A Human-Centered Data-Driven Planner-Actor-Critic Architecture via Logic Programming

Daoming Lyu
Auburn University, Auburn, AL, USA
daoming.lyu@auburn.edu

Fangkai Yang
NVIDIA Corporation, Redmond, WA, USA
fangkaiy@nvidia.com

Bo Liu *
Auburn University, Auburn, AL, USA
boliu@auburn.edu

Steven Gustafson
Maana Inc., Bellevue, WA, USA
steven.gustafson@gmail.com

Recent successes of Reinforcement Learning (RL) allow an agent to learn policies that surpass human experts but suffers from being time-hungry and data-hungry. By contrast, human learning is significantly faster because prior and general knowledge and multiple information resources are utilized. In this paper, we propose a **P**lanner-**A**ctor-**C**ritic architecture for hu**MAN**-centered planning and learning (**PACMAN**), where an agent uses its prior, high-level, deterministic symbolic knowledge to plan for goal-directed actions, and also integrates the Actor-Critic algorithm of RL to fine-tune its behavior towards both environmental rewards and human feedback. This work is the first unified framework where knowledge-based planning, RL, and human teaching jointly contribute to the policy learning of an agent. Our experiments demonstrate that PACMAN leads to a significant jump-start at the early stage of learning, converges rapidly and with small variance, and is robust to inconsistent, infrequent, and misleading feedback.

1 Introduction

Programming agent that behaves intelligently in a complex domain is one of the central problems of artificial intelligence. Many tasks that we expect the agent to accomplish can be seen as a "sequential decision-making" problem, i.e., the agent makes a series of decisions on how to act in the environment based on its current situation. Recently, reinforcement learning (RL) algorithms have achieved tremendous success involving high-dimensional sensory inputs such as a training agent to play Atari games from raw pixel images [25]. This approach can learn fine-granular policies that surpass human experts but is criticized for being "data-hungry" and "time-hungry". It usually requires several millions of samples. Besides, it usually has a slow initial learning curve, with a bad performance level of the initial policy before a satisfactory policy can be obtained. By contrast, human learn to play video games significantly faster than the state-of-the-art RL algorithms due to two reasons. First, humans are embodied with prior, general, and abstract knowledge that can be applied and tailored towards a wide class of problems, such that individual problems are treated as special cases [40]. Second, humans learn policy from multiple information resources, including environmental reward signals, human feedback, and demonstrations.

Theoretical studies that try to simulate human problem-solving from the two aspects above have been done in both knowledge representation(KR) and RL communities. From the first perspective, research from KR community on modular action languages [20, 5, 10] proposed formal languages to encode a general-purpose library of actions that can be used to define a wide range of benchmark planning problems as special cases, leading to a representation that is elaboration tolerant and addressing the problem

*Correspondence to: Bo Liu<boliu@auburn.edu>.

B. Bogaerts, E. Erdem, P. Fodor, A. Formisano,
G. Ianni, D. Inclezan, G. Vidal, A. Villanueva,
M. De Vos, F. Yang (Eds.): International
Conference on Logic Programming 2019 (ICLP'19).
EPTCS 306, 2019, pp. 182–195, doi:10.4204/EPTCS.306.23

of *generality of AI* [24]. Meanwhile, researchers from the RL community focused on incorporating high-level abstraction into flat RL, leading to options framework for hierarchical RL [2], hierarchical abstract machines [27], and more recently, works that integrate symbolic knowledge represented in answer set programming (ASP) into reinforcement learning framework [19, 42, 21, 11]. From the second perspective, imitation learning, including learning from demonstration (LfD) [1] and inverse reinforcement learning (IRL) [26] tried to learn policies from examples of a human expert, or learn directly from human feedback [39, 15, 6], a.k.a, human-centered reinforcement learning (HCRL). In particular, recent studies showed that human feedback should be interpreted in terms of an advantage estimate (a value roughly corresponding to how much better or worse an action is compared to the current policy) as well to combat positive reward cycles and forgetting, leading to COACH framework [23, 22] based on Actor-Critic (AC) algorithm of RL [3]. While all of the existing research has shed lights on the importance of prior knowledge and learning from multiple information resources, there is no unified framework to bring them together.

In this paper, we argue that prior knowledge, learning from environmental reward and human teaching can jointly contribute to obtaining the optimal behavior. Prior, explicitly encoded knowledge is general and not sufficient to generate an optimal plan in a dynamic and changing environment, but can be used as a useful guideline to act and learn, leading to a jump start at early stages of learning. The agent further learns domain details to refine its behavior simultaneously from both environmental rewards and human feedback. Thus the prior knowledge is enriched with experience and tailored towards individual problem instances. Based on the motivation above, we propose the **P**lanner–**A**ctor–**C**ritic architecture for hu**MAN** centered planning and RL (**PACMAN**) framework. PACMAN interprets human feedback as the advantage function estimation similar to COACH framework and further incorporates prior, symbolic knowledge. The contribution of this paper is as follows.

- First we propose the **P**lanner–**A**ctor–**C**ritic (PAC) architecture for hu**MAN** centered planning and RL (**PACMAN**), featuring symbolic planner-actor-critic trio iteration, where planning and RL mutually benefit each other. In particular, the logical representation of action effects is dynamically generated by sampling a stochastic policy learned from Actor-Critic algorithms of RL. PACMAN allows the symbolic knowledge and actor-critic framework to integrate into a unified framework seamlessly.
- Second, we enable learning simultaneously from both environmental reward and human feedback, which can accelerate the learning process of interactive RL, and improve the tolerance of misleading feedback from human users.

To the best of our knowledge, this paper is the first work in which learning happens simultaneously from human feedback, environmental reward, and prior symbolic knowledge. While our framework is generic in nature, we choose to use ASP-based action language $\mathscr{BC}$ [17], answer set solver CLINGO to do symbolic planning and conduct our experiment. In principle, each component can be implemented using different techniques. The evaluation of the framework is performed on RL benchmark problems such as Four Rooms and Taxi domains. Various scenarios of human feedback are considered, including cases of ideal, infrequent, inconsistent, and both infrequent and inconsistent with helpful feedback and misleading feedback. Our experiments show that PACMAN leads to a significant jump-start at early stages of learning, converges faster and with smaller variance, and is robust in inconsistent, infrequent cases even with misleading feedback.

