logits that are transformed into $\alpha^i_\psi(j)$ by applying column-wise softmax function so that $\sum_j \alpha^i_\psi(j) = 1$. These logits can be updated in the same backward pass as the actors θ^j since they share the loss function.

The intuition behind this update rule is that the probability $\alpha^i_\psi(j)$ only increases for the best-performing representative, i.e., such j that maximizes the advantage $\tilde{A}^{ij}(s, a)$ averaged over the mini-batch. Effectively, this is a relaxation of the hard re-assignment (11) of our EM-like algorithm.

4 Experiments

As we move into the experimental phase of our study, we first describe the environments selected for testing our algorithms, as well as the baselines used for comparison. This is followed by an in-depth analysis of the experimental results, demonstrating the performance of our algorithms in diverse scenarios. Through this, we aim to substantiate the theoretical aspects of our work with empirical evidence, highlighting the strengths and limitations of our approach.[1]

[1]Code: https://github.com/dimonenka/RL_policy_budget

Environments To evaluate our algorithms, we employ two distinct types of environments, each serving a specific purpose in our study.

Our initial objective is to scrutinize the behavior of our algorithms in a controlled, simplified setting. We use the Resource Gathering environment adapted from (Barrett and Narayanan 2008; Alegre et al. 2022), where a policy directs a character in a 5x5 grid world to collect resources. In our r-MDP modification, each of the $n = 25$ unique agents is assigned a specific resource tile. The goal is to collect these resources efficiently, with the episode ending when the character returns to the starting tile. The agents' rewards for collecting the respective resources, calculated as $r = 100 - T$, incentivize quick resource collection and require optimal pathfinding. In this scenario, we explore policy budgets ranging from $k \in \{1, 2, 3, 5, 10, 25\}$, examining how the number of policies affects efficiency and agent satisfaction.

To rigorously test our algorithms in more complex scenarios, we employ MuJoCo environments (Todorov, Erez, and Tassa 2012; Tassa et al. 2018; Tunyasuvunakool et al. 2020), including HalfCheetah, Ant, Hopper, and Walker2d. These tasks involve controlling robots with continuous actions in high-dimensional states. Each episode lasts for 1000 time steps or until the robot falls. To adapt these environments as r-MDPs, we define $n \in \{100, 1000\}$ agents, and for each agent, uniformly sample a target velocity $v^i \sim U[0, b]$, where b is selected as 2.5 for Walker2d and Hopper, 3 for Ant, and 4 for HalfCheetah. This is inspired by the meta-RL literature, where each sampled velocity is treated as a different task (Finn, Abbeel, and Levine 2017). Each time step, the agents are rewarded for the proximity of the robot to their target velocities according to the reward function $r^i(s_t, a_t) = 1 - \min(1, 20 \cdot \left|v^i - v_{t+1}\right| / b)$. The rewards are normalized s.t. the cumulative reward over an episode equals 100 for an agent when its target speed is maintained perfectly. Note that the reward of a particular agent is non-zero in only a narrow velocity interval, which echoes the costly error scenarios depicted in our introductory examples. We experiment with policy budgets $k \in \{1, 2, 5, 10, 50\}$.

Both environments offer distinct challenges: the Resource Gathering environment tests the algorithms' effectiveness in a discrete, straightforward setting, while the MuJoCo tasks present a more complex and continuous challenge. Together, they comprehensively evaluate our algorithms' ability to handle diverse agent preferences and policy budget constraints, reflecting the scenarios discussed in our introduction.

Algorithms In our experiments, we utilize two algorithms developed in this study, referred to as the EM algorithm and the end-to-end algorithm, as detailed in Sections 3.3 and 3.4.

