Intervention Aware Shared Autonomy

Weihao Tan [*1] **David Koleczek** [*1 2] **Siddhant Pradhan** [*1] **Nicholas Perello** [1] **Vivek Chettiar** [3] **Nan Ma** [3] **Aaslesha Rajaram** [3] **Vishal Rohra** [3] **Soundar Srinivasan** [3] **H M Sajjad Hossain** [† 3] **Yash Chandak** [† 1]

Abstract

Shared autonomy refers to approaches for enabling an autonomous agent to collaborate with a human with the aim of improving human performance. However, besides improving performance, it may often be beneficial that the agent concurrently accounts for preserving the user's experience or satisfaction of collaboration. In order to address this additional goal, we examine approaches for improving the user experience by constraining the number of interventions by the autonomous agent. We propose two model-free reinforcement learning methods that can account for both hard and soft constraints on the number of interventions. We show that not only does our method outperform the existing baseline, but also eliminates the need to manually tune an arbitrary hyperparameter for controlling the level of assistance. We also provide an in-depth analysis of intervention scenarios in order to further illuminate system understanding.

1. Introduction

Human-AI collaboration forms an integral part of advancing science in domains where neither advances in independent AI nor human experts are able to fully realize the solution to a problem. Synergistic systems for control between humans and autonomous systems thus span a vast spectrum, ranging from safety critical systems such as aircraft operations (Matni & Oishi, 2008), semi-autonomous driving (de Winter & Dodou, 2011), and UAV control (Backman et al., 2021), to robotics tasks for assistive eating (Jeon et al., 2020), drinking (Schröer et al., 2015), wheelchair control (Erdogan & Argall, 2017; Trieu et al., 2008), and other teleoperation tasks (Kofman et al., 2005; Aarno et al., 2005). Human-AI collaboration has also shown promising advances in microsurgery (Kragic et al., 2005), brain-computer interfaces (Muelling et al., 2017; Shanechi et al., 2016; Kim et al., 2006), myoelectric devices (Pilarski et al., 2011), and in leisure applications (e.g., enabling people with disabilities to enjoy playing Xbox video games (Xbox, 2018)).

Shared autonomy provides a framework to allow human and autonomous agent interact congruently to improve performance at the task, while still allowing the human to maintain control (Abbink et al., 2018). Several prior works on shared autonomy rely on specific knowledge of the environment's dynamics, the ability to model them accurately (Gopinath et al., 2017; Javdani et al., 2018; Backman et al., 2021), or having data a priori in order to determine the goals of the collaborating human (Reddy et al., 2020; 2018b; Carroll et al., 2019; Javdani et al., 2016). However, for many tasks the computational dynamics model may be unavailable, or determining user goals accurately in advance of system design may not be feasible. Therefore, similar to the works by Reddy et al. (2018a), Schaff & Walter (2020), and Du et al. (2020) we focus on a model-free approach and do not aim to model user's goals.

An important aspect not addressed by the aforementioned prior works is how to allow humans to use the full depth of their expertise towards achieving their goals. In other words, the agent should assist the human minimally or intervene only when necessary, while also ensuring that the human achieves optimal performance. This consideration is especially important in domains such as assistive robotics (Javdani et al., 2018; Gopinath et al., 2017; Erdogan & Argall, 2017) where patients may want to feel as independent as possible while performing the task at hand (Verdonck et al., 2011; Palmer et al., 2005), or in AI assisted employee training (Seo et al., 2021) where the AI agent is expected to gradually reduce assistance to ensure the user is learning the objective. Our work focuses on addressing this gap by incentivizing the agent to minimize interventions, while making sure that the human-agent collaboration achieves near-optimal performance.

We introduce two new, model-free reinforcement learning methods: one based on soft-constraints and the other based on hard-constraints on the number of interventions permit-

[*]Equal contribution [†]Equal Advising [1]Department of Computer Science, University of Massachusetts Amherst [2]MassMutual Data Science, Amherst, MA [3]Microsoft. Correspondence to: Weihao Tan <weihaotan@umass.edu>.

HumanAI workshop at the 38th International Conference on Machine Learning, 2021. Copyright 2021 by the author(s).

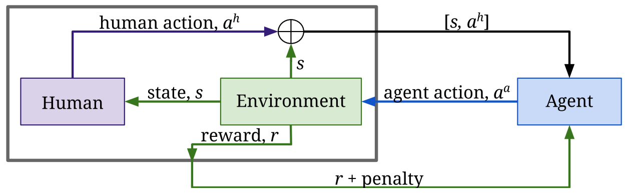

Figure 1. Overview of the proposed human in the loop reinforcement learning framework.

ted by the agent. In the cases where the user is flexible in the number of interventions acceptable, the soft constraint method provides an approach to consider an intervention penalty along with the reward function, and solves it using the dual formulation (Puterman, 2014). On the other hand, if a user desires strict upper-bounds on the acceptable number of interventions, the hard constraint method provides a procedure to include the intervention constraint directly in the state representation ensuring that the constraint is *never* violated.