The rest of the paper is organized as follows: Section 2 introduces the related work. Section 3 briefly introduces the background about symbolic planning and actor-critic (AC) framework. PACMAN architecture is presented in Section 4 and experimental results are shown in Section 5.

2 Related Work

There is a long history of work that combines symbolic planning with reinforcement learning [27, 32, 31, 9, 18, 19]. These approaches were based on integrating symbolic planning with value iteration methods of reinforcement learning, and in their work, there was no bidirectional communication loop between planning and learning so that they could not mutually benefit each other. The latest work in this direction is PEORL framework [42] and SDRL [21], where ASP-based planning was integrated with R-learning [35] into planning–learning loop. PACMAN architecture is a new framework of integrating symbolic planning with RL, in particular, integrating planning with AC algorithm for the first time, and also features bidirectional communication between planning and learning.

Learning from human feedback takes the framework of reinforcement learning, and incorporate human feedback into the reward structure [39, 14, 16], information directly on policy [38, 15, 6], or advantage function [23, 22]. Learning from both human feedback and environmental rewards were investigated [39, 16, 6], mainly integrating the human feedback to reward or value function via reward shaping or Q-value shaping. Such methods do not handle well the samples with missing human feedback, and in reality, human feedback may be infrequent. They also suffer from the ambiguity of statement-like reward such as "that's right" or "no, this is wrong": such statements are easy to be transformed to binary or discrete value but are difficult to incorporate with continuous-valued reward and value signals, as pointed out by [6]. Recent work of COACH [23, 22] showed that human feedback was more likely to be policy-dependent, and advantage function provides a better model of human feedback, but it does not consider learning simultaneously from environmental reward and human feedback. Furthermore, none of these work considers the setting where an agent is equipped with prior knowledge and generates a goal-directed plan that is further to be fine-tuned by reinforcement learning and a human user. Integrating COACH-style human feedback into PACMAN, our framework allows the integration of symbolic planning into the learning process, where environmental reward and human feedback can be unified into the advantage function to shape the agent's behavior in the context of long-term planning.

3 Preliminaries

This section introduces the basics of symbolic planning and actor-critic framework.

3.1 Symbolic Planning

Symbolic planning is concerned with describing preconditions and effects of actions using a formal language and automating plan generation. An *action description D* in the language $\mathscr{BC}$ includes two kinds of symbols, *fluent constants* that represent the properties of the world, with the signature denoted as $\sigma_F(D)$, and *action constants*, with the signature denoted as $\sigma_A(D)$. A *fluent atom* is an expression of the form $f = v$, where f is a fluent constant and v is an element of its domain. For the Boolean domain, denote $f = \mathbf{t}$ as f and $f = \mathbf{f}$ as $\sim f$. An action description is a finite set of *causal laws* that describe how fluent atoms are related with each other in a single time step, or how their values are changed from one step to another, possibly by executing actions. For instance,

$$A \textbf{ if } A_1, \ldots, A_m \tag{1}$$

is a *static law* that states at a time step, if $A_1, \ldots, A_m$ holds then A is true.

$$a \textbf{ causes } A_0 \textbf{ if } A_1, \ldots, A_m \tag{2}$$

is a *dynamic law*, stating that at any time step, if $A_1,\ldots,A_m$ holds, by executing action a, A_0 holds in the next step.[1] An action description captures a dynamic transition system. Let I and G be states. The triple (I,G,D) is called a planning problem. (I,G,D) has a plan of length $l-1$ iff there exists a transition path of length l such that $I=s_1$ and $G=s_l$. Throughout the paper, we use Π to denote both the plan and the transition path by following the plan. Genearating a plan of length l can be achieved by solving the answer set program $PN_l(D)$, consisting of rules translated from D and appending timestamps from 1 to l, via a translating function PN. For instance, PN_l turns (1) to

$$i{:}A \leftarrow i{:}A_1,\ldots,i{:}A_m,$$

where $1\leq i\leq l$ and (2) to

$$i+1{:}A \leftarrow i{:}a, i{:}A_1,\ldots,i{:}A_m,$$

where $1\leq i<l$. See [17] for details.

3.2 Actor-Critic Architecture

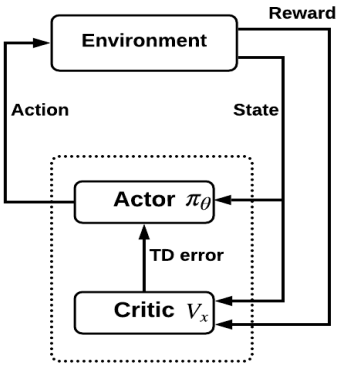


Figure 1: The framework of actor-critic

A Markov Decision Process (MDP) is defined as a tuple $(\mathscr{S},\mathscr{A},P^a_{ss'},r,\gamma)$, where $\mathscr{S}$ and $\mathscr{A}$ are the sets of symbols denoting state space and action space, the transition kernel $P^a_{ss'}$ specifies the probability of transition from state $s\in\mathscr{S}$ to state $s'\in\mathscr{S}$ by taking action $a\in\mathscr{A}$, $r(s,a):\mathscr{S}\times\mathscr{A}\mapsto\mathbb{R}$ is a reward function bounded by $r_{\max}$, and $0\leq\gamma<1$ is a discount factor. A solution to an MDP is a policy $\pi:\mathscr{S}\mapsto\mathscr{A}$ that maps a state to an action. RL concerns learning a near-optimal policy by executing actions and observing the state transitions and rewards, and it can be applied even when the underlying MDP is not explicitly given.