Finding suitable baselines for comparison proved challenging due to the unique constraints of the r-MDP framework, which are not addressed by most methods in related fields. However, we identified a relevant baseline in a clustering-based algorithm used in RL for personalization in healthcare (see Section 1.1). This algorithm typically pre-trains a universal policy for all agents, employs K-Means clustering on sampled trajectories to group agents, and then trains a policy for each cluster. To align this method with our r-MDP setting,

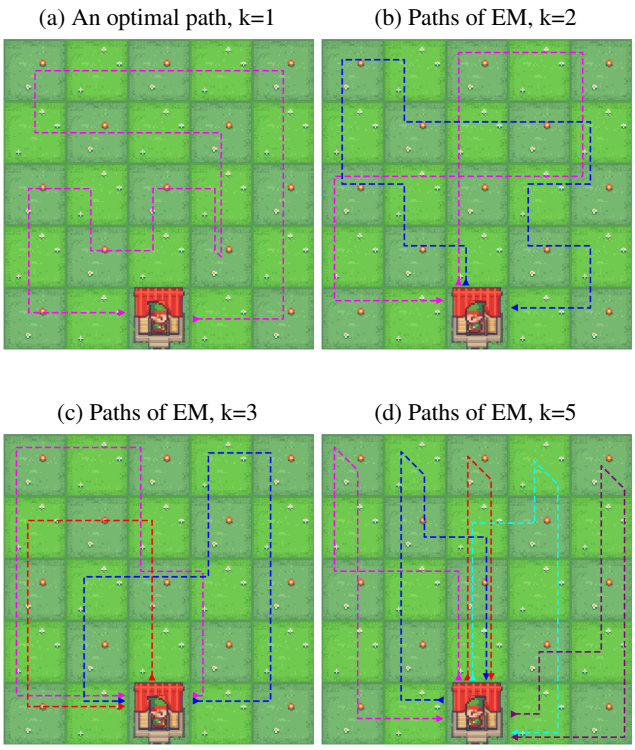

Figure 1: Paths that representatives learn in Resource Gathering after being trained with our EM algorithm for different k and $n = 25$ (0-th random seed). The representatives divide the map such that 1) each tile is visited by some policy and 2) policies jointly minimize the average episode length.

where sample efficiency is less of a concern, we extend both the pre-training of the universal policy and the training of cluster-specific policies to approximate convergence.

As a control, we also include a weak baseline where agents are randomly assigned to representatives, with policies subsequently trained without personalization considerations. This serves to benchmark the minimum expected performance and to emphasize the impact of personalized approaches.

To ensure reproducibility and transparency, hyperparameters and technical details are provided in the Appendix. All experiments were conducted ten times to ensure robustness, with standard errors reported alongside mean values.

4.1 Resource Gathering

The results, as depicted in Figure 2, showcase the effectiveness of our EM and end-to-end algorithms in comparison to the clustering baseline across different values of k. Notably, the performances $k = 1$ and $k = n$ are intentionally identical for all algorithms, as these scenarios either involve a single representative for all agents or individual representatives for each, eliminating the need for assignment learning.

For intermediate values of k, our algorithms perform similarly. Remarkably, while the optimal social welfare is 93.6 for $k = 25$, our algorithms attain social welfare above 90

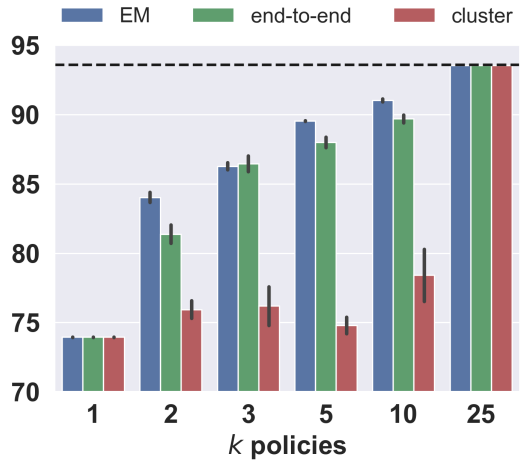

Figure 2: Performance of ours and baseline algorithms in Resource Gathering for different k and $n = 25$. The black dashed line represents the optimum for $k = 25$. For each k, all algorithms are trained for 1 million transitions per policy. For $k = 1$, all algorithms reduce to solving an MDP with a single policy. Confidence intervals represent standard errors.

with just $k = 10$ policies and above 85 with as few as $k = 3$ policies. This efficiency is illustrated through the representatives' paths in Figures 1b-1d, where they efficiently divide the map to cover smaller areas more rapidly, showcasing our approach's efficacy in achieving meaningful personalization under a strict policy budget.