To benchmark our algorithms, we conduct experiments using simulated human agents in the Lunar Lander (Brockman et al., 2016) environment. We show that our method outperforms the previous baseline (Reddy et al., 2018a) in terms of intervention rate and environment returns. Finally, we also provide analysis of the states where the agent takes control as an additional means of validation of the proposed methods.

2. Related Work

Shared Autonomy. In shared autonomy, the control of a system is shared by human and agent to accomplish a common goal. Earlier works assumed that the agent is aware of the common goal (Crandall & Goodrich, 2002; Kofman et al., 2005; You & Hauser, 2012). However, this is a strong assumption, and as a result recent works have proposed agents that can infer the user's intent and predict the user's goal from the user's action and environment dynamics. They do so by formulating this problem as a partially observable Markov decision process (POMDP). Bayesian inference (Javdani et al., 2018; Muelling et al., 2017; Sadigh et al., 2016; 2018), inverse reinforcement learning (Ng et al., 2000; Ratliff et al., 2006; Levine & Koltun, 2012) and hindsight optimization (Javdani et al., 2015) are the most common approaches to predicting the user's goal in this formulation. These approaches are based on an assumption that the environment dynamics, the goal space, and the user's policy are known to the agent, which is often violated in practice. Reddy et al. (2018a) proposed a model-free shared autonomy framework to relax these assumptions and formulate the problem as a Markov decision process (MDP). This method does not require prior knowledge of environment dynamics, the human's policy, the goal space, and goal representation. Instead, it uses human-in-the-loop reinforcement learning with functional approximation to formulate a policy for the agent learned only from environmental observation and user input.

Intervention optimization. In the setting of user assistance, there are only a few works that emphasise the influence of the assistive agent on the user. An agent that intervenes too often will reduce a user's advocacy and ultimately affect their experience in the system.To tackle this challenge, Reddy et al. (2018a) employed deep Q-learning to train the assistive agent. Instead of taking the action with the highest q-value, they first sort the actions in terms of similarity to the user provided action, in descending order, using an action-similarity function. A final action is then selected as the one that is closest to the user's action while ensuring its q-value is not significantly worse than the optimal action. This action selection addresses the intervention problem implicitly. By tuning the tolerance of the system to suboptimal human suggestions, which is a hyperparameter, this method can provide different levels of assistance. However, designing such an action-similarity function makes this approach an environment-dependent problem. Broad et al. (2019) proposed to minimize the interventions in a similar way. They accept the user's action only when it is close enough to the optimal action as determined by an optimal controller. However, developing such an optimal controller may require complete knowledge of the environment dynamics and the user's goal. Schaff & Walter (2020) formulate the problem as a constrained MDP where they consider minimizing the amount of interventions an agent makes, subject to the returns of the agent's learned policy being greater than a given constant. In contrast, our work describes a method to maximize returns of the human-AI collaboration, subject to the number of interventions taken being less than a given constant.

3. Method

We first introduce the problem setup based on the shared autonomy framework (Reddy et al., 2018a). We then present our proposed methods that enable the agent to optimize the frequency of interventions, while maximizing the collaborative performance. Our method is model-free and does *not* assume the knowledge of the environment dynamics, the goal space, or, the user's policy.

Shared autonomy can be formulated as a MDP, given by a tuple $\{\mathcal{S}, \mathcal{A}_h, \mathcal{A}_a, \mathcal{T}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ is the set of environment states and $\mathcal{A}_h$ is the set of actions that can be taken by the human. The set of actions available to the agent is denoted using $\mathcal{A}_a$. Dynamics, the reward function, and discount factor are represented using $\mathcal{T}$, $\mathcal{R}$ and γ, respectively. Before choosing an action according to its policy, the agent observes both the environment state $s \in \mathcal{S}$ and the user's action $a^h \in \mathcal{A}_h$. Therefore, the effective set of states for the agent is $\overline{S} = \mathcal{S} \times \mathcal{A}_h$, and, we define an instance of this set as $\overline{s} := [s, a^h]$, a simple concatenation. The MDP that our agent acts in then can be formalized as $\mathcal{M} = \{\overline{\mathcal{S}}, \mathcal{A}_a, \overline{\mathcal{T}}, \mathcal{R}, \gamma\}$, where $\overline{\mathcal{T}}(\overline{s}, a^a, \overline{s}') = \mathcal{T}(s, a^a, s')\pi_h(s', a^{h'})$, where $a^a \in \mathcal{A}_a$, and, π_h is the policy of the human user. The goal of the agent is to find the optimal policy π_a^* that maximizes the expected discounted sum of rewards: $\pi_a^* = \arg\max_{\pi_a} \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(\overline{s}_t, a_t) \right]$. We provide an overview of the proposed agent-human-environment interaction in Figure 1.