An actor-critic [28, 3] approach is a framework of reinforcement learning, which has two components: the actor and the critic, as shown in Figure 1. Typically, the actor is a policy function π_θ parameterized by θ for action selection, while the critic is a state-value function V_x parameterized by x to criticize the action made by the actor. For example, after each action selection, the critic will evaluate the new state to determine whether things have gone better or worse than expected by computing the temporal difference (TD) error [36],

$$\delta(s,a,s')=r(s,a)+\gamma V_x(s')-V_x(s).$$

[1] In $\mathscr{BC}$, causal laws are defined in a more general form. In this paper, without loss of generality, we assume the above form of causal laws for defining effects of actions.

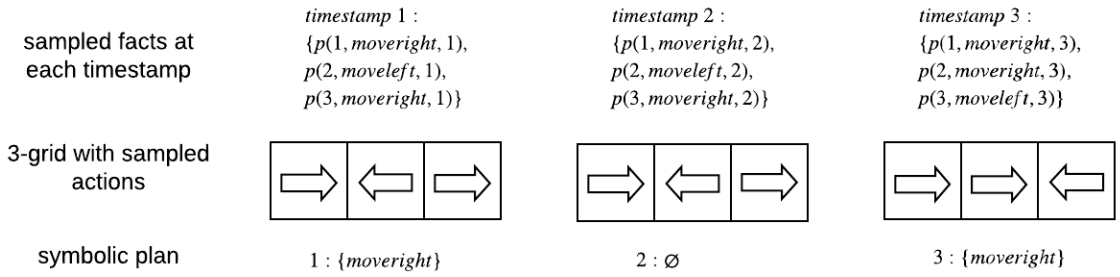


Figure 2: A possible sample-based planning result for 3-grid domain

If the TD error is positive, it suggests that the tendency to select current action a should be strengthened for the future, whereas if the TD error is negative, it suggests the tendency should be weakened. This TD error is actually an estimate of advantage function [34].

4 PACMAN Architecture

In this section, we will present our PACMAN in detail.

4.1 Sample-based Symbolic Planning

We introduce a *sample-based planning problem* as a tuple (I,G,D,π_θ) where I is the initial state condition, G is a goal state condition, D is an action description in $\mathscr{BC}$, and π_θ is a stochastic policy function parameterized by θ, i.e., a mapping $\mathscr{S}\times\mathscr{A}\mapsto[0,1]$. For D, defines its l-step *sampled action description* $D^l_\pi = D_s \cup D_d \cup \bigcup_{t=1}^{l} P^t_\pi$ with respect to policy π and time stamp $1 \le t \le l$, where

- D_s is a set of causal laws consisting of static laws and dynamic laws that does not contains action symbols;
- D_d is a set of causal laws obtained by turning each dynamic law of the form

  $$a \textbf{ causes } A_0 \textbf{ if } A_1,\ldots,A_m,$$

  into rules of the form

  $$a \textbf{ causes } A_0 \textbf{ if } A_1,\ldots,A_m,p(s,a)$$

  where p is a newly introduced fluent symbol and $\{A_1,\ldots,A_m\}\subseteq s$, for $s\in\mathscr{S}$; and
- P^t_π is a set of facts sampled at timestamp t that contains $p(a,s)$ such that

  $$p(s,a)\in P^t_\pi \sim \pi_\theta(\cdot|s)$$

  where for $s\in\mathscr{S}, A\in\mathscr{A}$.

Define the translation $\mathscr{T}(D^l_\pi)$ as

$$PN_l(D_s\cup D_d)\cup\bigcup_{t=1}^{l}\{p(s,a,t),\text{for } p(s,a)\in P^t_\pi\}$$

Algorithm 1 Sample-based Symbolic Planning

Require: a sample based planning problem (I, G, D, π_θ)

$\Pi \Leftarrow \emptyset$
calculate D_π^0
$k \Leftarrow 1$
while $\Pi = \emptyset$ and $k < maxstamp$ **do**
  sample $P_{\pi,k}$ over $p(a,s) \sim \pi_\theta(\cdot|s)$ for $s \in \mathscr{S}$, $a \in \mathscr{A}$
  $D_\pi^k \Leftarrow D_\pi^{k-1} \cup P_\pi^k$
  $\Pi \leftarrow$ CLINGO.*solve*$(I \cup G \cup \mathscr{T}(D_\pi^k))$
  $k \leftarrow k+1$
end while
return Π

that turns D_π^l into answer set program.

A *sample-based plan* up to length l of (I, G, D, π_θ) can be calculated from the answer set of program $\mathscr{T}(D_\pi^l)$ such that I and G are satisfied. The planning algorithm is shown in Algorithm 1.

Example. Consider 3×1 horizontal gridworld where the grids are marked as state 1, 2, 3, horizontally. Initially the agent is located in state 1. The goal is to be located in state 3. The agent can move to left or right. Using action language $\mathscr{BC}$, moving to the left and moving to the right can be formulated as dynamic laws

$$\begin{aligned} \mathit{moveleft} \ \ &\textbf{causes}\ \mathit{Loc} = L-1 \ \textbf{if}\ \mathit{Loc} = L. \\ \mathit{moveright} \ \ &\textbf{causes}\ \mathit{Loc} = L+1 \ \textbf{if}\ \mathit{Loc} = L. \end{aligned}$$

Turning them into sample-based action description leads to

$$\begin{aligned} \mathit{moveleft} \ \ &\textbf{causes}\ \mathit{Loc} = L-1 \ \textbf{if}\ \mathit{Loc} = L, p(L, \mathit{moveleft}). \\ \mathit{moveright} \ \ &\textbf{causes}\ \mathit{Loc} = L+1 \ \textbf{if}\ \mathit{Loc} = L, p(L, \mathit{moveright}). \end{aligned}$$

A policy estimator π_θ accepts an input state and output probability distribution on actions *moveleft* and *moveright*. Sampling π_θ with input s at time stamp i generates a fact of the form $p(a,s,i)$ where $a \in \{\mathit{moveleft}, \mathit{moveright}\}$ following the probability distribution of $\pi_\theta(\cdot|s)$.

