Efficient Imitation Learning with Local Trajectory Optimization

Jialin Song [1] **Joe Wenjie Jiang** [2] **Amir Yazdanbakhsh** [2] **Ebrahim Songhori** [2] **Anna Goldie** [2] **Navdeep Jaitly** [3] **Azalia Mirhoseini** [2]

Abstract

Imitation learning is a powerful approach to optimize sequential decision making policies from demonstrations. Most strategies in imitation learning rely on per-step supervision from pre-collected demonstrations as in behavioral cloning (Pomerleau, 1989) or from interactive expert policy queries such as DAgger (Ross et al., 2011). In this work, we present a unified view of behavioral cloning and DAgger through the lens of local trajectory optimization, which offers a means of interpolating between them. We provide theoretical justification for the proposed local trajectory optimization algorithm and show empirically that our method, POLISH (Policy Optimization by Local Improvement through Search), is much faster than methods that plan globally, speeding up training by a factor of up to 14 in wall clock time. Furthermore, the resulting policy outperforms strong baselines in both reinforcement learning and imitation learning.

1. Introduction

Reinforcement learning (RL) has seen a great deal of success in recent years in various domains including gaming (Mnih et al., 2015; Silver et al., 2016), robotics (Gu et al., 2017; Singh et al., 2019) and systems and chip design (Mirhoseini et al., 2017; 2018; 2020; Zhou et al., 2019). However, a well-known disadvantage of reinforcement learning approaches is high sample complexity. This issue can be mitigated by learning from expert behavior, i.e., imitation learning. A variety of strategies have been developed in imitation learning to learn from expert behavior, where the expert can be a human (Stadie et al., 2017) or a pre-trained policy (Ho & Ermon, 2016). Earlier approaches, such as behavioral cloning (BC), rely on training models to mimic expert behavior at various states in the demonstration data (Pomerleau, 1989). However, these models suffer from generalization issues arising from state distribution shift (Ross & Bagnell, 2010). To mitigate this issue, DAgger (Ross et al., 2011) uses an expert policy to provide interactive feedback. Although BC and DAgger differ on how expert feedback is collected, they both reduce the learning problem to a (cost-sensitive) supervised learning problem in the end. In this work, we present a unified framework for interactive imitation learning that uses local search to improve the policy, which balances generalization with computation. Furthermore, we show that BC and DAgger are special cases within the proposed framework.

Most of imitation learning algorithms assume access to an expert policy. However, it is not always possible to have such an expert policy, for example for novel tasks. To address this issue, we apply Monte Carlo Tree Search (MCTS) to perform local trajectory optimization and use the improved trajectories as demonstrations. In this setting, we show that a new algorithm, called POLISH, can provide wall-clock time speedup for learning as well.

In summary, our main contributions are:

- A unified framework for imitation learning that includes BC and DAgger as special cases. We show that by varying the time horizon for local trajectory improvements, we can interpolate between BC and DAgger.
- A theoretical analysis for how the time horizon parameter affects the imitation learning outcome.
- A strong empirical study on a suite of high-dimensional continuous control problems based on both sample efficiency and training time. Additionally, we validate using MCTS as a local trajectory improvement procedure.

2. Related Work

Imitation learning (IL) refers to the problem of learning to perform a task from expert demonstrations. Behavioral cloning (Pomerleau, 1989) is a popular approach which maximizes the likelihood of expert actions under the agent's policy but it suffers from distributional shift (Ross & Bagnell, 2010).

[1]Work done as a Google intern; Caltech [2]Google Research, Brain Team [3]Work done at Google; The D. E. Shaw Group. Correspondence to: Jialin Song <jssong@caltech.edu>.