3.1. Hard Constrained Shared Autonomy

A simple way to control the level of assistance is to limit the discrete amount of instances where the assistive agent can intervene, or override user's input in an episode. Thus, we propose to set a *budget*, which is the maximum number of times an agent can intervene in an episode, as a hard constraint to limit the behaviour of agent. When the budget is greater than 0, the agent can take any corrective action. However once the budget is exhausted, the agent is only able to accept the user's action. Furthermore, if the agent still attempts to intervene, we add a penalty to the reward in order to reinforce the need for the agent to assist optimally while there is a remaining budget.

We introduce budget as part of the environment observed by the agent. The hard constrained shared autonomy MDP can be formalized as $\mathcal{M}_b = \{\overline{\mathcal{S}}_b, \mathcal{A}_a, \mathcal{T}_b, \mathcal{R}_b, \gamma\}$, where $s^b \in \overline{\mathcal{S}}_b := [s, a^h, b]$, $\mathcal{T}_b(s^b, a^a, s^{b'}) = \mathcal{T}(s, a^a, s')\pi_h(s', a^{h'})\mathbb{P}(b'|b, a^a, a^h)$. The reward function $\mathcal{R}_b$ is defined in the following equation:

$$\mathcal{R}_b(s^b, a^a, s^{b'}) = R(s, a^a, s') - \begin{cases} \lambda & b = 0 \text{ and } a^a \neq a^h \\ 0 & \text{else} \end{cases}$$

Here, the hyperparameter $\lambda \geq 0$ refers to the penalty for intervening after the budget is exhausted. The goal of the agent is still to find the optimal policy π_a^* that maximizes the expected discounted sum of rewards: $\pi_a^* = \arg\max_{\pi_a} \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t R(s_t^b, a_t) \right]$. We refer to this method as the *budget method* in the rest of the paper.

3.2. Soft Constrained Shared Autonomy

In hard constrained shared autonomy, we gave the agent an intuitive constraint of a maximum number of interventions. This introduces the problem of balancing utility of the agent and the preservation of user's experience manually using the budget. The optimal budget can be difficult to estimate depending on the environment. Hence, we now relax this hard constraint, and, instead penalize the agent for every intervention it takes, compared to the previous method where we start penalizing the agent only after the exhaustion of the budget. In certain use cases, it may be beneficial to not have a hard cap on the number of interventions. The magnitude of the penalty establishes a trade-off between the increased reward accumulated by intervening over the human action, and the penalty associated with doing so. As with hard constrained shared autonomy, we introduce a modified reward function:

$$\overline{\mathcal{R}}(\overline{s}, a^a, \overline{s}') = \mathcal{R}(s, a, s') + \mathcal{R}_{\texttt{penalty}}(\overline{s}, a^a), \tag{1}$$

$$\text{where} \qquad \mathcal{R}_{\texttt{penalty}}(\overline{s}, a^a) = \begin{cases} 0 & a^h = a^a \\ -\lambda & a^h \neq a^a \end{cases}$$

where $\lambda \geq 0$ and $\mathcal{R}_{\texttt{penalty}}$ penalizes the agent for intervening. Using $\overline{\mathcal{R}}$, shared autonomy can be recast into the standard reinforcement learning setup. The MDP of the penalty method can be formalized as $\mathcal{M} = \{\overline{\mathcal{S}}, \mathcal{A}_a, \overline{\mathcal{T}}, \overline{\mathcal{R}}, \gamma\}$. The goal of the agent is to find the optimal policy π_a^* that maximizes the expected discounted sum of rewards with the modified reward function: $\pi_a^* = \arg\max_{\pi_a} \mathbb{E}_{\pi_a} \left[\sum_{t=0}^{T} \gamma^t \overline{\mathcal{R}}(\overline{s}_t, a_t) \right]$.

Intuitively, λ is a hyperparameter that encodes the trade-off between performance and the user's willingness to receive assistance from the agent. A large λ discourages the agent from intervening, but limits the agent's capability as a consequence. In contrast, a small λ encourages the agent to intervene more which might enable greater performance, but at the cost of potentially interrupting user's experience more often. However, choosing an optimal value for λ is not intuitive and also varies significantly according to the environment and the scale of rewards. Instead of leaving the user to fine-tune this sensitive hyperparameter, we propose a method to automatically optimize λ using a dual formulation (Haarnoja et al., 2018).

