Influence-Augmented Local Simulators
a Scalable Solution for Fast Deep RL in Large Networked Systems

Suau, Miguel; He, Jinke ; Spaan, Matthijs T.J.; Oliehoek, Frans A.

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 39th International Conference on Machine Learning

Citation (APA)
Suau, M., He, J., Spaan, M. T. J., & Oliehoek, F. A. (2022). Influence-Augmented Local Simulators: a Scalable Solution for Fast Deep RL in Large Networked Systems. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th International Conference on Machine Learning* (Vol. 162, pp. 20604-20624). (Proceedings of Machine Learning Research; Vol. 162). PMLR. https://proceedings.mlr.press/v162/suau22a.html

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Influence-Augmented Local Simulators: A Scalable Solution for Fast Deep RL in Large Networked Systems

Miguel Suau [1] **Jinke He** [1] **Matthijs T. J. Spaan** [1] **Frans A. Oliehoek** [1]

Abstract

Learning effective policies for real-world problems is still an open challenge for the field of reinforcement learning (RL). The main limitation being the amount of data needed and the pace at which that data can be obtained. In this paper, we study how to build lightweight simulators of complicated systems that can run sufficiently fast for deep RL to be applicable. We focus on domains where agents interact with a reduced portion of a larger environment while still being affected by the global dynamics. Our method combines the use of local simulators with learned models that mimic the influence of the global system. The experiments reveal that incorporating this idea into the deep RL workflow can considerably accelerate the training process and presents several opportunities for the future.

1. Introduction

The remarkable success of Deep Reinforcement Leaning (RL) on paper is in stark contrast with its narrow applicability to real-world problems. Among many other reasons, the most important factor preventing the practical deployment of this framework is perhaps its high sample complexity (Botvinick et al., 2019). This is a very well-known issue and there is a long list of previous works that in one way or another have tried to alleviate it (Kakade, 2003; Mnih et al., 2015; Ha & Schmidhuber, 2018). Nonetheless, there have been relatively few real-world successes thus far. Here, rather than proposing yet another method that tries to solve the problem directly, we present a more pragmatic approach to get around it. Our solution is based on the observation that Deep RL's best results have been obtained in domains like video games (Bellemare et al., 2013; Vinyals et al., 2019) or simulated environments (Brockman et al., 2016; Ganesh et al., 2019; Bellemare et al., 2020) where data collection is extremely fast. Unfortunately, real-world problems are typically more complex and simulators, if available, are usually very slow (Dulac-Arnold et al., 2019).

In this work, we design lightweight versions of large simulators with the goal of speeding up the overall training process. The method we propose applies to domains where agents only interact with a reduced local part of a larger environment, yet they are indirectly being affected by the global dynamics. Traffic control is one example of such environments. Say, for instance, that we wanted to train an agent to control the traffic lights of a particular intersection in a very large city. To do so we could build a small local simulator that captures only the information that is directly relevant to the agent (traffic density in the neighborhood; van der Pol & Oliehoek 2016). However, after training, we may find out that an agent that does very well in the small simulator, performs poorly in the real intersection. The performance gap would be caused by a data distribution shift (Quionero-Candela et al., 2009; Arjovsky, 2021). Even though the simulator might be able to closely mimic the local dynamics (i.e. cars moving within the intersection), it would fail to account for the interactions of the local neighborhood with the rest of the city. Thus, the agent learns a policy based on certain transition dynamics that turn out to be very different in the real world. Alternatively, we could try to model the dynamics of a sufficiently large portion of the city, but this would surely result in a very slow simulator.

One important property of the traffic domain is that, although the agent's local problem may be *affected* by many external variables (traffic densities in other parts of the city), it is only *directly influenced* by the road segments that connect the intersection with the rest of the city. Hence, we can simply monitor the traffic densities at these road segments since, from the agent's local perspective, they summarize the effect of all the external variables. This insight is not specific to the traffic domain. It is in fact common in networked systems (e.g. warehouse commissioning, Claes et al. 2017; electrical power grids, Wang et al. 2021; heating systems, Gupta et al. 2021; telecommunication networks, Suau et al. 2021) that interactions between different components occur through a limited number of variables.

[1]Delft University of Technology. Correspondence to: Miguel Suau <m.suadecastro@tudelft.nl>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Supported by the formal framework of *influence-based abstraction* (IBA) (Oliehoek et al., 2021), we exploit the above property to build local simulators that mirror the response of the global system through the so called *influence predictor*.

Contributions The main contribution of this paper is the integration of the IBA framework with the Deep RL workflow. Our experiments reveal, that the combination of local simulators and influence predictors can considerably accelerate the reinforcement learning process. We also demonstrate both theoretically and empirically that, under mild conditions, the memory needs of the influence predictor are fully determined by the agent's memory capacity. Moreover, we study the impact that distribution shifts caused by changes in the agent's policy may have on the influence predictor, and explore how to prevent the model from picking up on spurious correlations that are not invariant across policies. Finally, we investigate the effect of transfer learning and show how inaccurate simulators might also be able to render effective policies.

2. Related Work

The problem of sample complexity has been extensively studied by the RL community. Among many others, the most promising solutions are: replaying previous experiences to make more efficient use of the available data (Mnih et al., 2015; Schaul et al., 2016; van Hasselt et al., 2019), or learning surrogate models of the environment dynamics (Sutton, 1990; Ha & Schmidhuber, 2018; Schrittwieser et al., 2020; Moerland et al., 2020). Yet, these techniques are only effective when provided enough real samples. If not, replay buffers might not be sufficient to obtain good policies and surrogate models might generalize poorly. An alternative is to train agents with synthetic data coming from a simulator. However, most real-world scenarios are excessively complex and simulators, if available, are computationally expensive (Dulac-Arnold et al., 2019). Here we argue that building a simulator of the entire system is often unnecessary. In fact, as we explain in the following sections, in many situations we can get away by just modelling the dynamics around the agent's local neighborhood.