At any timestamp, CLINGO solves answer set program consisting of rules translated from the above causal laws:

```
loc(L-1,k+1):-moveleft(k),loc(L,k), p(L,moveleft,k).
loc(L+1,k+1):-moveright(k),loc(L,k), p(L,moveright,k).
```

for time stamp $1,\ldots,k$, plus a set of facts of the form `p(a,s,i)` sampled from π_θ where for states $s \in \{1,2,3\}$ and timestamps $i \in \{1,\ldots,k\}$. Note that the planner can skip time stamps if there is no possible actions to use to generate plan, based on sampled results. Figure 2 shows a possible sampling results over 3 timestamps, and a plan of 2 steps is generated to achieve the goal, where time stamp 2 is skipped with no planned actions.

Since sample-based planning calls a policy approximator as an oracle to obtain probability distribution and samples the distribution to obtain available actions, it can be easily applied to other planning techniques such as PDDL planning. For instance, the policy appropriator can be used along with heuristics on relaxed planning graph [8].

4.2 Planning and Learning Loop

The planning and learning loop for PACMAN, as shown in Algorithm 2, starts from a random policy (uniform distribution over action space), and then generate a sample-based symbolic plan. After that, it follows the plan to explore and update the policy function π_θ, leading to an improved policy, which is used to generate the next plan.

Algorithm 2 Planner-Actor-Critic (PAC) Learning

Require: (I, G, D, π_θ) and a value function estimator V
1: $episode = 0$
2: **for** $episode < maxepisode$ **do**
3: Generate symbolic plan Π from (I, G, D, π_θ) by Algorithm 1
4: **for** $\langle s_i, a_i, r_i, s_{i+1} \rangle \in \Pi$ **do**
5: Compute TD error as in Eq. (3) as a stochastic estimation of advantage function.
6: Update V_x via Eq. (4).
7: **if** human feedback f_i is available **then**
8: Replace TD error with human feedback f_i.
9: **end if**
10: Update π_θ via Eq. (5).
11: **end for**
12: $episode \leftarrow episode + 1$
13: **end for**

At the i-th time step, the sample is formulated as (s_i, a_i, r_i, s_{i+1}), then the TD error is computed as

$$\delta_i = r_i + \gamma V_{x_{i+1}}(s_{i+1}) - V_{x_{i+1}}(s_i), \tag{3}$$

which is a stochastic estimation of the advantage function. The value function V_x is updated using reinforcement learning approaches, such as TD method [36]:

$$x_{i+1} = x_i + \alpha \delta_i \nabla V_{x_i}(s_i), \tag{4}$$

where α is the learning rate. The policy function π_θ will be updated by

$$\theta_{i+1} = \theta_i + \beta \delta_i \nabla \log \pi_{\theta_i}(a_i|s_i), \tag{5}$$

where β is the learning rate. If the human feedback signal f_i is available, then this feedback signal will replace the previous computed TD error and be used to update the policy function; If there is no human feedback signal available at this iteration, TD error will be used to update the policy function directly. For this reason, human feedback here can be interpreted as guiding exploration towards human preferred state-action pairs.

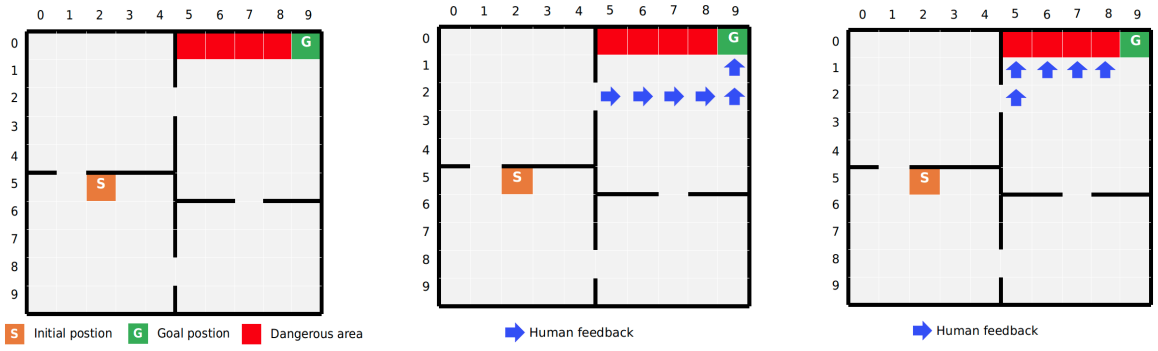
5 Experiment

We evaluate our method in two RL-benchmark problems: Four Rooms [37] and Taxi domain [2]. For all domains, we consider the discrete value of (positive or negative) feedback with the cases of ideal (feedback is always available without reverting), infrequent (only giving feedback at 50% probability),

inconsistent (randomly reverting feedback at 30% probability) and infrequent+inconsistent (only giving feedback at 50% probability, while randomly reverting feedback at 30% probability). We compare the performance of PACMAN with 3 methods: TAMER+RL Reward Shaping from [16], BQL Reward Shaping from [6], and PACMAN without symbolic planner (AC with Human Feedback) as our ablation analysis. All plotting curves are averaged over 10 runs, and the shadow around the curve denotes the variance.

5.1 Four Rooms

Four Rooms domain is shown in Figure 3a. In this 10×10 grid, there are 4 rooms and an agent navigating from the initial position (5,2) to the goal position (0,9). If the agent can successfully achieve the task, it would receive a reward of +5. And it may obtain a reward of -10 if the agent steps into the red grids (dangerous area). Each move will cost -1.