Our objective is to find a optimal policy that maximizes expected return while satisfying a minimum expected intervention constraint. Note that we do not discount rewards as it would result in discounting the intervention penalty as well. We cannot justify this approach in our use case as it is unreasonable to allow for more interventions as an episode grows longer. Formally, we want to solve the following constrained optimization problem

$$\pi_a^* = \max_{\pi_a} \mathbb{E}_{\pi_a, \pi_h} \left[\sum_{t=0}^{T} r(s_t, a_t) \right] \text{s.t. } \mathbb{E}_{\pi_a, \pi_h} \left[\sum_{t=0}^{T} I(a_t^a, a_t^h) \right] \leq c \tag{2}$$

where we define $I(a^a, a^h) = \begin{cases} 1 & a^a \neq a^h \\ 0 & a^a = a^h \end{cases}$ as an indicator variable for the occurrence of an intervention. Written as a Lagrangian, we obtain the following constrained objective:

$$\max_{\pi_a} \min_{\lambda \geq 0} \mathbb{E}_{\pi_a, \pi_h} \left[\sum_{t=0}^{T} r(s_t, a_t) + \lambda \left(c - \sum_{t=0}^{T} I(a_t^a, a_t^h) \right) \right] \tag{3}$$

According to the preceding equation, we would require an entire episode to be able to evaluate the amount of interventions that occur based on our constraint c. If we wish to optimize at every timestep, we can rewrite the constraint to be a linear function of time: $c' = \frac{c}{t}$. In the case of interventions, this can be thought of as the *intervention rate*.

$$\max_{\pi_a} \min_{\lambda \geq 0} \mathbb{E}_{\pi_a, \pi_h} \left[\sum_{t=0}^{T} \left(r(s_t, a_t) + \lambda \left(c' - I(a_t^a, a_t^h) \right) \right) \right] \quad (4)$$

We perform the maximization of π using a model-free RL algorithm such as DDQN (Hasselt et al., 2016) or SAC (Haarnoja et al., 2018), while we optimize λ using stochastic gradient descent where the update for each time-step is written as follows. Note that the reward term does not depend on λ.

$$\lambda \leftarrow \lambda - \alpha \nabla_\lambda \left(r(s_t, a_t) + \lambda \left(c' - I(a_t^a, a_t^h) \right) \right) \quad (5)$$

$$\lambda \leftarrow \lambda - \alpha (c' - I(a_t^a, a_t^h)) \quad (6)$$

We refer to this method as the *penalty adapting method*. At each timestep we update the policy, and then additionally perform a single gradient step on the dual variables. This does not give us an exact solution to the dual problem which is impractical to solve in RL. However, we find that this approach works in practice to automatically tune λ according to the constraint on interventions. Automatically tuning the penalty can be considered optional if the user desires to set it manually and we provide experimental results with and without it. We refer to the method without penalty adapting as simply the *penalty method*.

4. Experiments

In this section we evaluate the performance of our methods on the popular Lunar Lander environment using different simulated users. The goal of our experiments is to test the central hypothesis that the proposed methods can provide varying levels of assistance to improve the user's performance.

Lunar Lander: A 2D simulation game from OpenAI Gym (Brockman et al., 2016). The goal of the game is to control the engines of a lunar lander spacecraft to land at a designated landing zone without crashing the spacecraft. Each episode lasts at most 1000 steps. It provides a choice between discrete and continuous action spaces for the pilot to control. For the two-dimensional discrete action space, it consists of six discrete actions that are combinations of the main engine {on, off} and the lateral engines {left, right, off}. For the continuous action space version, the pilot can fire each engine at any power level. The state space is 8 dimensional: position, velocity, angular position, angular velocity, and whether or not each leg is in contact with the ground. The reward function penalizes speed, tilt, fuel usage, and crashing while rewarding landing at the target location.

	Lunar Lander
Tolerance	$[0, 0.1, 0.2, 0.3, 0.4, ..., 1]$
Budget	$[0, 25, 50, 75, .., , 500, 1000]$
Penalty (10^{-2})	$[0, 1, 2, 5, 10, 20, 50, ..5000]$
Intervention Rate	$[0, 0.1, 0.2, 0.3, 0.4, ..., 1]$

Table 1. The hyperparameters that the methods use to train on Lunar Lander. Tolerance for the baseline method, budget for the budget method, penalty for the penalty method and intervention rate for the penalty adapting method.