A few prior works have investigated the computational benefits of factorizing large systems into independent local regions (Nair et al., 2005; Varakantham et al., 2007; Kumar et al., 2011; Witwicki & Durfee, 2011). Unfortunately, since local regions are often coupled to one another, such factorizations are not always appropriate. Nonetheless, in many cases, the interactions between regions occur through a limited number of variables. Using this property, the theoretical work by Oliehoek et al. (2021) on influence-based abstraction (IBA) describes how to build influence-augmented local simulators (IALS) of local-FPOMDPs, which model only the variables in the environment that are directly relevant to the agent while monitoring the response of the rest of the system with the influence predictor. The problem is, the exact computation of the conditional influence distribution is intractable and we can only try to estimate it from data. Congeduti et al. (2021) provide theoretical bounds on the value loss when planning with approximate influence predictors. The work by He et al. (2020) has empirically demonstrated the advantage of this approach to improve the efficiency of online planning in two discrete problems.

Here, we extend the method to more realistic problems and study how to integrate the IBA framework with Deep RL. This has profound implications that do not arise in the planning context, namely the relation between the agent's memory capacity and the history dependence of the influence predictor (Section 4.1), and the problem of off-policy generalization (Section 4.2). Moreover, while He et al. showed that the IALS outperforms the global simulator only when the time budget is limited, our results reveal that the IALS can train policies in a fraction of the time and that these can match the same performance as policies trained on the GS, without imposing any time constraints, and despite the IALS being only approximate.

3. Preliminaries

In this section, we introduce the notation used throughout the paper, provide the mathematical definition of the problem, and describe the Influence-based abstraction (IBA) framework (Oliehoek et al., 2021), which gives theoretical support to the method we introduce in Section 4.

3.1. Problem definition

As explained in the introduction, we target domains where the agent, although affected by the global dynamics, can only observe its local neighborhood. These can be modelled as factored partially observable Markov decision processes (FPOMDP) (Kaelbling et al., 1996; Boutilier et al., 1999).

Definition 1 (FPOMDP)**.** A factored partially observable Markov decision process (FPOMDP) is a tuple $\langle S, A, T, R, \Omega, O \rangle$ where S is the set of k state variables $S = \{S^1, ..., S^k\}$, such that every state $s \in \times_{i=1}^{k} S^i$ is a vector $s = \langle s^1, ..., s^k \rangle$, A is the set of actions, T is the transition probability function $T(s_{t+1}|s_t, a_t)$, $R(s_t, a_t)$ defines the immediate reward, Ω is the observation space, and O is the observation function, $O(o_{t+1}|s_{t+1})$.

The task consists in finding a policy π that maximizes the expected discounted sum of rewards (Sutton & Barto, 1998). Since the agent receives only a local observation o of the true state s, a policy that is based only on the most recent information can be sub-optimal (Singh et al., 1994; McCallum, 1995). In general, the agent is

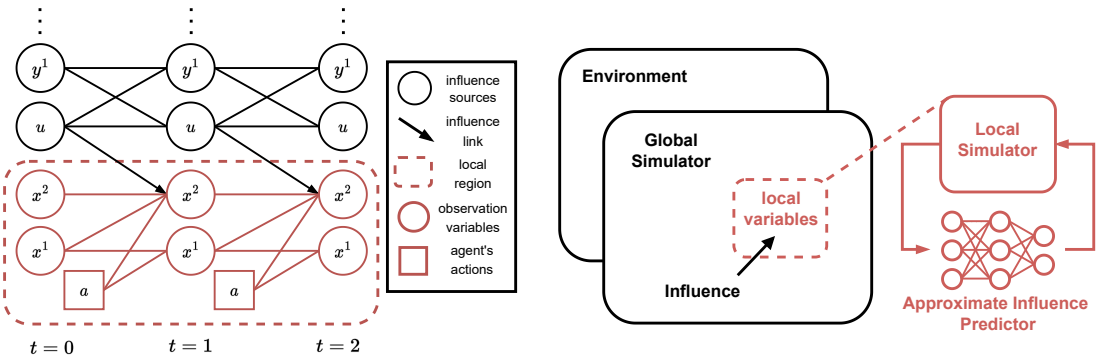

Figure 1. Left: A Dynamic Bayesian Network of a Local-FPOMDP unrolled 3 timesteps. Right: A diagram of the IALS: the GS is a full-scale representation of the environment that models the global dynamics. In contrast, the IALS only models the local dynamics and is equipped with an AIP that monitors the response of the global system to a given ALSH.

required to keep track of its past actions and observations to make the right action choices. Policies are therefore mappings from the past action-observation history (AOH), $h_t = \langle o_1, a_1 ..., a_{t-1}, o_t \rangle$, to actions, $\pi(a_t|h_t)$.

We consider, in particular, a special case of FPOMDP hereinafter referred to as *Local-FPOMDP*.

Definition 2 (Local-FPOMDP)**.** A Local-FPOMDP is an FPOMDP where O and R depend only on a subset of state variables $X = \{X^1, ..., X^j\} \subseteq S$ with $j \leq k$, such that the agent's local states $x \in \times_{i=1}^{j} X^i$ are vectors $x = \langle x^1, ..., x^j \rangle$. Hence, we have that $O(o_{t+1}|s_{t+1}) = \dot{O}(o_{t+1}|x_{t+1})$ and $R(s_t, a_t) = \dot{R}(x_t, a_t)$, where $\dot{O}$ and $\dot{R}$ are the local observation and local reward functions.

Looking at the above definition, one can argue that simulating the transitions $T(s_{t+1}|s_t, a_t)$ of the full set of state variables is unnecessary, and while doing so is possible in small problems, it might become computationally intractable in large domains. We can instead define a new transition function $\bar{T}$ that models only the local state variables x, $\bar{T}(x_{t+1}|x_t, a_t)$. The problem is that x_{t+1} may still depend on the rest of state variables s_t and thus $\bar{T}$ is not well defined. Hence, we are forced to keep track of the action-local-state history (ALSH) $l_t = \langle x_1, a_1 ..., a_{t-1}, x_t \rangle$ such that we can sample the next local state as

$$P(x_{t+1}|l_t, a_t) = \sum_{s_t} T(x_{t+1}|s_t, a_t) P(s_t|l_t). \quad (1)$$

3.2. Influence-based Abstraction

With the above formulation, the problem is only partially solved. We no longer need to simulate the global state transitions, yet we have to maintain a distribution over the full state given the ALSH, $P(s_t|l_t)$, and then calculate the local transitions with $P(x_{t+1}|s_t, a_t)$. As explained in the introduction, in many local-FPOMDP problems, however, only a fraction of the state variables will *directly influence* the local region.