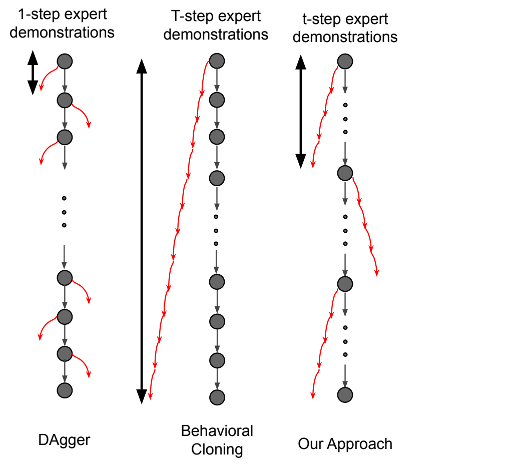

1-step expert demonstrations
T-step expert demonstrations
t-step expert demonstrations
DAgger
Behavioral Cloning
Our Approach

Figure 1. DAgger (Ross et al., 2011) collects 1-step feedback at every state. Behavioral cloning collects T-step feedback only from the initial state. Our method collects t-step feedback every t states.

Various solutions to the problem of distributional shift have been proposed (Daumé et al., 2009; Ross & Bagnell, 2010; Ross et al., 2011; Laskey et al., 2017). Some of these methods reduce the effects of distributional shift by interactively querying the expert (Daumé et al., 2009; Ross et al., 2011). DAgger (Ross et al., 2011), one of the most widely used of these solutions, queries the expert on every state that is visited by the policy and uses expert actions to improve the policy. There are other offline imitation learning approaches that do not require interactive queries. Inverse reinforcement learning (Abbeel & Ng, 2004; Ziebart et al., 2008) attempts to recover the reward function. Distributional matching approaches optimize for the agreement of occupancy measures (Ho & Ermon, 2016; Kostrikov et al., 2019). In this work, we focus on interactive imitation learning.

Normally, interactive imitation learning requires a predefined expert policy for feedback. Recently, several approaches have been proposed to relax this constraint (Silver et al., 2017; Anthony et al., 2017; Sun et al., 2018) by using look-ahead planning to generate feedback. We plan with Monte Carlo Tree Search (MCTS) in our experiments.

3. The POLISH Algorithm

We first present a general framework for imitation learning from local trajectory improvements in Section 3.1. We then describe a method based on MCTS to perform the proposed local trajectory optimization in Section 3.2.

3.1. Main Algorithm

Our main algorithm builds on the observation that behavioral cloning (Pomerleau, 1991) and DAgger (Ross et al., 2011) are two extremes along the spectrum of possible algorithms using experts to generate trajectory improvement.

Algorithm 1 POLISH (Policy Optimization by Local Improvement through Search)

```
 1: Input: an initial policy $\pi_1$, the expert policy $\pi^*$, the segment
    length $t$, number of iterations $N$, a policy class $\Pi$.
 2: for $1 \leq i \leq N$ do
 3:   Sample $\{\tau_j\}_{j=1}^n$ from $\pi_i$
 4:   $S \leftarrow \emptyset$
 5:   $D \leftarrow \emptyset$
 6:   for each trajectory $\tau_j = (s_0, a_0, s_1, \cdots, s_{T-1}, a_{T-1}, s_T)$
      do
 7:     $k \leftarrow \lceil T/t \rceil - 1$
 8:     $S_j \leftarrow \{s_0, s_t, s_{2t}, \cdots, s_{kt}\}$
 9:     $S \leftarrow S \cup S_j$
10:   end for
11:   for each $s \in S$ do
12:     Run $\pi^*$ for $t$ steps from the state $s$ to generate a partial
        trajectory $\tau_s = (s, a'_0, s'_1, \cdots, a'_{t-1}, s'_t)$
13:     $D \leftarrow D \cup \{(s, a'_0), (s'_1, a'_1), \cdots, (s'_{t-1}, a'_{t-1})\}$
14:   end for
15:   $\pi_{i+1} \leftarrow \arg\min_{\pi \in \Pi} L(D, \pi)$
16: end for
17: Return $\pi_{N+1}$
```

This is illustrated in Figure 1. The solid nodes represent the states visited by a current policy π during a rollout of length T. DAgger collects expert feedback for every state along the trajectory τ, while behavioral cloning only imitates a single complete trajectory from the expert. There is a spectrum of algorithms that lie in between, parametrized by the time horizon t to collect expert demonstrations and the frequency with which to choose the start state for the expert policy.