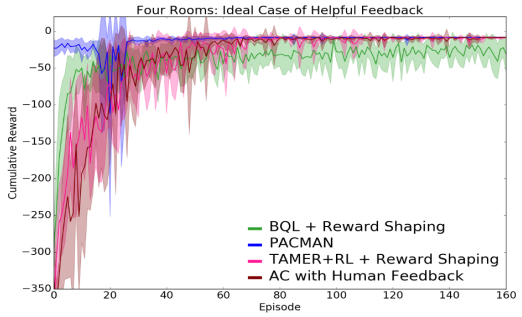
(a) Four-room domain (b) Helpful Feedback (c) Misleading Feedback

Figure 3: The snapshot of 2 scenarios on Four Rooms domain

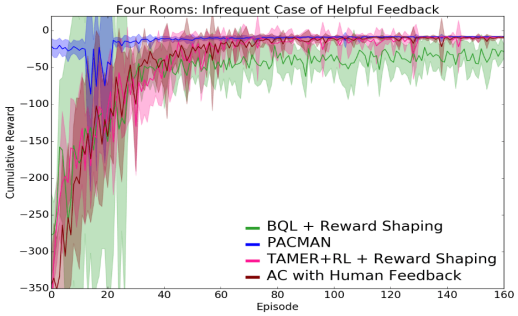
The human feedback of Four Rooms domain concerns 2 scenarios.

- **Helpful feedback**. Consider an experienced user that wants to help the agent to navigate safer and better, such that the agent can stay away from the dangerous area and reach the goal position with the shortest path. Therefore, human feedback can guide the agent to improve its behavior towards the task, as shown in Figure 3b.
- **Misleading feedback**. Consider an inexperienced user who doesn't know there is a dangerous area, but wants the agent to step into those red grids (Figure 3c). In this case, human feedback contradicts with the behavior that the agent learns from an environmental reward.

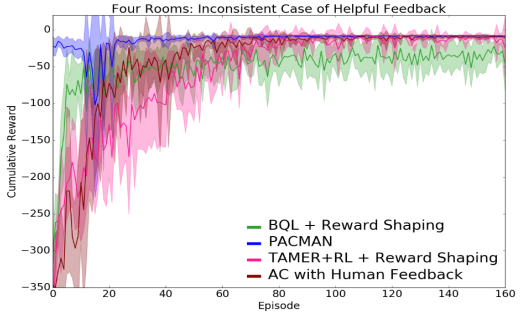
The results are shown in Figure 4 and Figure 5. As we can see, PACMAN has a jump-start and quickly converged with small variance, compared to BQL Reward Shaping, TAMER+RL Reward Shaping, and AC with Human Feedback under four different cases. This is because symbolic planning can lead to goal-directed behavior, which would bias exploration. Though the infrequent case, inconsistent case, and their combination case for both helpful feedback and misleading feedback can lead to more uncertainty, the performance of PACMAN remains unaffected, which means more robust than others. Meticulous readers may find that there is a large variance in the initial stage of PACMAN, especially in Figure 4 and Figure 5, this is due to the reason that the symbolic planner will first generate a short plan that is reasonably well, then the symbolic planner will perform exploration by generating longer plans. After doing the exploration, the symbolic planner will converge to the short plan with the optimal



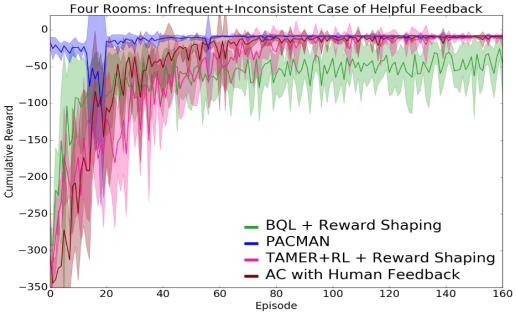
(a) Ideal Case



(b) Infrequent Case

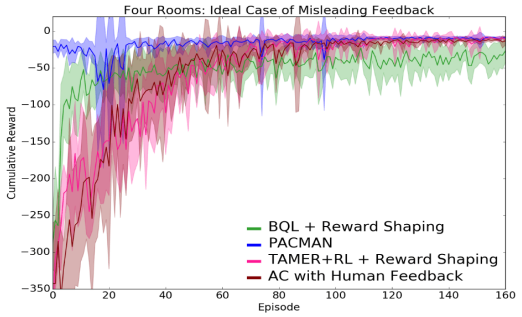


(c) Inconsistent Case

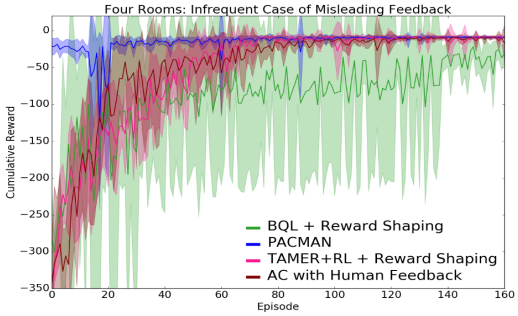


(d) Infrequent+Inconsistent Case

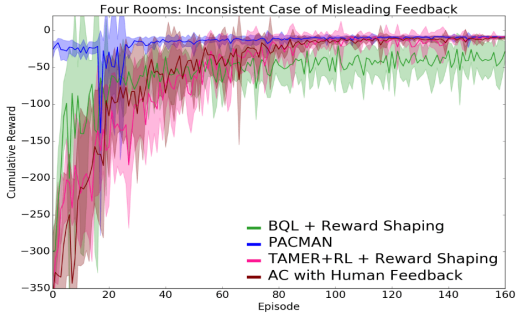
Figure 4: Four-room with Helpful Feedback: Learning Curves