In contrast, the clustering baseline exhibits minimal personalization, with its performance remaining largely unchanged regardless of k. This limitation is evident when analyzing the unanimous policy (Figure 1a), which traverses all tiles and thus yields identical rewards for all agents, providing no informative data for effective clustering.

These findings underscore the qualitative superiority of our algorithms over the clustering baseline. Our methods excel by learning assignments that directly optimize social welfare, in contrast to the baseline's reliance on a heuristic unaligned with the primary task. While diversifying the behaviors of the unanimous policy might enhance the baseline's performance, such an approach would still be heuristic and exceed the scope of existing literature, thus not qualifying as a conventional baseline.

4.2 MuJoCo Environments

In the MuJoCo environments, our algorithms consistently outperform the baselines across various policy budgets k, as shown in Figure 3. Both the EM and end-to-end algorithms demonstrate high levels of performance and significantly outperform random assignments in all tested scenarios. While the clustering baseline improves upon random assignments as well, it generally falls short of the performance achieved by our algorithms, with the exception of the Ant environment.

A deeper analysis of the assignments learned by the different algorithms (Figure 4) reveals intriguing patterns. Both our EM and end-to-end algorithms tend to group agents with sim-

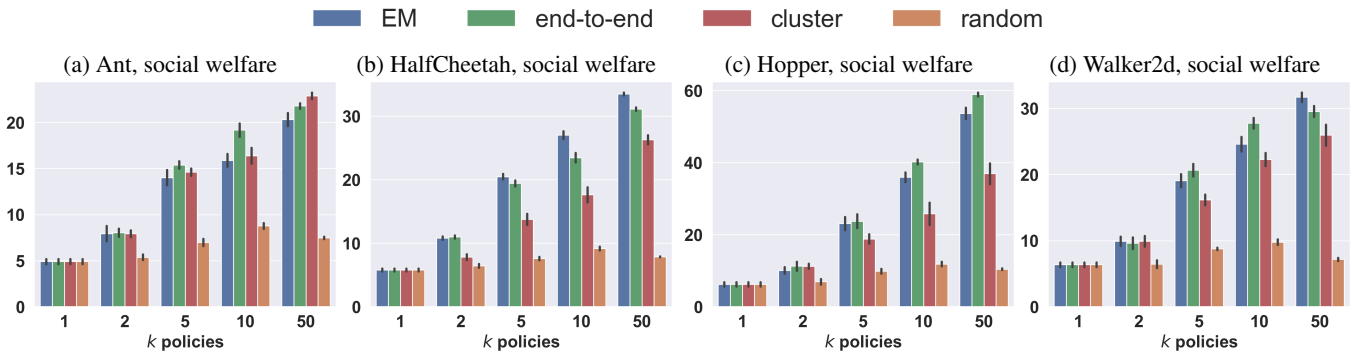

Figure 3: Performance of ours and baseline algorithms in MuJoCo environments. For each k, all algorithms are trained for 2 million transitions per policy. The number of agents is $n = 1000$ for $k = 50$ and $n = 100$ for smaller k. For $k = 1$, all algorithms reduce to solving an MDP with a single policy. Confidence intervals represent standard errors.