We present our main algorithm in Algorithm 1 that collects locally improved partial trajectories, as shown in Figure 1 and trains a policy to imitate the expert on the collected trajectories. For each iteration, we first collect trajectories from a current policy π_i (Line 3). Then, we collect every kth state from these trajectories as starting states for running the expert policy (Line 7, 8, 9). From each state collected, we rollout π^* for t steps to generate a partial improved trajectory (Line 12, 13). Once all the data has been generated, we optimize for the behavioral cloning loss $L(D, \pi) = \frac{1}{|D|} \sum_{(s,a^*) \in D} (I(\pi(s) \neq a^*))$, i.e., the supervised learning loss from data collected in D.

3.2. MCTS as the Expert Policy

General imitation learning requires an expert policy. We follow a series of recent works that apply MCTS to provide feedback for policy improvement (Guo et al., 2014; Silver et al., 2017; Anthony et al., 2017). Specifically, for every iteration, π^* becomes the policy obtained by running MCTS with π_i, following a variant of MCTS that uses the following UCT rule to select the leaf node in MCTS: $\text{UCT}(s, a) = \frac{r(s,a)}{n(s,a)} + c \cdot \pi(a|s) \sqrt{\frac{\log n(s)}{n(s,a)}}$, where $n(s)$ is the number of times the node s has been visited so far, $n(s, a)$ is the number of times action a is selected at node s, and

$r(s, a)$ is the sum of all rewards obtained in simulations by taking a in s. At a leaf node, Monte Carlo rollouts for value estimation (Browne et al., 2012) is replaced by an estimate obtained from a value network trained with the policy.

4. Theoretical Analysis

In this section, we provide theoretical justification for our local trajectory improvement in relation to both behavioral cloning and DAgger. We provide proofs in the Appendix.

We define quantities related to the state distributions induced by policies at different time steps. Let d_π^t denote the state distribution obtained by following π for t steps. We use $d_\pi = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_\pi^t$ to denote the discounted state distribution. With these quantities, we can rewrite $J(\pi) = \sum_{t=0}^{\infty}\mathbb{E}_{s\sim d_\pi^t, a\sim\pi(s)}[\gamma^t r(s,a)] = \mathbb{E}_{s\sim d_\pi, a\sim\pi(s)}[r(s,a)]$.

4.1. Motivation for Local Trajectory Improvement

We first make some assumptions on the quality of the expert policy π^*: (1) $J(\pi^*) \geq J(\pi)$; (2) one-step deviation from π^* according to π will result in lower-valued states if we were to follow π^* from that point on.

To motivate the idea of using local trajectory improvement instead of complete trajectory imitation learning, we extend the analysis in (Ross & Bagnell, 2010) on behavioral cloning. A major issue with (long-horizon) behavioral cloning is the potential for cascading errors if a learned policy makes a mistake early on. The cascading error manifests as a quadratic term of the time horizon T, $J(\pi) \geq J(\pi^*) - T^2\epsilon$, where T is the task horizon and $\epsilon = \mathbb{E}_{s\sim d_{\pi^*}}[I(\pi(s) \neq \pi^*(s))]$ is the error rate.

We can do a more careful analysis that motivates the design of an algorithm that clones behaviors at a shorter time scale. Define $J_\pi^{t_i:t_{i+1}}(\pi')$ to be the rewards obtained by following π' from time step t_i to t_{i+1} from the start distribution $d_\pi^{t_i-1}$. Assume we divide the time horizon into equal parts with length t so that we have T/t segments with $t_i = it$.

$$\begin{aligned} J(\pi) &= \sum_{i=1}^{T/t}\gamma^{t(i-1)}J_\pi^{t_i:t_{i+1}}(\pi) \\ &\geq \sum_{i=1}^{T/t}\gamma^{t(i-1)}(J_\pi^{t_i:t_{i+1}}(\pi^*) - t^2\epsilon_i) \\ &= \sum_{i=1}^{T/t}\gamma^{t(i-1)}J_\pi^{t_i:t_{i+1}}(\pi^*) - \frac{1-\gamma^T}{1-\gamma^t}t^2\epsilon \qquad (1) \end{aligned}$$

where ϵ_i is the error rate computed between steps t_i and t_{i+1}. For simplicity, assume $\epsilon_i = \epsilon$ holds across different segments, which gives us Equation 1.

The objective is to maximize $J(\pi)$ and this is done by optimizing the lower bound on the right hand side via imitating π^*. Next, we consider how the two terms on the right hand side vary as the time horizon t changes.