The diagram in Figure 1 is a Dynamic Bayesian Network (DBN) (Pearl, 1988; Boutilier et al., 1999) of a local-FPOMDP prototype. The agent's local region, corresponds to the variables, denoted by $x \in X$, that lie within the red box, $x = \langle x^1, x^2 \rangle$. The diagram also shows the non-local variables, known as influence sources $u \in U \subseteq S \setminus X$, that influence the local region directly. The three dots on the top indicate that there can be, potentially many, other non-local variables in S. These are denoted by y and, as shown in the diagram, can only influence x via u. As such, given u_t, x_{t+1} is conditionally independent of y_t, $P(x_{t+1}|x_t, u_t, y_t) = P(x_{t+1}|x_t, u_t)$.

Definition 3 (IALM)**.** An influence-augmented local Model (IALM) is a FPOMDP with local states $x \in \times_{i=1}^{j} X^i$, influence sources $u \in \times_{i=1}^{k} U^i$ with $U \subseteq S \setminus X$, local observation function $\dot{O}(o_{t+1}|x_{t+1})$, local reward function $\dot{R}(x_t, a_t)$, local transition function $\dot{T}(x_{t+1}|x_t, u_t, a_t)$ and an influence distribution $I(u_t|l_t)$.

Using the IALM we can simulate the local transitions as

$$P(x_{t+1}|l_t, a_t) = \sum_{u_t} \dot{T}(x_{t+1}|x_t, u_t, a_t) I(u_t|l_t). \quad (2)$$

Note that, as opposed to the Local-FPOMDP, the transition function $\dot{T}$ in the IALM is defined purely in terms of the local state variables and the influence sources. Moreover, the influence distribution is just the conditional probability over u_t instead of the full set of of state variables s_t. All in all, this translates into a much more compact, yet *exact* representation of the problem (Oliehoek et al., 2021), which should be computationally much lighter than the original FPOMDP.

4. Influence-Augmented Local Simulators for Deep RL

In the following, we describe how we make use of the IALM formulation to design lightweight simulators that can

speed up the long training times imposed by neural network policies. Figure 1 (right) shows a diagram of the influence-augmented local simulator (IALS), which is composed of a *local simulator* and an *approximate influence predictor*.

Local simulator (LS): The LS is an abstracted version of the environment that only models a small portion of it. As opposed to a global simulator (GS), which should closely reproduce the dynamics of every state variable, the LS focuses on characterizing the transitions of those variables x_t that the agent directly interacts with, $\dot{T}(x_{t+1}|x_t, u_t, a_t)$. Recall that, in our setting, both the reward R and observation O functions are defined in terms of x_t and a_t.

Approximate influence predictor (AIP): The AIP monitors the interactions between the local region and the external variables y_t by estimating $I(u_t|l_t)$. Ideally, we would like the approximation to match the true influence distribution. However, due to combinatorial explosion, computing the exact probability $I(u_t|l_t)$ is generally intractable (Oliehoek et al., 2021), and so we can only try to approximate it using, for instance, a parametric function. We write $\hat{I}_\theta$ to denote the AIP, where θ are the parameters, which need to be learned from data. Replacing the true influence distribution with an approximation implies that we are no longer guaranteed to find the optimal policy (Congeduti et al., 2021). Nonetheless, as we show in our experiments, it is often worth trading accuracy for computational efficiency. We model $\hat{I}_\theta$ using a neural network, which we train on a dataset of N samples of the form (l_n, u_n) collected from the GS (see Algorithm 1 in Appendix G). Since role of the AIP is to estimate the conditional probability of the influence sources u_t given the past ALSH, we can formulate the task as a classification problem.[1] The neural network is optimized using the expected cross-entropy loss,

$$L(\hat{I}_\theta) = -\frac{1}{N}\sum_{n=0}^{N} \log \hat{I}_\theta(u_n|l_n), \tag{3}$$

which minimizes the KL divergence between the true probabilities $I(u_t|l_t)$ and those predicted by $\hat{I}_\theta(u_t|l_t)$.[2] Once the network has been trained, we can simulate trajectories with the IALS (Algorithm 2 in Appendix G). These trajectories can then be used to train policies with any standard Deep RL method (Mnih et al., 2015; Schulman et al., 2017).

In the next two sections we discuss two important considerations when training AIPs for the RL setting. We first show that, since the agent's memory is inevitably finite, we normally will not need to train complicated AIPs that condition on the full ALSH (Section 4.1). In Section 4.2, we study the impact that distribution shifts caused by changes in the agent's policy may have on the AIP, and explore how to prevent the AIP from picking up on spurious correlations that are not invariant across policies.

4.1. Finite memory agents and AIP history dependence

As mentioned before, we can only rely on approximations of $I(u_t|l_t)$ since computing the exact distribution that conditions on the full ALSH is intractable. Unfortunately though, even with the most sophisticated RNNs (Hochreiter & Schmidhuber, 1997; Cho et al., 2014), learning long term temporal dependencies is also very difficult. However, it is worth noting that, if capturing long term dependencies is hard for the AIP, it is too for the agent. In fact, it is very common in Deep RL to find policies that have access only to finite memories (Mnih et al., 2015; Oh et al., 2016) or that are trained on short sequences (Schmidhuber, 1991; Hausknecht & Stone, 2015). Thus, if the agents memory is finite, one might well wonder whether the extra level of accuracy that an influence predictor which conditions on the full ALSHs provides is needed. The result below shows that, the history dependence of the influence predictor is, under mild conditions, determined by the agent's memory capacity.

Theorem 1. *Let $l_{t-k:t}$ be a truncated version of l_t containing only the last k action-local-state pairs and let $a_{0:t-k}$ be the sequence of past actions from time 0 up to k. Let the agent's policy $\bar{\pi}$ be a function of $h_{t-k:t}$ $\bar{\pi}(a_t|h_{t-k:t})$, where $h_{t-k:t}$ is also a truncated version of h_t. If for all u_t we have $P(u_t|l_{t-k:t}, a_{0:t-k}) = P(u_t|l_{t-k:t})$, then an influence predictor that conditions only on $l_{t-k:t}$, $\bar{I}(u_t|l_{t-k:t})$, is sufficient to guarantee no loss in value.*[3]

Proof. Found in Appendix A.