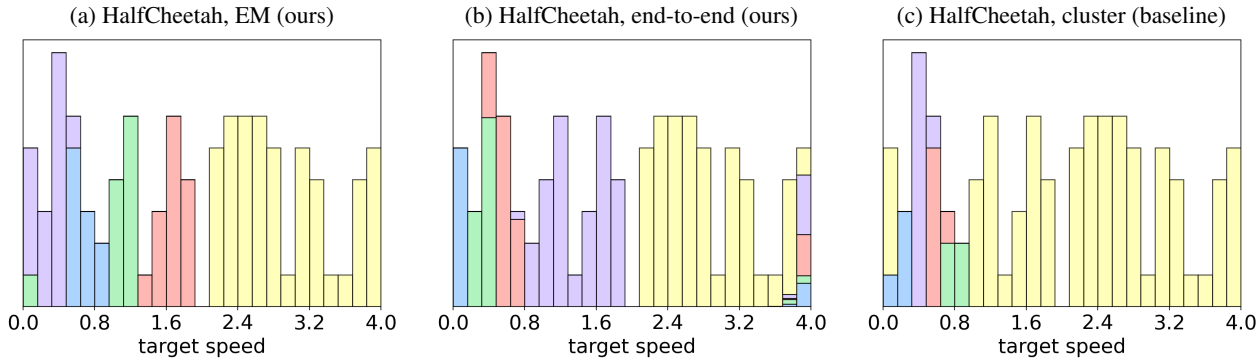

Figure 4: Histograms of agent assignments learned by ours and baseline algorithms for $n = 100$, $k = 5$ in HalfCheetah (0-th random seed). Each color denotes one of five representatives and bars of this color denote the target velocities of agents assigned to this representative. The expected behavior is a division of the agents' velocities into five intervals of similar sizes, one for each representative. Histograms for other environments are reported in the Appendix.

ilar target velocities and maintain relatively balanced group sizes. This suggests that they effectively identify cutoff points in the latent target velocity space for agent assignment. Notably, the end-to-end algorithm does not rigidly assign agents with the highest target velocities to any single representative, likely due to the absence of a strong enough learning signal from any representative for these high-velocity agents.

In contrast, the clustering baseline primarily segregates agents with low target velocities and lumps the majority into a single cluster. This pattern aligns with the baseline's heuristic nature, which focuses less on optimizing social welfare and more on simplistic clustering based on sampled trajectories.

5 Conclusion

In this study, we addressed the significant challenge of personalizing solutions in domains where regulatory assessments impose high costs of implementation. Based on the formalism of represented Markov Decision Processes (r-MDPs), we developed two deep reinforcement learning algorithms and theoretically validated their monotonic convergence to local optima. Empirically, our results underscored the efficacy of these algorithms in delivering meaningful personalization under policy budget constraints.

While our research represents a substantial stride forward, it also opens several avenues for further investigation. An important future direction is the refinement of social welfare functions to integrate fairness more comprehensively, ensuring that personalization does not come at the cost of equity. Additionally, exploring the incorporation of outside options that guarantee a minimal level of welfare for all individuals is crucial. Our current experiments, primarily centered on simulated tasks, set the stage for applying our methodology to real-world scenarios. Extending our approach to practical applications will be instrumental in verifying its effectiveness in diverse settings where personalization is key.

Overall, our work contributes to personalized reinforcement learning by addressing the dual challenges of regulatory compliance and maintaining a high level of personalization. We hope that our framework and algorithms will inspire future research in this domain and facilitate the practical deployment of personalized solutions in various complex and critical environments.

References

Alegre, L. N.; Felten, F.; Talbi, E.-G.; Danoy, G.; Nowé, A.; Bazzan, A. L. C.; and da Silva, B. C. 2022. MO-Gym: A Library of Multi-Objective Reinforcement Learning Environments. In *Proceedings of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn 2022*.

Ayer, T.; Alagoz, O.; and Stout, N. K. 2012. OR Forum—A POMDP approach to personalize mammography screening decisions. *Operations Research*, 60(5): 1019–1034.

Barrett, L.; and Narayanan, S. 2008. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, 41–47.

Baucum, M.; Khojandi, A.; Vasudevan, R.; and Davis, R. 2022. Adapting Reinforcement Learning Treatment Policies Using Limited Data to Personalize Critical Care. *INFORMS Journal on Data Science*, 1(1): 27–49.

Breton, M. D.; Kanapka, L. G.; Beck, R. W.; Ekhlaspour, L.; Forlenza, G. P.; Cengiz, E.; Schoelwer, M.; Ruedy, K. J.; Jost, E.; Carria, L.; et al. 2020. A randomized trial of closed-loop control in children with type 1 diabetes. *New England Journal of Medicine*, 383(9): 836–845.

Capponi, A.; Olafsson, S.; and Zariphopoulou, T. 2022. Personalized robo-advising: Enhancing investment through client interaction. *Management Science*, 68(4): 2485–2512.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22.

den Hengst, F.; Grua, E. M.; el Hassouni, A.; and Hoogendoorn, M. 2020. Reinforcement learning for personalization: A systematic literature review. *Data Science*, 3(2): 107–147.