Proposition 1. *Under the assumptions, $\sum_{i=1}^{T/t}\gamma^{t(i-1)}J_\pi^{t_i:t_{i+1}}(\pi^*) \leq \sum_{i=1}^{T/t'}\gamma^{t'(i-1)}J_\pi^{t'_i:t'_{i+1}}(\pi^*)$ if t' is a multiple of t. Furthermore, it is maximized at $t = T$.*

This proposition says that by multiplying the length of a segment, the first term, which models the improvement from utilizing an expert policy, grows. Next we consider the second term, simple algebraic calculation yields that $\frac{1-\gamma^T}{1-\gamma^t}t^2\epsilon$ is an increasing function of $t > 0$ for $\gamma \in (0, 1)$. Since the second term is also an increasing function in t, there can be a balance point where the right hand side is maximized, giving the tightest lower bound.

5. Experiment

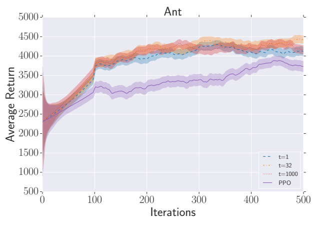
The goals of our experimental evaluation are threefold: (1) to verify that there exists a balance point for optimized length for local trajectory improvement, (2) to show that there is a trade-off between reward improvement by MCTS and the divergence of action distributions between the policy and the expert (MCTS) as proved in the theoretical analysis, (3) to demonstrate that the POLISH algorithm, by design, enables a significant speedup with a parallel implementation that is simple and efficient.

5.1. Experiment Setup

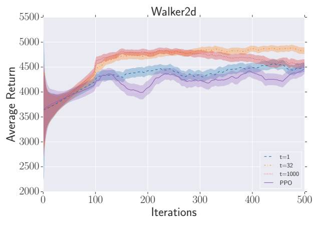
We evaluate on four MuJoCo (Todorov et al., 2012) environments (Ant, Walker2d, Hopper and Humanoid). For each environment, we use the standard maximum trajectory length of 1000. For each trajectory, there are multiple MCTS rollouts starting from every t states, each of which is distributed on a separate machine. This implementation takes advantage of the parallelizable nature of Lines 11, 12 in Algorithm 1. For instance, with a trajectory length of 1000 and a segment length of $t = 32$, we can achieve maximum parallelization by using 32 machines. For MCTS to provide meaningful improvement, we pre-train each policy with 1000 iterations of PPO (Schulman et al., 2017) and use it to guide the MCTS described in Section 3.2. Our MCTS implementation is based on (MiniGo, 2020).

5.2. Main Results

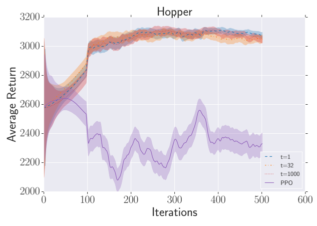
Figure 2 compares POLISH with different segment lengths: $t = 1000$ (behavioral cloning), $t = 32$ and DAgger (Ross et al., 2011), which can be regarded as $t = 1$ with dataset aggregation across iterations, as well as the vanilla PPO. Overall, MCTS-driven imitation learning achieves a higher return compared to PPO in all three environments, and by a significant margin with the best choice of segment length in particular. Regarding the choice of segment lengths, a