Intuitively, a finite memory agent whose policy conditions on the last k elements in the AOH is only be capable of computing an expectation over the action-values given these k elements. In turn, the distribution of u_t given the full ALSH l_t, $I(u_t|l_t)$, is effectively washed away by the same expectation. The condition $P(u_t|l_{t-k:t}, a_{0:t-k}) = P(u_t|l_{t-k:t})$ is needed for $P(u_t|l_{t-k:t})$ to be well-defined independently of the policy. Intuitively, this just means that $a_{0:t-k}$ cannot have any long-term effects on the influence distribution such that $l_{t-k:t}$ contains enough information to predict u_t (See proof in Appendix A for more details). The upshot is that the agent's memory capacity limits the temporal dependen-

[1] Note that we assume for simplicity that the influence sources U are discrete variables. However, our approach can generalize to continuous variables by formulating the AIP learning problem as a regression rather than a classification problem.

[2] Please refer to Appendix F for more details on the implementation of the AIP.

[3] Note that this refers only to the value of polices of the form $\bar{\pi}(a_t|h_{t-k:t})$, which might perform arbitrarily worse than policies that condition on the full AOH, $\pi(a_t|h_t)$ (Singh et al., 1994).

cies of the influence predictor,[4] meaning that I may as well condition directly on $l_{t-k:t}$ rather than l_t. This insight is empirically evaluated in Section 5.4.

4.2. Off-policy generalization and d-sets

Given that the *true influence distribution* conditions on the full ALSH, it is, in principle, independent of the exploratory policy π_0 that we use to collect the data. Indeed, if we would represent $I(u_t|l_t)$ with a table, we could compute unbiased estimates of the true probabilities $I(u_t|l_t)$ by using any arbitrary policy that visits every possible ALSH. The problem arises when we use function approximators to estimate the influence sources. In particular, if we call $P^{\pi_0}(l_t, u_t)$ the stationary joint distribution of influence sources and ALSHs induced by the exploratory policy π_0 that we use to collect the data $D^{\pi_0} = \{(l_1, u_1), ...(l_N, u_N)\}$ from the GS (as described by Algorithm 1 in Appendix G), we can write

$$\theta^* = \arg\min_\theta L(\hat{I}_\theta, D^{\pi_0}), \tag{4}$$

where we see that, in fact, the loss in (3) depends on D^{π_0}. Therefore, because we are fitting $\hat{I}_\theta$ to D^{π_0}, the model will be biased towards P^{π_0}. This is not so bad considering that we want the influence predictions to be more reliable for those ALSHs that the agent visits more often. However, it can pose a problem when the policy π that we train starts deviating from π_0. In general, we want the AIP to perform well on any distribution that the agent may experience during training, $\{P^\pi\}_{\pi\in\Pi}$, where Π denotes the set of possible policies. We then rewrite the optimization problem in (4) as

$$\theta^* = \arg\min_\theta \max_{\pi\in\Pi} L(\hat{I}_\theta, D^\pi) \tag{5}$$

to indicate that we wish to minimize the worst-case loss.

The key issue here is that, before training, we have only access to the exploratory policy π_0. Equation (5) is an instance of a well-studied problem in the field of out-of-distribution generalization (Quionero-Candela et al., 2009; Arjovsky, 2021). In principle, according to Arjovsky (2021, Section 3.6.1), finding a solution to (5) by minimizing (4) requires, **(i)** $\text{supp}(P^\pi) \subseteq \text{supp}(P^{\pi_0})$, **(ii)** infinite number of samples, and **(iii)** infinite capacity models.

The first condition is met so long as $\pi_0(a_t|l_t) > 0$ for all a_t and l_t. The second and third conditions, on the other hand, can never be fulfilled in practice. On top of that, high-capacity influence predictors are particularly undesirable for our purpose because they are computationally demanding. The consequence of training low-capacity models on finite datasets is that they may pick-up on spurious correlations (Pearl, 2000) between ALSHs and influence sources.[5] These correlations could be just an artifact of π_0 and may disappear after the policy is updated. One way to prevent the above, is to find a representation of l_t that elicits an invariant predictor of u_t across all P^π (Peters et al., 2016; Arjovsky et al., 2019; Krueger et al., 2021).

Definition 4 (invariant predictor)**.** A subset of variables d_t from l_t elicit an invariant predictor of u_t across policies $\pi \in \Pi$ if for all d_t in the intersection of supports, $\text{supp}(P^{\pi_1}(d, u)) \cap \text{supp}(P^{\pi_2}(d, u))$, we have

$$P^{\pi_1}(u_t|d_t) = P^{\pi_2}(u_t|d_t) \quad \text{for all} \quad \pi_1, \pi_2 \in \Pi. \tag{6}$$

The notion of d-separation (Bishop, 2006) comes in handy here. Given a DBN, such as the one depicted in Figure 1, one can determine a subset of variables in the ALSH that is sufficient to predict u_t (Oliehoek et al., 2021).

Definition 5 (d-separating set)**.** The d-separating set (d-set) is a subset of variables d_t from l_t, such that influence sources u_t and the remaining parts of the ALSH $l_t \setminus d_t$ are conditionally independent given d_t, $(u_t \perp\!\!\!\perp l_t \setminus d_t \mid d_t)$. The d-set is said to be minimal d_t^*, when no more variables can be removed from d_t^* while the above still holds.

Theorem 2. *A subset of variables d_t from l_t is guaranteed to elicit an invariant predictor of u_t, across all $\pi \in \Pi$, if (i) d_t constitutes a d-separating set and (ii) all policies are functions of the local AOH, $\pi(h_t)$.*

Proof. Found in Appendix A.