Diana, E.; Dick, T.; Elzayn, H.; Kearns, M.; Roth, A.; Schutzman, Z.; Sharifi-Malvajerdi, S.; and Ziani, J. 2021. Algorithms and learning for fair portfolio design. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, 371–389.

el Hassouni, A.; Hoogendoorn, M.; Ciharova, M.; Kleiboer, A.; Amarti, K.; Muhonen, V.; Riper, H.; and Eiben, A. 2022. pH-RL: A personalization architecture to bring reinforcement learning to health practice. In *Machine Learning, Optimization, and Data Science: 7th International Conference, LOD 2021, Grasmere, UK, October 4–8, 2021, Revised Selected Papers, Part I*, 265–280. Springer.

El Hassouni, A.; Hoogendoorn, M.; Eiben, A. E.; Van Otterlo, M.; and Muhonen, V. 2019. End-to-end Personalization of Digital Health Interventions using Raw Sensor Data with Deep Reinforcement Learning. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 258–264.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Goindani, M.; and Neville, J. 2020. Cluster-based social reinforcement learning. *arXiv preprint arXiv:2003.00627*.

Grua, E. M.; and Hoogendoorn, M. 2018. Exploring clustering techniques for effective reinforcement learning based personalization for health and wellbeing. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 813–820. IEEE.

Hans, A.; Schneegaß, D.; Schäfer, A. M.; and Udluft, S. 2008. Safe exploration for reinforcement learning. In *ESANN*, 143–148. Citeseer.

Hassouni, A. e.; Hoogendoorn, M.; van Otterlo, M.; and Barbaro, E. 2018. Personalization of health interventions using cluster-based reinforcement learning. In *PRIMA 2018: Principles and Practice of Multi-Agent Systems: 21st International Conference, Tokyo, Japan, October 29-November 2, 2018, Proceedings 21*, 467–475. Springer.

Hayes, C. F.; Rădulescu, R.; Bargiacchi, E.; Källström, J.; Macfarlane, M.; Reymond, M.; Verstraeten, T.; Zintgraf, L. M.; Dazeley, R.; Heintz, F.; et al. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1): 1–59.

Huang, S.; Dossa, R. F. J.; Raffin, A.; Kanervisto, A.; and Wang, W. 2022. The 37 Implementation Details of Proximal Policy Optimization. In *ICLR Blog Track*.

JDRF. 2022. FDA Authorizes a Fourth Artificial Pancreas System. https://www.jdrf.org/blog/2022/01/28/fda-authorizes-a-fourth-artificial-pancreas-system/. Accessed: 2024-01-11.

Junges, S.; Jansen, N.; Dehnert, C.; Topcu, U.; and Katoen, J.-P. 2016. Safety-constrained reinforcement learning for MDPs. In *Tools and Algorithms for the Construction and Analysis of Systems: 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, 130–146. Springer.

Kar, D.; Kosan, M.; Mandal, D.; Medya, S.; Silva, A.; Dey, P.; and Sanyal, S. 2023. Feature-based Individual Fairness in k-clustering. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2772–2774.

Lee, J.; Sun, M.; and Lebanon, G. 2012. Prea: Personalized recommendation algorithms toolkit. *The Journal of Machine Learning Research*, 13(1): 2699–2703.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, 157–163. Elsevier.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2): 129–137.

MacQueen, J. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, 281–297.

Mandal, D.; and Gan, J. 2022. Socially fair reinforcement learning. *arXiv preprint arXiv:2208.12584*.

Martin, K. N.; and Arroyo, I. 2004. AgentX: Using reinforcement learning to improve the effectiveness of intelligent tutoring systems. In *Intelligent Tutoring Systems: 7th International Conference, ITS 2004, Maceió, Alagoas, Brazil, August 30-September 3, 2004. Proceedings 7*, 564–572. Springer.

Moldovan, T. M.; and Abbeel, P. 2012. Safe exploration in markov decision processes. *arXiv preprint arXiv:1205.4810*.

Nadiger, C.; Kumar, A.; and Abdelhak, S. 2019. Federated reinforcement learning for fast personalization. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 123–127. IEEE.