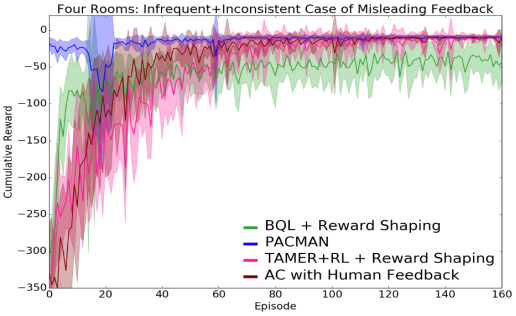
(a) Ideal Case



(b) Infrequent Case



(c) Inconsistent Case



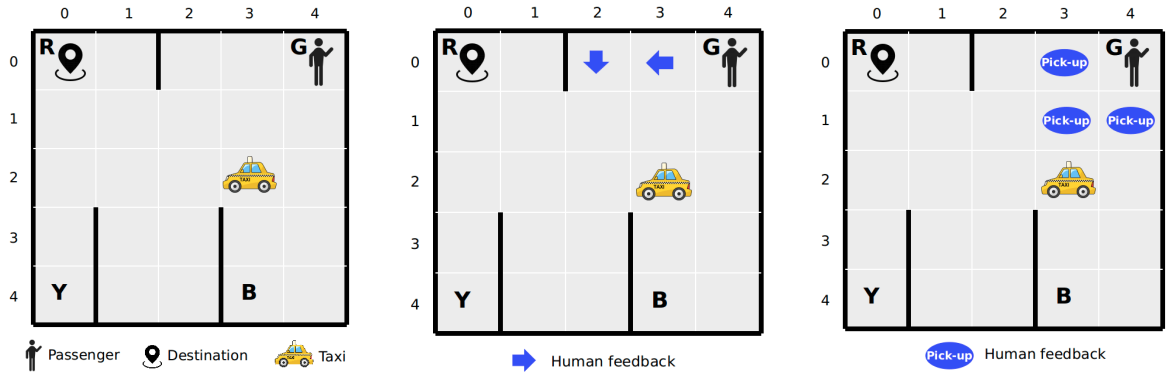
(d) Infrequent+Inconsistent Case

Figure 5: Four-room with Misleading Feedback: Learning Curves

solution. This large variance can be partially alleviated by setting the maximal number of actions in a plan to reduce plan space.

5.2 Taxi Domain

Taxi domain concerns a 5×5 grid (Figure 6a) where a taxi needs to navigate to a passenger, pick up the passenger, then navigate to the destination and drop off the passenger. Each move has a reward of -1. Successful drop-off received a reward of +20, while improper pick-up or drop-off would receive a reward of -10. When formulating the domain symbolically, we specify that precondition of performing picking up a passenger is that the taxi has to be located in the same place as the passenger.



(a) Taxi domain (b) Helpful Feedback (c) Misleading Feedback

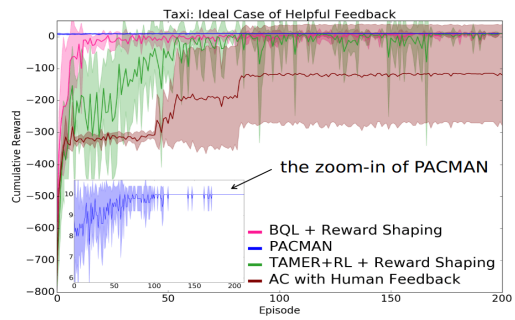
Figure 6: The snapshot of 2 scenarios on Taxi domain

We consider human feedback in the following two scenarios:

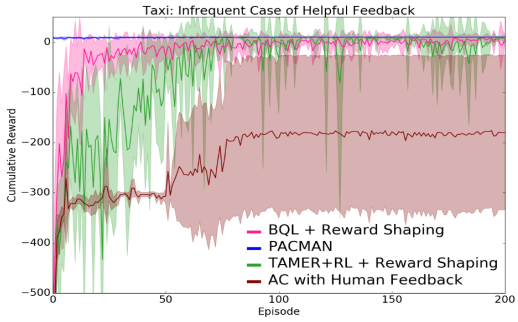
- **Helpful feedback**. During the rush hour, the passenger can suggest a path that would guide the taxi to detour and avoid the slow traffic, which is shown in Figure 6b. The agent should learn a more preferred route from human's feedback.

- **Misleading feedback**. Consider a passenger who is not familiar enough with the area and may inaccurately inform the taxi of his location before approaching the passenger (Figure 6c), which is the wrong action and will mislead the taxi. In this case, the feedback conflicts with symbolic knowledge specified by PACMAN and the agent should learn to ignore such feedback.

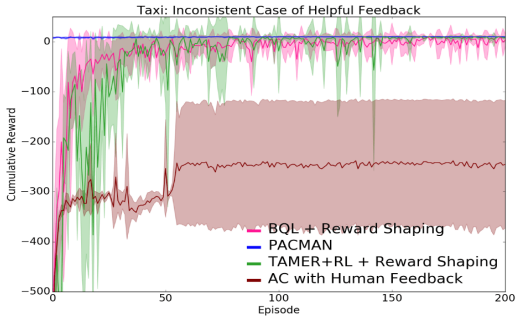
The results are shown in Figure 7 and Figure 8. In the scenario of helpful feedback, the curve of PACMAN has the smallest variance so that it looks like a straight line. But in the case of Infrequent+Inconsistent, there is a big chattering in the initial stage of PACMAN, that's because the symbolic planner is trying some longer plans to do the exploration. In the misleading feedback scenario, the learning speed of the other methods except for PACMAN is quite slow. That's because the human feedback will misguide the agent to perform the improper action that can result in the penalty, and the agent needs a long time to correct its behavior via learning from the environmental reward. But PACMAN keeps unaffected in this case due to the symbolic knowledge that a taxi can pick up the passenger only when it moves to the passenger's location.