As our work is closely related to (Reddy et al., 2018a), we use their methodology as our baseline to compare the performances of our proposed methods and adopt their terminology of referring to the human operator as the *pilot* and the assistive agent as the *copilot*. Much like our method, (Reddy et al., 2018a) can also provide different levels of assistance by tuning a hyperparameter, α, which is the tolerance of the system to suboptimal pilot actions. If q-values of the user's action is less than the threshold of the tolerance, $\alpha q*$, the copilot will intervene. Comparing the q-values of different actions limits the baseline method to value-based methods which consequently makes it difficult to apply to continuous action spaces.

4.1. Setup

Similar to (Reddy et al., 2018a), we design four agents applying different strategies to simulate real human players in the Lunar Lander environment to evaluate our methods: **Noop Pilot:** does not apply any action (i.e. always executes a noop action), **Laggy Pilot:** the optimal agent with 80% possibility to repeat the previously executed action, **Noisy Pilot:** the optimal agent with 25% possibility to take a random action), and **Sensor Pilot:** simply tries to move toward the landing site by firing its left or right engine and does not use the main engine.

For each simulated pilot, we train the copilot using the baseline method (Reddy et al., 2018a) and our proposed methods: the penalty, budget, and penalty adapting methods. In order to make our method more comparable with the baseline, we use DDQN (Hasselt et al., 2016) with experience replay (Lin, 1993) to train all the agents on Lunar Lander with discrete action space. Our Q-network is a multi-layer perceptron with two layers of 64 neurons in each layer. We use the same method to train the optimal agent for the simulated pilot.

For each method, we train copilots with their respective hyperparameters (tolerance, budget, penalty, and intervention rate) which determine different levels of assistance. A detailed listing of the hyperparameters used is located in Table 1. We train 10 copilots with different random seeds for each parameter setting and test each over 100 episodes. Finally,

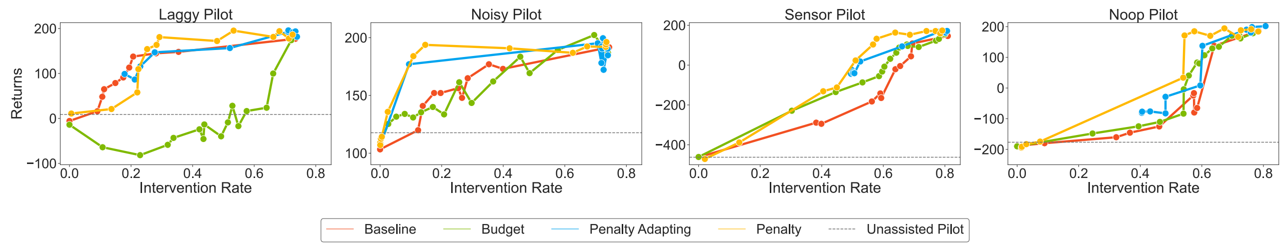

Figure 2. The performance of different simulated pilots assisted by the copilots trained by (Reddy et al., 2018a) and our methods on Lunar Lander. The dashed line represents the average performance of a simulated pilot without a copilot over 100 episodes. Other lines represent the average performance of the corresponding agent for each of our methods.

we take the average performance of these 10 copilots for the final results.

4.2. Evaluation

We evaluate the performance of our methods with respect to increasing intervention rate. Figure 2 shows the trend of total return at different intervention rates for individual methods on Lunar Lander with discrete action space, respectively. It is evident that our methods greatly improve the performance of most pilots. Intuitively, as tolerance threshold α and penalty λ decrease, or budget b and intervention rate c' increase, the copilot should intervene more frequently and accumulate better reward. Although the hyperparameters are not shown in the figures, the appearance of the data points are consistent with the value of the parameters. For example, data points with lower intervention rate use a lower budget or use a higher penalty. So by tuning the hyperparameters, the copilot can assist the pilot with a different frequency of interventions, which means that the both of the baseline method and our methods can provide different levels of assistance to the user.

Next, we discuss the performance of different methods. Since we aim to provide different levels of assistance to the user, the most practical way for comparison is setting a target return, which is the user's expectation, and assess which method achieves the target return with a lower intervention rate. Alternatively, from the perspective of intervention rate, we can also form this comparison by setting a target intervention rate, which is the level of assistance that the user needs, and finding which method can achieve higher return.