Perkins, T. J.; and Barto, A. G. 2002. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3(Dec): 803–832.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015a. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015b. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shepitsen, A.; Gemmell, J.; Mobasher, B.; and Burke, R. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, 259–266.

Tabatabaei, S. A.; Hoogendoorn, M.; and van Halteren, A. 2018. Narrowing reinforcement learning: Overcoming the cold start problem for personalized health interventions. In *PRIMA 2018: Principles and Practice of Multi-Agent Systems: 21st International Conference, Tokyo, Japan, October 29-November 2, 2018, Proceedings 21*, 312–327. Springer.

Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; Casas, D. d. L.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Tunyasuvunakool, S.; Muldal, A.; Doron, Y.; Liu, S.; Bohez, S.; Merel, J.; Erez, T.; Lillicrap, T.; Heess, N.; and Tassa, Y. 2020. dm_control: Software and tasks for continuous control. *Software Impacts*, 6: 100022.

Wu, C.; Wu, F.; Huang, Y.; and Xie, X. 2023. Personalized news recommendation: Methods and Challenges. *ACM Transactions on Information Systems*, 41(1): 1–50.

Yao, J.; Dou, Z.; Xu, J.; and Wen, J.-R. 2020. RLPer: A reinforcement learning model for personalized search. In *Proceedings of The Web Conference 2020*, 2298–2308.

Zhu, F.; Guo, J.; Xu, Z.; Liao, P.; Yang, L.; and Huang, J. 2018. Group-driven reinforcement learning for personalized mhealth intervention. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, 590–598. Springer.

Personalized Reinforcement Learning with a Budget of Policies Appendix

Dmitry Ivanov, Omer Ben-Porat

Technion, Israel
divanov@campus.technion.ac.il, omerbp@technion.ac.il

A Convergence of the EM-like Algorithm

We use notations from Section 2 of the main text.

The objective is to maximize utilitarian social welfare:

$$SW(\boldsymbol{\alpha}, \boldsymbol{\pi}) = \mathbb{E}_{\mathcal{T}_0} \sum_{i,j} \alpha^i(j) V^{ij}(s_0),$$

where $\boldsymbol{\alpha} = (\alpha^i)_{i \in N}$ and $\boldsymbol{\pi} = (\pi^j)_{j \in K}$.

Lemma 1. *A function V^j defined by $V^j(s) = \sum_{i \in N} \alpha^i(j) V^{ij}(s)$ is a value function.*

Proof. Let $s \in S$.

Apply the definition of $V^{ij}(s)$:

$$V^j(s) = \sum_{i \in N} \alpha^i(j) \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \tilde{r}_t^i \mid s_0 = s, \pi^j \right].$$

Rearrange the terms:

$$V^j(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \sum_{i \in N} \alpha^i(j) \tilde{r}_t^i \mid s_0 = s, \pi^j \right].$$

Substitute $\tilde{r}_t^j = \sum_{i \in N} \alpha^i(j) \tilde{r}_t^i$:

$$V^j(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \tilde{r}_t^j \mid s_0 = s, \pi^j \right].$$

Observe that $V^j(s)$ is a value function by definition. □

Define the *E-step* as updating the assignments to $\boldsymbol{\alpha}^*$ given the policies $\boldsymbol{\pi}$:

$$\alpha^{i*}(j^*) = \begin{cases} 1, & j^* = \arg\max_j \mathbb{E}_{\mathcal{T}_0} V^{ij}(s_0) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Note: ties are broken arbitrarily, e.g., lexicographically.

Define the *M-step* as updating the policies to $\boldsymbol{\pi}^*$ given the assignments $\boldsymbol{\alpha}$:

$$\forall \{j \mid \sum_i \alpha^i(j) > 0\} : \pi^{j*} = \arg\max_{\pi^j} \mathbb{E}_{\mathcal{T}_0} V^j(s_0) \quad (2)$$

Note: if some representative j is not assigned any agents after an E-step (i.e., $\sum_i \alpha^i(j) = 0$), we may assign a random agent to this representative prior to the M-step. After the M-step, the representative will implement the optimal policy for this agent. For the formal proof, this is not required.