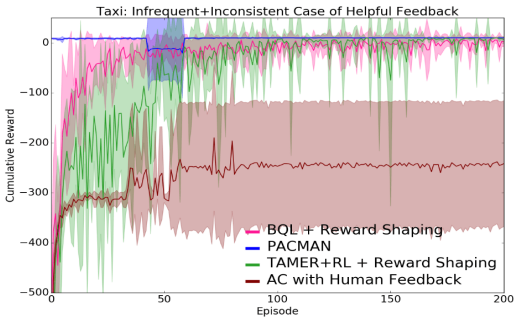
(a) Ideal Case



(b) Infrequent Case

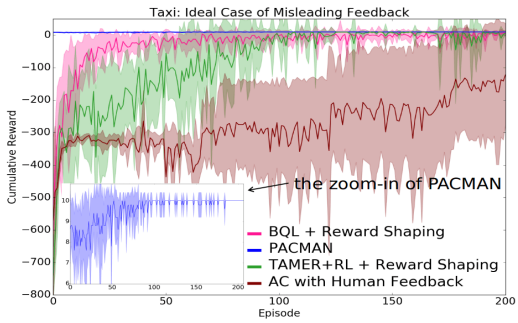


(c) Inconsistent Case

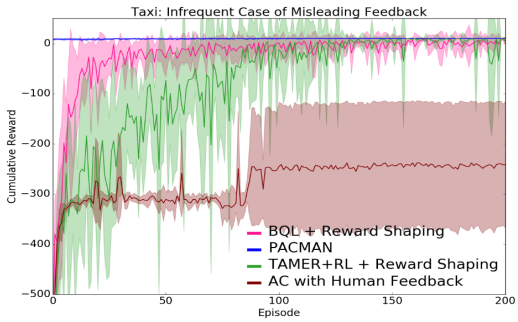


(d) Infrequent+Inconsistent Case

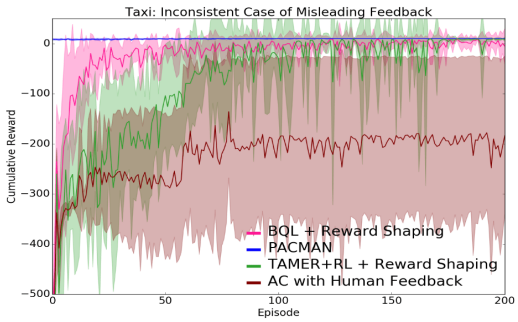
Figure 7: Taxi with Helpful Feedback: Learning Curves



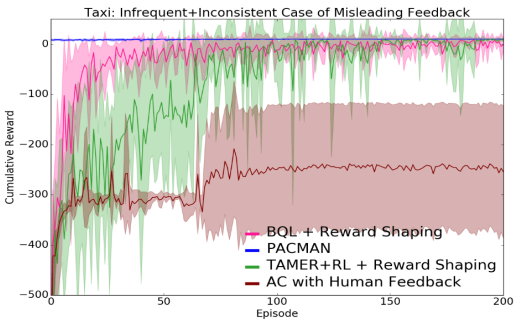
(a) Ideal Case



(b) Infrequent Case



(c) Inconsistent Case



(d) Infrequent+Inconsistent Case

Figure 8: Taxi with Misleading Feedback: Learning Curves

6 Conclusion

In this paper, we propose the PACMAN framework, which can simultaneously consider prior knowledge, learning from environmental reward and human teaching together and jointly contribute to obtaining the optimal policy. Experiments show that the PACMAN leads to significant jump-start at early stages of learning, converges faster and with smaller variance, and is robust to inconsistent and infrequent cases even with misleading feedback.

Our future work involves investigating using the PACMAN to perform decision making from high-dimensional sensory input such as pixel images, autonomous driving where the vehicle can learn human's preference on comfort and driving behavior, as well as mobile service robots.

7 Acknowledgment

This research was supported in part by the National Science Foundation (NSF) under grants NSF IIS-1910794. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] B. D. Argall, S. Chernova, M. Veloso & B. Browning (2009): *A survey of robot learning from demonstration*. *Robotics and autonomous systems* 57(5), pp. 469–483, doi:10.1016/j.robot.2008.10.024.

[2] A. Barto & S. Mahadevan (2003): *Recent Advances in Hierarchical Reinforcement Learning*. *Discrete Event Systems Journal* 13, pp. 41–77, doi:10.1023/A:1022140919877.

[3] S. Bhatnagar, R. Sutton, M. Ghavamzadeh & M. Lee (2009): *Natural Actor-Critic Algorithms*. *Automatica* 45(11), pp. 2471–2482, doi:10.1016/j.automatica.2009.07.008.

[4] A. Cimatti, M. Pistore & P. Traverso (2008): *Automated planning*. In Frank van Harmelen, Vladimir Lifschitz & Bruce Porter, editors: *Handbook of Knowledge Representation*, Elsevier, doi:10.1016/S1574-6526(07)03022-2.

[5] S. T. Erdoğan (2008): *A Library of General-Purpose Action Descriptions*. Ph.D. thesis, University of Texas at Austin.

[6] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell & A. L. Thomaz (2013): *Policy shaping: Integrating human feedback with reinforcement learning*. In: *Advances in neural information processing systems (NeurIPS)*, pp. 2625–2633.

[7] M. Hanheide, M. Göbelbecker, G. S Horn et al. (2015): *Robot task planning and explanation in open and uncertain worlds*. *Artificial Intelligence*, doi:10.1016/j.artint.2015.08.008.

[8] M. Helmert (2006): *The fast downward planning system*. *Journal of Artificial Intelligence Research* 26, pp. 191–246, doi:10.1613/jair.1705.

[9] C. Hogg, U. Kuter & H. Munoz-Avila (2010): *Learning Methods to Generate Good Plans: Integrating HTN Learning and Reinforcement Learning.* In: *Association for the Advancement of Artificial Intelligence (AAAI).*

[10] D. Inclezan & M. Gelfond (2016): *Modular action language ALM*. *Theory and Practice of Logic Programming* 16(2), pp. 189–235, doi:10.1080/11663081.2013.798954.

[11] Y. Jiang, F. Yang, S. Zhang & P. Stone (2019): *Task-Motion Planning with Reinforcement Learning for Adaptable Mobile Service Robots.* In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

[12] P. Khandelwal, F. Yang, M. Leonetti, V. Lifschitz & P. Stone (2014): *Planning in Action Language $\mathscr{BC}$ while Learning Action Costs for Mobile Robots.* In: International Conference on Automated Planning and Scheduling (ICAPS).

[13] P. Khandelwal, S. Zhang, J. Sinapov, M. Leonetti, J. Thomason, F. Yang, I. Gori, M. Svetlik, P. Khante & V. Lifschitz (2017): *BWIBots: A platform for bridging the gap between AI and human–robot interaction research*. The International Journal of Robotics Research 36(5-7), pp. 635–659, doi:10.1007/978-3-319-23264-5_42.