Note that, according to the result above, the full ALSH l_t does elicit an invariant predictor of u_t since it is, by definition, a d-set. Hence, feeding l_t should be sufficient for the model to find stable and invariant correlations. The problem, however, is that in practice, models tend to converge to low-capacity solutions that require little "effort" to learn (i.e. least-norm solution) (Arjovsky, 2021). As such, the representations formed by an influence predictor that is fed the full ALSH may or may not constitute a d-set. The solution we propose is to feed solely a minimal d-set d_t^* into the AIP $\hat{I}_\theta$. Not only does this reduce the dimensionality of the input space but also prevents the AIP from learning shortcuts (Geirhos et al., 2020) resulting from spurious correlations that are not stable under policies other than π_0[5]. Provided we have the DBN of the problem, there are many algorithms available that can help us find a minimal d-set (Acid & De Campos, 1996; Tian et al., 1998). If not, some domain knowledge may be sufficient in most cases, to remove a few variables from l_t and prevent confounding (Suau et al., 2022).

Working in the joint space (d_t, u_t) instead of (l_t, u_t) has a another positive effect on off-policy generalization. Since l_t includes the agents actions the distribution $P^{\pi_0}(l, u)$ might be arbitrarily far from a distribution in $\{P^\pi(l, u)\}_{\pi\in\Pi}$. This

[4] More details on the practical implications of this result are given in Appendix F.

[5] A visual example of a spurious correlation appearing in the traffic problem is given in Appendix B.

is problematic because our low-capacity influence predictor may be unable to generalize well under large distribution shifts.

Proposition 1. *Let π_0 be the exploratory policy used to collect the dataset. Then, for all $\pi \in \Pi$*

$$KL(P^{\pi_0}(d_t, u_t)||P^{\pi}(d_t, u_t)) \leq KL(P^{\pi_0}(l_t, u_t)||P^{\pi}(l_t, u_t))$$

where KL is the Kullback–Leibler divergence.

Proof. Found in Appendix A.

The result above shows that the distributions of pairs (d^*, u) induced by two policies in Π are never further apart than their full AOHs counterparts. In fact, if actions and d-sets are only weakly-coupled the distributions may be very close. In the extreme case, if d-sets are causally independent with respect to the agent's actions, the influence sources can be considered exogenous (Boutilier et al., 1996; Chitnis & Lozano-Pérez, 2020) and the two distributions are equivalent.

Corollary 1. *Let $a_{0:t}$ be the past sequence of actions from 0 to t. If $P(d_t|do(a_{o:t})) = P(d_t)$ for all d_t and $a_{0:t}$. Then, for any $\pi_1 \neq \pi_2 : \pi_1, \pi_2 \in \Pi$ we have $P^{\pi_1}(d_t, u_t) = P^{\pi_2}(d_t, u_t)$ even though $P^{\pi_1}(l_t, u_t) \neq P^{\pi_2}(l_t, u_t)$.*

Proof. Found in Appendix A.

Of course, in some domains, certain instantiations of the influence sources may only occur when very specific action sequences are followed. In such cases, a finite dataset collected with a random uniform policy will not be sufficient to fully cover the joint space of influence sources and d-sets, and thus the AIP may be unable to generalize well. If this happens, assuming a GS is available during training, the obvious solution would be to retrain the AIP every certain number of policy updates. Nonetheless, as we explain in the next section, in many cases, it might not be worth paying the computational cost associated to this solution since inaccurate simulators might still provide good enough experiences to learn from.

4.3. Sufficiently similar training conditions

Up to this point we have discussed the importance of training accurate AIPs that are also invariant to distribution shifts caused by changes in the agent's policy. Nonetheless, here we pose a different question: to what extent is it possible to achieve near-optimal performance with inaccurate AIPs? As a matter of fact, it is very common in real-life engineering to model individual components in isolation without considering whether they belong to a larger system. Here we will fall short of giving a complete answer but we will make some observations to suggest that agents might not always require the most accurate AIPs. A view that is also supported by our experiments.

First, if the IALS can produce at least a few observation sequences that are similar to the ones generated by the true environment, an agent with sufficient capacity will be able to learn from those, and perform well in the real world. Given that one of the premises of our method is the need for accurate LS, even if the influence predictor was entirely random, the chances of getting just a small percentage of useful experiences may be in fact quite high. Yet, as discussed in Section 4, different frequencies in some of the transitions might change the value of certain ALSHs and in turn affect the agent's policy. This will occur if some of the unrealistic trajectories generated by an inaccurate AIP overlap partly with the realistic ones. Our second argument, is that different influence distributions may still lead to the same, or very similar, optimal policy (Becker et al., 2003). That is, even if the agent is trained on an inaccurate IALS, some of the strategies that the agent will develop might be transferable and could also be useful when followed under real trajectories (Lazaric, 2012).

5. Experiments

The performance of the method is empirically evaluated on two benchmark domains: traffic control and warehouse commissioning. The goal of the experiments is to: (1) Study whether we can reduce training times by replacing the GS with the IALS. (2) Measure the impact of the AIP accuracy on the learning process and the agent's final performance. (3) Investigate the memory needs of the AIP when the agent's memory is finite.

5.1. Experimental setup

Agents are trained separately with PPO (Schulman et al., 2017) on (1) the global simulator (GS), (2) the influence-augmented local simulator (IALS) with an AIP trained offline on a dataset collected from the GS, (3) an IALS that uses an untrained AIP to make predictions (untrained-IALS). To measure the agent's performance, training is interleaved with periodic evaluations on the GS. The results are averaged over 5 runs with different random seeds.

5.2. Traffic control

Figure 2 shows a grid-like traffic network composed of 25 intersections. The agent controls the traffic lights at one of the intersections only. The rest of the traffic lights are controlled by fixed actuators that use sensors to adapt to the traffic. The policies used in this experiment for the actuated traffic light controllers were extensively optimized by Wu et al. (2017). We train agents separately on the two intersections highlighted in Figure 2. The goal is to

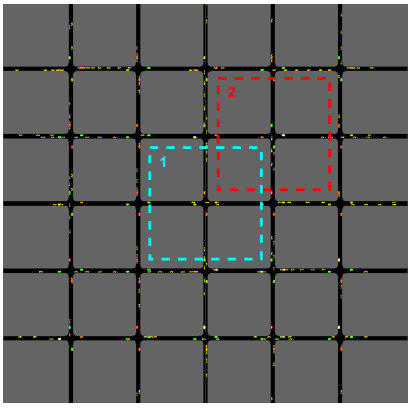

Figure 2. A screenshot of the traffic environment. We train agents separately on the two intersections highlighted by the blue and the red dashed boxes. The rest of the intersections are controlled by fixed actuators that use sensors to adapt to the traffic. The goal is to maximize the average speed of cars within the intersection. The agent can only observe cars inside the dashed boxes.

maximize the average speed of cars within the intersection. The agent can only observe cars inside the dashed boxes.