By Lemma 1, we can use any RL algorithm that has convergence guarantees to perform the M-step for each j.

The EM-like meta-algorithm is defined in Algorithm A. A specific implementation is discussed in the main text.

[tb] **Input**: r-MDP $\mathcal{M}_r$ [1] Arbitrarily initialize $(\alpha^i)_{i \in N}$ s.t. $\forall j : \exists i, \alpha^i(j) > 0$ assignments or policies change Perform M-step to update policies Perform E-step to update assignments

Theorem 1. *Given an r-MDP $\mathcal{M}_r$, the EM-like meta-algorithm converges to a local maximum of $SW(\boldsymbol{\alpha}, \boldsymbol{\pi})$.*

Proof. Observe that an E-step may not decrease social welfare:

$$SW(\boldsymbol{\alpha}^*, \boldsymbol{\pi}) - SW(\boldsymbol{\alpha}, \boldsymbol{\pi}) = \mathbb{E}_{\mathcal{T}_0} \sum_i \left[\max_j V^{ij}(s_0) - \sum_j \alpha^i(j) V^{ij}(s_0) \right] \geq 0.$$

Likewise, observe that an M-step may not decrease social welfare:

$$SW(\boldsymbol{\alpha}, \boldsymbol{\pi}^*) - SW(\boldsymbol{\alpha}, \boldsymbol{\pi}) = \mathbb{E}_{\mathcal{T}_0} \sum_{i,j} \left[\alpha^i(j) \left(V^i(s_0 \mid \pi^{j*}) - V^i(s_0 \mid \pi^j) \right) \right] \geq 0.$$

Therefore, the iterative application of E-step and M-step monotonically increases social welfare until convergence. Because the number of possible assignments is finite, the convergence is guaranteed. □

B Hyperparameters and Technical Details

PPO The PPO algorithm updates the policy to minimize the following loss function:

$$L(\theta) = -\sum_{t \in B} \min[\rho_\theta(s_t, a_t)\tilde{A}(s_t, a_t), \\ \min(\max(\rho_\theta(s_t, a_t), 1-\epsilon), 1+\epsilon)\tilde{A}(s_t, a_t)] \tag{3}$$

Our implementation of PPO is based on the PyTorch package (Paszke et al. 2019) for Python 3. We followed the procedure of (Huang et al. 2022) to replicate the performance from the original paper (Schulman et al. 2017). Specifically, we implemented:

- Orthogonal weight initialization;
- Generalized advantage estimation (Schulman et al. 2015);
- Normalization of advantages over the batch (per policy);
- Entropy bonus to encourage exploration;
- Gradient norm clipping;
- Continuous actions via normal distributions plus reparameterization trick;
- State-independent log standard deviations as learnable parameters;
- Independent action components;
- Action clipping;
- Normalization and clipping of observations.

Actors and critics were both trained with ADAM optimizers (Kingma and Ba 2014). Furthermore, we used hyperparameters standard for MuJoCo environments:

- Learning rate initialized at 0.0003 and annealed throughout the training to 0.0001;
- Entropy loss coefficient of 0.001;
- GAE $\lambda = 0.95$;
- One hidden layer with 64 neurons;
- Batch of 2048 transitions, divided into mini-batches of 64 transitions;
- A batch is used for training for 10 epochs;
- PPO clipping parameter $\epsilon = 0.2$;
- ADAM $\epsilon = 10^{-5}$.

Our algorithms For our EM-like algorithm, we used a mixing coefficient $\lambda = 0.05$. For our end-to-end algorithm, we updated ψ with ADAM optimizer with a learning rate of 0.002 in the same backward passes as the policies θ_i.

C Additional Histograms

Figure 5 reports histograms of assignments learned by ours and baseline algorithms. These echo the conclusions in the main text: our algorithms divide the latent velocity space better than the baseline, and thus provide more personalization.