[14] W. B. Knox & P. Stone (2009): *Interactively shaping agents via human reinforcement: The TAMER framework*. In: Proceedings of the fifth International Conference on Knowledge Capture, ACM, pp. 9–16, doi:10.1145/1597735.1597738.

[15] W. B. Knox & P. Stone (2010): *Combining manual feedback with subsequent MDP reward signals for reinforcement learning*. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, pp. 5–12.

[16] W. B. Knox & P. Stone (2012): *Reinforcement learning from simultaneous human and MDP reward*. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, pp. 475–482.

[17] J. Lee, V. Lifschitz & F. Yang (2013): *Action Language $\mathscr{BC}$: A Preliminary Report*. In: International Joint Conference on Artificial Intelligence (IJCAI).

[18] M. Leonetti, L. Iocchi & F. Patrizi (2012): *Automatic generation and learning of finite-state controllers*. In: International Conference on Artificial Intelligence: Methodology, Systems, and Applications, Springer, pp. 135–144, doi:10.1007/3-540-61474-5_68.

[19] M. Leonetti, L. Iocchi & P. Stone (2016): *A synthesis of automated planning and reinforcement learning for efficient, robust decision-making*. Artificial Intelligence 241, pp. 103–130, doi:10.1016/j.artint.2016.07.004.

[20] V. Lifschitz & W. Ren (2006): *A modular action description language*. In: Association for the Advancement of Artificial Intelligence (AAAI), pp. 853–859.

[21] D. Lyu, F. Yang, B. Liu & S. Gustafson (2019): *SDRL: Interpretable and Data-efficient Deep Reinforcement LearningLeveraging Symbolic Planning*. In: Association for the Advancement of Artificial Intelligence (AAAI).

[22] J. MacGlashan, M. K Ho, R. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor & M. L. Littman (2017): *Interactive Learning from Policy-Dependent Human Feedback*. In: International Conference on Machine Learning (ICML).

[23] J. MacGlashan, M. L. Littman, D. L. Roberts, R. Loftin, B. Peng & M. E. Taylor (2016): *Convergent Actor Critic by Humans*. In: International Conference on Intelligent Robots and Systems.

[24] John McCarthy (1987): *Generality in Artificial Intelligence*. Communications of the ACM (CACM), doi:10.1145/33447.33448.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A Rusu, J. Veness, M. G Bellemare, A. Graves, M. Riedmiller, A. K Fidjeland, G. Ostrovski et al. (2015): *Human-level control through deep reinforcement learning*. Nature 518(7540), pp. 529–533, doi:10.1016/S0004-3702(98)00023-X.

[26] A. Y. Ng, S. J. Russell et al. (2000): *Algorithms for inverse reinforcement learning.* In: International Conference on Machine Learning (ICML), 1, p. 2.

[27] R. Parr & S. J. Russell (1998): *Reinforcement learning with hierarchies of machines*. In: Advances in neural information processing systems (NeurIPS), pp. 1043–1049.

[28] J. Peters & S. Schaal (2008): *Natural actor-critic*. Neurocomputing 71(7), pp. 1180–1190, doi:10.1016/j.neucom.2007.11.026.

[29] S. Rosenthal, M. M. Veloso & A. K. Dey (2011): *Learning Accuracy and Availability of Humans Who Help Mobile Robots.* In: Association for the Advancement of Artificial Intelligence (AAAI).

[30] S. L. Rosenthal (2012): *Human-centered planning for effective task autonomy*. Technical Report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

[31] M. R.K. Ryan (2002): *Using abstract models of behaviours to automatically generate reinforcement learning hierarchies*. In: *In Proceedings of The 19th International Conference on Machine Learning (ICML)*, Morgan Kaufmann, pp. 522–529.

[32] M. R.K. Ryan & M. D. Pendrith (1998): *RL-TOPs: An Architecture for Modularity and Re-Use in Reinforcement Learning*. In: *In Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, Morgan Kaufmann, pp. 481–487.

[33] J. Schulman, S. Levine, P. Abbeel, M. Jordan & P. Moritz (2015): *Trust region policy optimization*. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 1889–1897.

[34] J. Schulman, P. Moritz, S. Levine, M. Jordan & P. Abbeel (2015): *High-dimensional continuous control using generalized advantage estimation*. arXiv preprint arXiv:1506.02438.

[35] A. Schwartz (1993): *A Reinforcement Learning Method for Maximizing Undiscounted Rewards*. In: *International Conference on Machine Learning (ICML)*, Morgan Kaufmann, San Francisco, CA, doi:10.1016/B978-1-55860-307-3.50045-9.

[36] R. S. Sutton & A. G. Barto (2018): *Reinforcement learning: An introduction*. MIT press.

[37] R. S. Sutton, D. Precup & S. Singh (1999): *Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning*. *Artificial intelligence* 112(1-2), pp. 181–211, doi:10.1016/S0004-3702(99)00052-1.

[38] A. L. Thomaz & C. Breazeal (2008): *Teachable robots: Understanding human teaching behavior to build more effective robot learners*. *Artificial Intelligence* 172(6-7), pp. 716–737, doi:10.1016/j.artint.2007.09.009.

[39] A. L. Thomaz, C. Breazeal et al. (2006): *Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance*. In: *Aaai*, 6, Boston, MA, pp. 1000–1005.

[40] P. A. Tsividis, T. Pouncy, J. L. Xu, J. B. Tenenbaum & S. J. Gershman (2017): *Human learning in Atari*.

[41] R. J Williams (1992): *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. *Machine learning* 8(3-4), pp. 229–256, doi:10.1023/A:1022672621406.

[42] F. Yang, D. Lyu, B. Liu & S. Gustafson (2018): *PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-Making*. In: *International Joint Conference of Artificial Intelligence (IJCAI)*, doi:10.24963/ijcai.2018/675.