Budget Method: Compared to other methods, the budget method distributes more evenly across the intervention rate. The intervention rate is always less than or equal to the ratio of the budget and the total time steps. So if we keep increasing the budget at a constant rate, the intervention rate will not change dramatically. We note that this method performs extremely poorly when assisting the laggy pilot. This is due to the laggy pilot always taking significantly more time steps to play the game compared to other simulated pilots. As a result a larger budget is required to achieve better return. For the other three simulated pilots, the budget method shows almost similar performance to baseline method and worse than penalty and penalty adapting method.

Penalty Method: This method always achieves the optimal reward, which is about 200, with lowest intervention rate among the other methods. But compared to the other methods, the hyperparameter for this method is more environment-dependent. We need to have a full understanding of the reward function to chose an appropriate penalty.

Penalty Adapting Method: This method tends to demonstrate high intervention rate and high reward regardless of the hyperparameter configuration. Even when the copilot is trained with low intervention rate, such as 0, 0.1 and 0.2, it still accrues significantly higher reward than the single pilot. Though the intervention rate serves as a constraint to limit the behaviour of the copilot, it also guides the copilot when to intervene. It encourages the copilot to intervene when the current intervention rate is less than the constraint. This is the key difference from the budget method, which does not reward intervention. And when the benefit of intervening is greater than the penalty, it will still choose to intervene regardless of the constraint, which is why we refer to this constraint as a soft constraint. Thus, the penalty adapting method has comparatively high lower bound, even without fine-tuning the intervention rate for the constraint. On the other hand, it exposes the limitation of this method that it cannot handle the case where the user wants very few interventions. Overall, it still has comparable performance with the baseline method.

4.3. Analysis of Interventions

In having reviewed performances of individual proposed methods, we discussed their efficiency i.e. how intervention rate and target reward effect their ability to assist. We must also assess whether the agents are meaningfully assisting and intervening where necessary. In this contrast, we focus on the states in which our individual methods are deciding to intervene in.

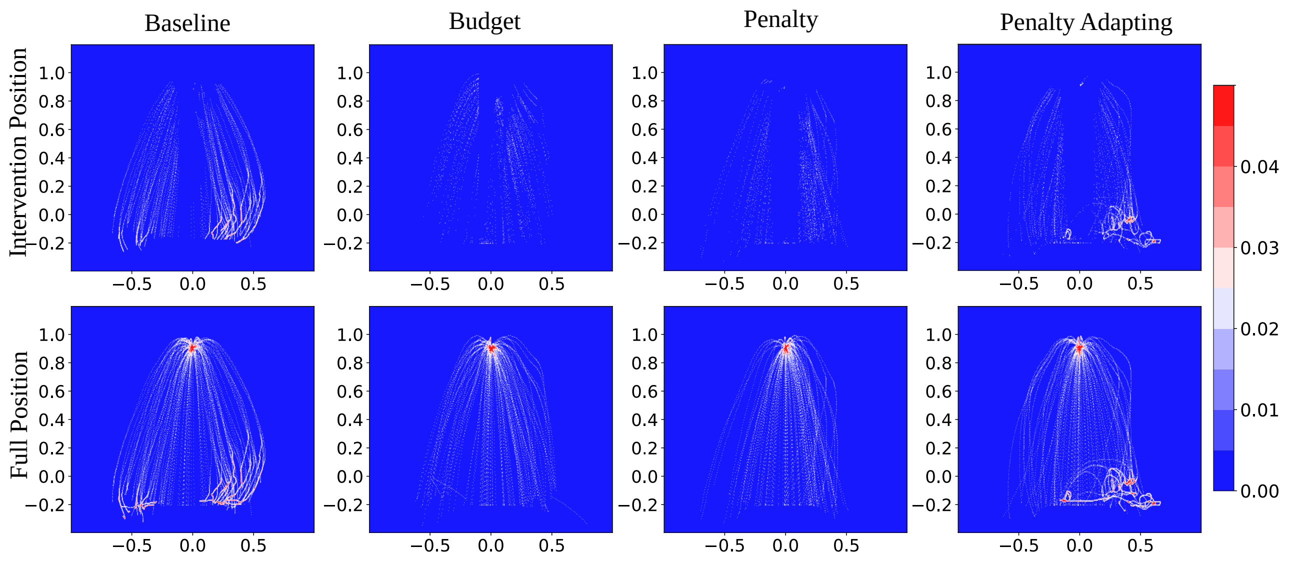

Figure 3. Heat maps which visualize the most frequently visited parts of the state space and where the interventions happen most frequently. The plots in the first row show only the positions where interventions occurred. The second row shows the full trajectories of the pilot. The intensity (legend) represents the percentage of time spent in that location and is normalized across all trials. Sensor pilot is used as the simulated agent. The parameter we use to train the copilots are: $\alpha = 0.7$ for the baseline, penalty = 5 for the penalty method, budget = 50 for the budget method and intervention rate = 0.7 for the penalty adapting method. Each method is run for 100 episodes.