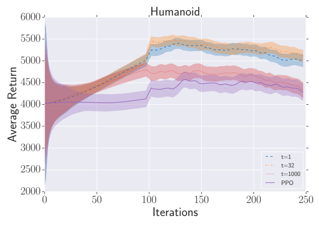
(a) Ant



(b) Walker2d

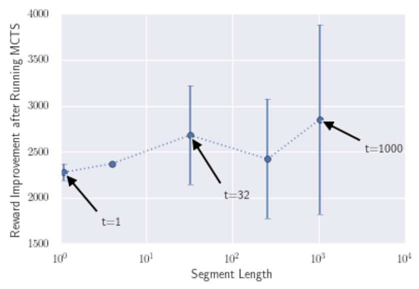


(c) Hopper

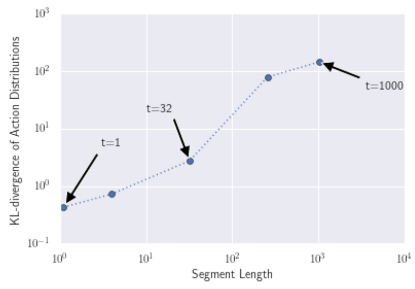


(d) Humanoid

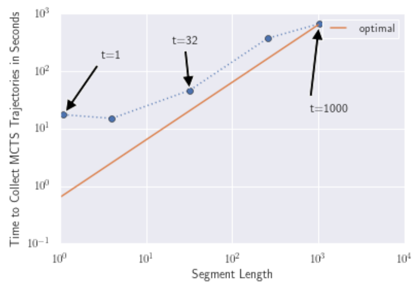
Figure 2. Average returns for continuous control tasks with different t values and the PPO baseline. Experiment results are across 5 random seeded runs. Shaded area indicates ± 1 standard deviation.



(a) The MCTS improvement becomes higher as segment length increases (as suggested by Proposition 1), but the variance also grows.



(b) The KL-divergence of action distributions between policy and MCTS grows as segment length grows.



(c) Wall clock time to generate MCTS trajectories in parallel increases as the segment length increases.

Figure 3. Tradeoff between t and improved rewards from MCTS, KL-divergence, time to generate data from MCTS in the Ant environment.

length of $t = 32$ is almost always better than $t = 1$. It is sometimes matched with $t = 1000$ (Figure 2a, 2c). In Section 5.4, we show that in those cases using $t = 32$ still has advantage in computation time. This shows that an interpolated version between BC and DAgger provides a better trade-off between the two terms in Equation 1.

5.3. Empirical Evidence of Theory behind POLISH

To further understand the impact of segment length on the return, we show the reward improvement after running MCTS with the current policy in Figure 3a, and the KL-divergence of the action distributions between the MCTS and the policy in Figure 3b. Figure 3a shows how the first term in Equation 1 changes as a function of segment length, and 3b depicts an approximation of the error term ϵ in the second term. For the Ant environment, we show both metrics for a select range of segment lengths of (1, 4, 32, 256, 1000). As shown in Proposition 1, the expert (MCTS) improvement (over the policy) generally increases with segment length. The KL-divergence, which can be viewed as an approximation to the ϵ term is growing with t. As a result, the second term in Equation 1 is growing as a function of t as well. Both observations validate our theoretical analyses.

5.4. Parallel Implementation Speedup

Lastly, we show that we are able to achieve a significant speedup with a parallel implementation. Our algorithm, by design, is easy to parallelize over distributed compute (Lines 11, 12 in Algorithm 1). Every MCTS rollout from an initial state is independent, and thus can be conducted on separate workers. Figure 3c shows the actual runtime of the parallel version of MCTS rollouts over a trajectory, compared to the ideal lower-bound which is the runtime for one single MCTS rollout at a segment length t. Even with a small segment length, e.g., $t = 32$, which implies with a high number of parallel tasks (=32) over a fixed trajectory length (=1000), the actual runtime is close to ideal. This suggests that we can easily achieve a 10-times speedup for OpenAI Gym-like environments.

6. Conclusion

In this work, we propose POLISH, a general imitation learning algorithm that provides flexible local trajectory optimization based on MCTS. We provide theoretical analysis that sheds light on the benefit of local trajectory improvement. We provide empirical support for our theory by demonstrating strong empirical performance on a suite of high-dimensional continuous control problems.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, pp. 5360–5370, 2017.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Daumé, H., Langford, J., and Marcu, D. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Guo, X., Singh, S., Lee, H., Lewis, R. L., and Wang, X. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems*, pp. 3338–3346, 2014.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573, 2016.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*, 2017.

MiniGo. Minigo: A minimalist go engine modeled after alphago zero, built on mugo, 2020. URL https://github.com/tensorflow/minigo.

Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., Norouzi, M., Bengio, S., and Dean, J. Device placement optimization with reinforcement learning. *ICML*, 2017. URL http://arxiv.org/abs/1706.04972.