5.2.1. GS, LS, AIP AND D-SET

The GS and LS are built using Flow (Wu et al., 2017) and SUMO (Lopez et al., 2018). The GS simulates the entire traffic grid (Figure 2) while the LS only models the local neighborhood of the intersection being trained (Figure 9 in Appendix D). The influence sources u_t are binary variables indicating whether or not a car will be entering the simulation from each of the four incoming lanes at the current timestep. The AIP $\hat{I}_\theta$ is a feedforward neural network that we train offline on a dataset of (d_t, u_t) pairs collected from the GS. The d-set d_t is a length 37 binary vector encoding the location of cars along the four incoming lanes. Traffic light information is not included in d_t to prevent confounding (Section 4.2).[6] Since the two intersections highlighted in Figure 2 are influenced differently by the rest of the traffic network we train separate AIPs for each of them.

5.2.2. RESULTS

The plot at the top of Figure 3 are the learning curves of agents trained with the GS, the IALS, and the untrained-IALS to control the traffic lights at intersection 1 (Figure 2).[7] The plot shows the mean episodic reward as a function of real wall-clock time, which for IALS includes the time for data collection and the AIP training time. Agents are trained for 2M timesteps on all three simulators. The dotted horizontal lines at the end of the red and blue curves show the agent's final performance. The black horizontal line indicates the performance of the actuated traffic light controller. The two bar charts at the bottom show the total training time when using each of the three simulators, and the AIP's accuracy with and without training. The results suggest that policies trained on the IALS (red) can match the performance of those trained on the GS (orange) in about $1/3$ of the total training time, despite the IALS is not as accurate as the GS. This is in line with our hypothesis in Section 4.3. Similar influence distributions, $I(u_t|l_t) \approx \hat{I}_\theta(u_t|l_t)$, may lead to the same, or very similar, optimal policy. More experiments exploring this phenomenon are provided in Appendix E.3. In contrast, since the distribution $P^\pi(l_t, u_t)$ induced by the untrained AIP is very different from the true distribution, as evidenced by the high cross entropy loss (blue bar bottom chart), agents trained on the untrained-IALS (blue) perform much worse. A table with a breakdown of the runtimes is included in Appendix C. A comparison of the mean episodic reward as a function of the number of timesteps is provided in Appendix E.2.

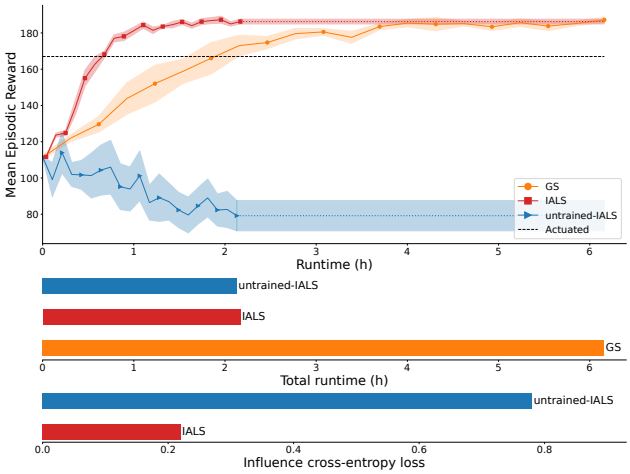

Figure 3. **Top:** Learning curves of agents trained with the GS, the IALS and the untrained-IALS on intersection 1 (Figure 2) as a function of wall-clock time. The dotted horizontal lines show the final performance of the agents after 2M timesteps of training. **Middle:** Total runtime of training for 2M training steps on the three simulators. **Bottom:** Cross entropy loss for the trained and untrained AIPs.

5.3. Warehouse commissioning

A team of 36 robots (blue and purple) need to fetch the items (yellow) that appear with probability 0.02 on the shelves (black dashed lines) of the warehouse in Figure 4. Each robot has been designated a 5×5 square region and can only collect the items that appear on the shelves at the edges. The regions overlap so that each of the 4 item shelves in a robot's region is shared with one of its 4 neighbors. The blue robots have been programmed to go for the oldest item in their region. The purple robot that is inside the

[6] A visual example of a spurious correlation appearing in the traffic problem is given in Appendix B.

[7] Results for intersection 2 are provided in Appendix E.1.

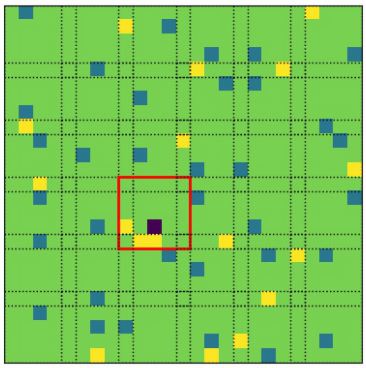

Figure 4. A screenshot of the warehouse environment. The robots (blue and purple) need to fetch the items (yellow) that appear on the shelves (black dashed lines). Each robot can only collect the items that appear on their designated region. We train the purple robot inside the red box. The blue robots have been programmed to go for the oldest item in their region. The purple robot only receives information about its own location and what items need to be collected. However, cannot see the location of the other robots.

region highlighted by the red box, still needs to be trained. This robot receives as observations a bitmap encoding its own location and a set of 12 binary variables that indicate whether or not each of the 12 items within its region needs to be collected. The purple robot, however, cannot see the location of the other robots even though all of them are directly or indirectly influencing it through their actions.