Figure 3 uses heat maps to demonstrate at which (x, y) positions in the Lunar Lander environment interventions happen most frequently. For these experiments, we fixed the landing zone, or the goal location, to be at the origin of the frame. Additionally, we executed these experiments using the sensor pilot as its behavior is the most intuitive to visualize. When the x (horizontal) position of the agent is less than -0.1, the lander is to the left of the goal and the pilot fires its left engine to move right towards the goal. It does the opposite when the x position is greater than 0.1, and the pilot does nothing when it is between -0.1 and 0.1. Both the penalty and penalty adapting methods perfectly capture the strategy of the sensor pilot. Almost all the interventions happen on both sides of the landing zone when the pilot is firing its left or right engine. Few interventions happen in the area when the pilot does not fire any engine (i.e. it lands via gravity) and only when the lander is close to the ground are interventions made to maintain balance. The baseline method almost captures the strategy, but still makes unnecessary interventions in the area where the agent does nothing.

5. Conclusion

We proposed a framework for training human-in-the-loop reinforcement learning agents with a focus on being able to control the amount of interventions of the autonomous agent. Our soft-constrained method allows a user to train an agent "out of the box" with minimal hyperparameter tuning. While our hard-constrained method provides a means to train under an environment where after it violates any constraint, it is not able to continue taking actions in the environment. We analyzed these methods on the Lunar Lander game environments and showed that it outperformed a previous baseline. We also analyzed where interventions occur with respect to the state space. We believe that shared autonomy with model-free RL is a promising direction to solving this problem because it is able to leverage the wealth of existing RL methods directly, while also giving flexibility to incorporate many other constraints outside of interventions. There is a large space of possible domains and different user experiences to consider. We could potentially try to leverage using real human data from video games and work on implementing our method on a real game. We also wish to test if our method is adaptable to a different distribution of human behavior than at training time and determine if means of goal inference are necessary.

References

Aarno, D., Ekvall, S., and Kragic, D. Adaptive virtual fixtures for machine-assisted teleoperation tasks. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 1139–1144. IEEE, 2005.

Abbink, D. A., Carlson, T., Mulder, M., de Winter, J. C. F., Aminravan, F., Gibo, T. L., and Boer, E. R. A topology of shared control systems—finding common ground in diversity. *IEEE Transactions on Human-Machine Systems*, 48 (5):509–525, 2018. doi: 10.1109/THMS.2018.2791570.

Backman, K., Kulić, D., and Chung, H. Learning to assist drone landings. *IEEE Robotics and Automation Letters*, 6 (2):3192–3199, 2021. doi: 10.1109/LRA.2021.3062572.

Broad, A., Murphey, T., and Argall, B. Highly parallelized data-driven mpc for minimal intervention shared control. *arXiv preprint arXiv:1906.02318*, 2019.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Carroll, M., Shah, R., Ho, M. K., Griffiths, T., Seshia, S., Abbeel, P., and Dragan, A. On the utility of learning about humans for human-ai coordination. In *Advances in Neural Information Processing Systems*, 2019.

Crandall, J. W. and Goodrich, M. A. Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 2, pp. 1290–1295. IEEE, 2002.

de Winter, J. and Dodou, D. Preparing drivers for dangerous situations: A critical reflection on continuous shared control. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1050–1056, 2011. doi: 10.1109/ICSMC.2011.6083813.

Du, Y., Tiomkin, S., Kiciman, E., Polani, D., Abbeel, P., and Dragan, A. Ave: Assistance via empowerment. *Advances in Neural Information Processing Systems*, 33, 2020.

Erdogan, A. and Argall, B. The effect of robotic wheelchair control paradigm and interface on user performance, effort and preference: An experimental assessment. *Robotics and Autonomous Systems*, 94:282–297, 2017. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2017.04.013.

Gopinath, D., Jain, S., and Argall, B. D. Human-in-the-loop optimization of shared autonomy in assistive robotics. *IEEE Robotics and Automation Letters*, 2(1):247–254, 2017. doi: 10.1109/LRA.2016.2593928.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

Hasselt, H. V., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

Javdani, S., Srinivasa, S. S., and Bagnell, J. A. Shared autonomy via hindsight optimization. *Robotics science and systems: online proceedings*, 2015, 2015.

Javdani, S., Bagnell, J. A., and Srinivasa, S. S. Minimizing user cost for shared autonomy. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 621–622, 2016. doi: 10.1109/HRI.2016.7451886.