Mirhoseini, A., Goldie, A., Pham, H., Steiner, B., Le, Q. V., and Dean, J. A hierarchical model for device placement. *ICLR*, 2018.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Bae, S., Nazi, A., Pak, J., Tong, A., Srinivasa, K., Hang, W., Tuncer, E., Babu, A., Le, Q. V., Laudon, J., Ho, R., Carpenter, R., and Dean, J. Chip placement with deep reinforcement learning, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pp. 305–313, 1989.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668, 2010.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

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

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Singh, A., Yang, L., Hartikainen, K., Finn, C., and Levine, S. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

Stadie, B. C., Abbeel, P., and Sutskever, I. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

Sun, W., Gordon, G. J., Boots, B., and Bagnell, J. Dual policy iteration. In *Advances in Neural Information Processing Systems*, pp. 7059–7069, 2018.

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

Zhou, Y., Roy, S., Abdolrashidi, A., Wong, D., Ma, P. C., Xu, Q., Zhong, M., Liu, H., Goldie, A., Mirhoseini, A., and Laudon, J. Gdp: Generalized device placement for dataflow graphs, 2019.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. pp. 1433–1438, 2008.

A. Proofs from Section 4

Proof of Proposition 1

Proof. Remember the two assumptions we make about the expert policy π^*: (1) $J(\pi^*) \geq J(\pi)$; (2) one-step deviation from π^* according to π will result in lower-valued states if we were to follow π^* from that point on.

Let $t' = kt$ for some positive integer k. Since Algorithm 1 collects initial state every t and t' steps, respectively. The set of starting states collected with the interval t' is a subset of those collected with an interval of t. We will show that for the first segment of length t' starting from some state s, $J_\pi^{0:t'}(\pi^*) \geq \sum_{i=1}^k \gamma^{t(i-1)} J_\pi^{t_i:t_{i+1}}(\pi^*)$. Similar proof holds for the other segments as well.

$$J_\pi^{0:t'}(\pi^*) = \sum_{i=1}^{k} \gamma^{t(i-1)} J_{\pi^*}^{t_i:t_{i+1}}(\pi^*) \tag{2}$$

$$\geq \sum_{i=1}^{k} \gamma^{t(i-1)} J_{\pi}^{t_i:t_{i+1}}(\pi^*) \tag{3}$$

The inequality holds term-by-term because by Assumption (2), following π instead of π^* generates state distributions that induce a lower valued states.

To show that the sum is maximized at $t = T$, consider any $t < T$.

$$J_\pi^{0:T}(\pi^*) = \sum_{i=1}^{T/t} \gamma^{t(i-1)} J_{\pi^*}^{t_i:t_{i+1}}(\pi^*) \tag{4}$$

$$\geq \sum_{i=1}^{T/t} \gamma^{t(i-1)} J_{\pi}^{t_i:t_{i+1}}(\pi^*) \tag{5}$$

It follows that the sum is maximized at $t = T$. □

Proposition 2. $\frac{1-\gamma^T}{1-\gamma^t} t^2 \epsilon$ *is an increasing function of* $t > 0$ *for* $\gamma \in (0, 1)$.

Proof. Let $f(t) = \frac{t^2}{1-\gamma^t}$. To prove this proposition, it suffices to show that $f(t)$ is an increasing function for $t > 0$. First we take the derivative and get $f'(t) = \frac{2t(1-\gamma^t)+t^2\gamma^t \ln\gamma}{(1-\gamma^t)^2}$.

Let $g(t) = 2t(1-\gamma^t) + t^2\gamma^t \ln\gamma$ be the numerator term. It is sufficient to show that $g(t) > 0$ for $t > 0$. Taking the derivative for $g(t)$, we obtain

$$\begin{aligned} g'(t) &= 2 - 2\gamma^t - 2t\gamma^t \ln\gamma + \ln\gamma(2t\gamma^t + t^2\gamma^t \ln\gamma) \\ &= 2 - 2\gamma^t + (\ln\gamma)^2 t^2 \gamma^t \\ &> 0 \end{aligned}$$

the last inequality follows from $\gamma \in (0, 1)$. So $g(t)$ is increasing, i.e., $g(t) > g(0) = 0$. It follows that $f'(t) > 0$ and $f(t)$ is an increasing function of t. □