5.3.1. GS, LS, AIP AND D-SET

The GS simulates the entire warehouse (Figure 4), while the LS models only the 5×5 square region delimited by the red box (Figure 9 in Appendix D). The influence sources u_t encode the location of the four neighbor robots. The AIP is a GRU (Cho et al., 2014) that we train offline on a dataset of (d_t, u_t) pairs collected from the GS. If the AIP predicts that any of the neighbor robots is at one of the 12 cells within the red box and there is an active item on that cell, that item is removed and the purple robot can no longer collect it. The d-set d_t includes the history of the 12 item variables and 12 additional binary variables encoding whether or not the controlled robot was (is) at one of the item locations. The latter variables are meant to differentiate between an item that is gone because the controlled robot collected it from an item that was picked up by the neighbor robots. The rest of variables in l_t (i.e. the robot's history of locations) are unnecessary for predicting u_t, and thus susceptible of becoming confounders (Section 4.2).

5.3.2. RESULTS

The plot at the top of Figure 5 shows the learning curves of the warehouse robot as a function of real wall-clock time, which for IALS includes the time for data collection and the

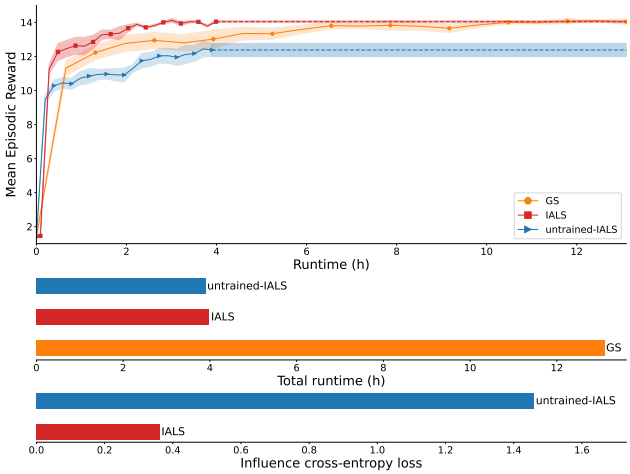

Figure 5. **Top:** Learning curves of agents trained with the GS, the IALS and the untrained-IALS on the the warehouse environment as a function of wall-clock time. The dotted horizontal lines at the end of the red and blue curves show the final performance of the agents after 2M timesteps of training. **Middle:** Total runtime of training for 2M steps on the three simulators. **Bottom:** Cross entropy loss for the trained and untrained AIPs.

AIP training time. Agents are trained for 2M timesteps on all three simulators. The dotted horizontal lines at the end of the red and blue curves show the agent's final performance. The short horizontal line at the beginning of the red curve represents the time for data collection and the AIP's training time. The two bar charts at the bottom show the total training time when using each of the three simulators, and the AIP's accuracy with and without training. Again, we see that robots trained on the IALS (red) are able to reach the same performance as those trained on the GS (orange) in about $1/3$ of the total training time despite the IALS is only approximate. Moreover, robots trained on the untrained-IALS (blue) perform reasonable well on the GS. Although the frequency at which items disappear with the untrained-IALS differs very much from that of the true environment, the basic strategy on how to collect items can still be learned. These results further confirm our hypothesis that inaccurate simulators may, in some cases, render effective policies (Section 4.3). More experiments experiments exploring this phenomenon are provided in Appendix E.3. A table with a breakdown of the runtimes is included in Appendix C. The learning curves as a function of the number of timesteps are provided in Appendix E.2.

5.4. Finite memory agents and AIP history dependence

Here we investigate whether our theoretical result from Theorem 1 also holds in practice. We want to show that when the agent's memory is finite, meaning it can only access observations from k timesteps in the past, an influence predictor which conditions on the same history length is

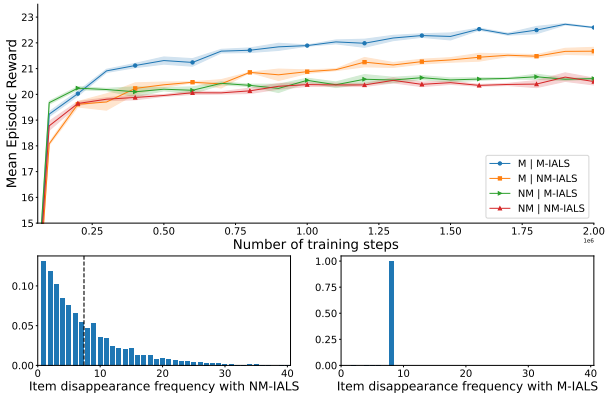

Figure 6. **Top:** Learning curves of agents with (M) and without memory (NM) trained on M-IALS and NM-IALS. **Bottom:** item disappearance frequencies with NM-IALS and with M-IALS.

sufficient. We test this on the warehouse domain. To make the need for memory more evident, we modify the environment so that items always disappear from the robot's region after exactly 8 timesteps. We first train, AIPs with and without memory. We call the resulting simulators M-IALS and NM-IALS respectively. A histogram showing for how long items are active before disappearing under each of the simulators is shown in Figure 6. As expected, while the former can reach an accuracy of 100% (items always remain for 8 timesteps with the M-IALS), the spectrum is much wider for the NM-IALS. This is because the latter can only estimate the marginal distribution $P^{\pi}(u_t|o_t)$. Then, we train agents with (M) and without memory (NM) on the M-IALS and the NM-IALS. The results for all four combinations are shown in Figure 6. As indicated by the theory, the plot shows that when agents have no memory AIPs may condition only on the current observation (red). In such cases, the extra level of detail that a more accurate history-dependent AIP can provide is wasted (green). In contrast, when agents can distinguish observations from one another based on their memories from the past, AIPs that make predictions by looking at the history are fundamental. This is evidenced by the gap between the blue and orange curves. It is worth to point out that, even though the memory agent performs worse when trained on the NM-IALS (orange) than when trained on the M-IALS (blue), it can still outperform the agents with no memory (red and green). We posit that this is because, with the NM-IALS items still disappear after 8 timesteps *on average* (see dashed vertical line in the bottom left histogram in Figure 6), which allows the memory agent to learn to not go for an item if this has been active for a long time. This once again suggests that, in some cases, inaccurate influence predictors might still provide good enough experiences to learn from.