Javdani, S., Admoni, H., Pellegrinelli, S., Srinivasa, S. S., and Bagnell, J. A. Shared autonomy via hindsight optimization for teleoperation and teaming. *The International Journal of Robotics Research*, 37(7):717–742, 2018. doi: 10.1177/0278364918776060.

Jeon, H. J., Losey, D., and Sadigh, D. Shared Autonomy with Learned Latent Actions. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.011.

Kim, H. K., Biggs, J., Schloerb, W., Carmena, M., Lebedev, M. A., Nicolelis, M. A., and Srinivasan, M. A. Continuous shared control for stabilizing reaching and grasping with brain-machine interfaces. *IEEE Transactions on Biomedical Engineering*, 53(6):1164–1173, 2006.

Kofman, J., Wu, X., Luu, T. J., and Verma, S. Teleoperation of a robot manipulator using a vision-based human-robot interface. *IEEE transactions on industrial electronics*, 52 (5):1206–1219, 2005.

Kragic, D., Marayong, P., Li, M., Okamura, A. M., and Hager, G. D. Human-machine collaborative systems for microsurgical applications. *The International Journal of Robotics Research*, 24(9):731–741, 2005.

Levine, S. and Koltun, V. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.

Lin, L.-J. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

Matni, N. and Oishi, M. Reachability-based abstraction for an aircraft landing under shared control. In *2008 American Control Conference*, pp. 2278–2284, 2008. doi: 10.1109/ACC.2008.4586831.

Muelling, K., Venkatraman, A., Valois, J.-S., Downey, J. E., Weiss, J., Javdani, S., Hebert, M., Schwartz, A. B., Collinger, J. L., and Bagnell, J. A. Autonomy infused teleoperation with application to brain computer interface controlled manipulation. *Autonomous Robots*, 41(6): 1401–1422, 2017.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Palmer, P., Thursfield, C., and Judge, S. An evaluation of the psychosocial impact of assistive devices scale. In *Assistive technology: from virtuality to reality*, number 16, pp. 740–744. IOS Press, 2005.

Pilarski, P. M., Dawson, M. R., Degris, T., Fahimi, F., Carey, J. P., and Sutton, R. S. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *2011 IEEE international conference on rehabilitation robotics*, pp. 1–7. IEEE, 2011.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736, 2006.

Reddy, S., Dragan, A., and Levine, S. Shared autonomy via deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018a. doi: 10.15607/RSS.2018.XIV.005.

Reddy, S., Dragan, A., and Levine, S. Where do you think you're going?: Inferring beliefs about dynamics from behavior. In *Advances in Neural Information Processing Systems*, 2018b.

Reddy, S., Dragan, A. D., Levine, S., Legg, S., and Leike, J. Learning human objectives by evaluating hypothetical behavior. In *International Conference on Learning Representations*, 2020.

Sadigh, D., Sastry, S. S., Seshia, S. A., and Dragan, A. Information gathering actions over human internal state. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 66–73. IEEE, 2016.

Sadigh, D., Landolfi, N., Sastry, S. S., Seshia, S. A., and Dragan, A. D. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, 42(7):1405–1426, 2018.

Schaff, C. and Walter, M. Residual Policy Learning for Shared Autonomy. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.072.

Schröer, S., Killmann, I., Frank, B., Völker, M., Fiederer, L., Ball, T., and Burgard, W. An autonomous robotic assistant for drinking. In *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 6482–6487. IEEE, 2015.

Seo, S., Kennedy-Metz, L. R., Zenati, M. A., Shah, J. A., Dias, R. D., and Unhelkar, V. V. Towards an AI coach to infer team mental model alignment in healthcare. *CoRR*, abs/2102.08507, 2021. URL `https://arxiv.org/abs/2102.08507`.

Shanechi, M. M., Orsborn, A. L., and Carmena, J. M. Robust brain-machine interface design using optimal feedback control modeling and adaptive point process filtering. *PLoS computational biology*, 12(4):e1004730, 2016.

Trieu, H. T., Nguyen, H. T., and Willey, K. Shared control strategies for obstacle avoidance tasks in an intelligent wheelchair. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 4254–4257, 2008. doi: 10.1109/IEMBS.2008.4650149.

Verdonck, M., McCormack, C., and Chard, G. Irish occupational therapists' views of electronic assistive technology. *British Journal of Occupational Therapy*, 74(4):185–190, 2011.

Xbox. Xbox adaptive controller. `https://www.xbox.com/en-US/accessories/controllers/xbox-adaptive-controller`, 2018.

You, E. and Hauser, K. Assisted teleoperation strategies for aggressively controlling a robot arm with 2d input. In *Robotics: science and systems*, volume 7, pp. 354, 2012.