References

Huang, S.; Dossa, R. F. J.; Raffin, A.; Kanervisto, A.; and Wang, W. 2022. The 37 Implementation Details of Proximal Policy Optimization. In *ICLR Blog Track*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

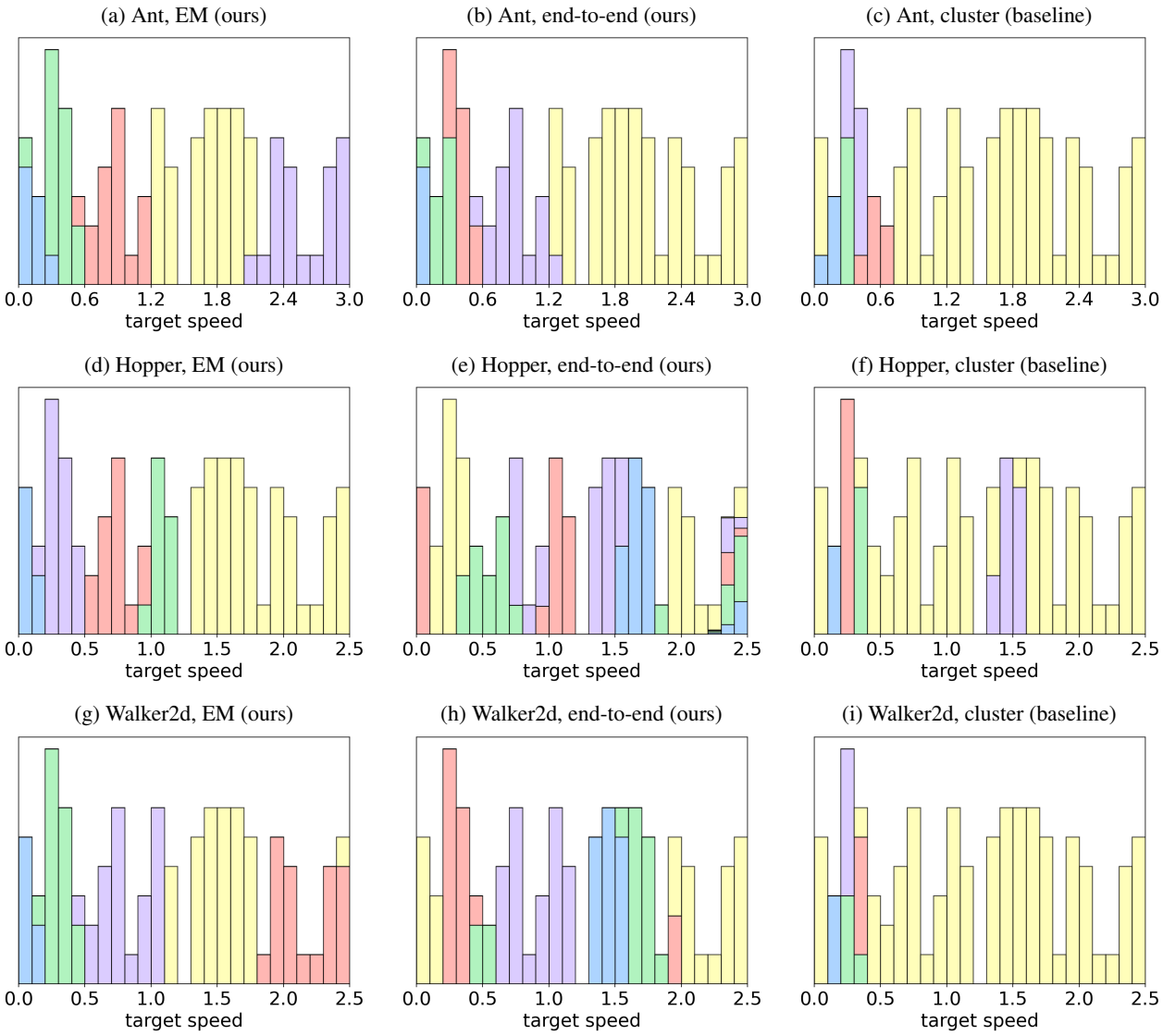

Figure 5: Histograms of agent assignments learned by ours and baseline algorithms for $n = 100$, $k = 5$ in Ant, Hopper, and Walker2d (0-th random seed). Each color denotes one of five representatives and bars of this color denote the target velocities of agents assigned to this representative. The expected behavior is a division of the agents' velocities into five intervals of similar sizes, one for each representative.