6. Conclusion

This paper has offered a practical solution to allow the application of Deep RL methods to large systems, where performing exhaustive simulations can not be afforded. We focused on domains where, although the agent only interacts with a local portion of the environment, it is influenced by the global dynamics. The main idea was to replace the computationally inefficient global simulator by a lightweight version that only models the agent's local problem. However, as we showed in our experiments, directly doing this sometimes translates in a distribution shift on the agent's experience that yields poor performing policies. A good simulator needs to account for the interactions between the local region and the global dynamics. The results of our experiments suggested that by combining a pretrained influence predictor with the local simulator, we could speed up the learning process considerably while matching the performance of agents trained on the global simulator. Moreover, we analyzed the consequences of training influence predictors on data distributions that are different from those the predictor sees when deployed and resolved that, when possible, the human designer should remove from the input those variables that are irrelevant for predicting the influence sources. Finally, in line with the results in Section 4.1, the last experiment revealed that the agent's memory capacity limits the memory needs of the influence predictor.

Future work may explore how to train multiple agents using independent IALS. This could lead to even further speedups since agents could train on separate simulators running in parallel. However, having agents learning simultaneously would imply that the influence distributions would no longer be stationary since a change in any of the agents' policies could alter the other agents' local dynamics. Hence the method would need to be reworked such that the AIPs can handle moving targets. Another direction for future research is to study how to deal with environments where coverage of untrained and trained policies is different. That is, environments where random policies are unlikely to reach certain ALSHs. If the AIPs cannot generalize to the new data points, a solution would be to retrain the AIPs when the agent starts visiting new ALSHs. Nonetheless, paying the computational cost associated to this solution might not be necessary in many cases since, as we discuss in section 4.3, slightly inaccurate simulators might be sufficient to train good performing policies.

Acknowledgements

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 —INFLUENCE).

References

Acid, S. and De Campos, L. M. An algorithm for finding minimum d-separating sets in belief networks. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, 1996.

Arjovsky, M. Out of distribution generalization in machine learning. *arXiv preprint arXiv:2103.02667*, 2021.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Becker, R., Zilberstein, S., Lesser, V., and Goldman, C. V. Transition-independent decentralized Markov decision processes. In *Proceedings of the Second Joint International Conference on Autonomous Agents and Multiagent Systems*, pp. 41–48, 2003.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., and Hassabis, D. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 115–123, 1996.

Boutilier, C., Dean, T., and Hanks, S. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11: 1–94, 1999.

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

Chitnis, R. and Lozano-Pérez, T. Learning compact models for planning with exogenous processes. In *Proceedings of the Conference on Robot Learning*, 2020.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

Claes, D., Oliehoek, F., Baier, H., Tuyls, K., et al. Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the 16th international conference on autonomous agents and multiagent systems*, pp. 492–500, 2017.

Congeduti, E., Mey, A., and Oliehoek, F. A. Loss bounds for approximate influence-based abstraction. In *Proceedings of the Twentieth International Conference on Autonomous Agents and MultiAgent Systems*, 2021.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Ganesh, S., Vadori, N., Xu, M., Zheng, H., Reddy, P., and Veloso, M. M. Reinforcement learning for market making in a multi-agent dealer market. *ArXiv*, abs/1911.05892, 2019.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Gupta, A., Badr, Y., Negahban, A., and Qiu, R. G. Energy-efficient heating control for smart buildings with deep reinforcement learning. *Journal of Building Engineering*, 34:101739, 2021.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2450–2462, 2018.

Hausknecht, M. and Stone, P. Deep recurrent Q-learning for partially observable MDPs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

He, J., Suau, M., and Oliehoek, F. Influence-augmented online planning for complex environments. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4392–4402, 2020.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kaelbling, L. P., Littman, M., and Moore, A. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–285, 1996.

Kakade, S. M. *On the sample complexity of reinforcement learning*. PhD thesis, University College London, 2003.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Priol, R. L., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Kumar, A., Zilberstein, S., and Toussaint, M. Scalable multiagent planning using probabilistic inference. In *Proc. of the International Joint Conference on Artificial Intelligence*, pp. 2140–2146, 2011.

Lazaric, A. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pp. 143–173. Springer, 2012.

Littman, M. L. Memoryless policies: Theoretical limitations and practical results. In *Proc. of the Third International Conference on Simulation of Adaptive Behavior : From Animals to Animats 3*, pp. 238–245, 1994.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. Microscopic traffic simulation using SUMO. In *The 21st International Conference on Intelligent Transportation Systems*, 2018.

McCallum, A. K. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1995.

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.

Moerland, T. M., Broekens, J., and Jonker, C. M. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

Nair, R., Varakantham, P., Tambe, M., and Yokoo, M. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proc. of the National Conference on Artificial Intelligence*, pp. 133–139, 2005.

van der Pol, E. and Oliehoek, F. A. Coordinated Deep Reinforcement Learners for traffic light control. NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems, 2016.

Oh, J., Chockalingam, V., Singh, S., and Lee, H. Control of memory, active perception, and action in minecraft. In *Proc. of the 33rd International Conference on Machine learning*, pp. 2790–2799, 2016.

Oliehoek, F., Witwicki, S., and Kaelbling, L. A sufficient statistic for influence in structured multiagent environments. *Journal of Artificial Intelligence Research*, 70: 789–870, 2021.

Pearl, J. *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Pearl, J. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

Peters, J., Bühlmann, P., and Meinshausen, N. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pp. 947–1012, 2016.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset shift in machine learning*. The MIT Press, 2009.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *4th International Conference on Learning Representations*, 2016.

Schmidhuber, J. Reinforcement learning in Markovian and non-Markovian environments. In *Advances in Neural Information Processing Systems*, pp. 500–506, 1991.

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

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

Singh, S. P., Jaakkola, T., and Jordan, M. I. Learning without state-estimation in partially observable Markovian decision processes. In *Proc. of the International Conference on Machine Learning*, pp. 284–292, 1994.

Suau, M., Agapitos, A., Lynch, D., Farrell, D., Zhou, M., and Milenovic, A. Offline contextual bandits for wireless network optimization. *arXiv preprint arXiv:2111.08587*, 2021.

Suau, M., He, J., Congeduti, E., Starre, R. A., Czechowski, A., and Oliehoek, F. A. Influence-aware memory architectures for deep reinforcement learning in pomdps. *Neural Computing and Applications. Special Issue on Adaptive and Learning Agents*, 2022.

Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224, 1990.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Tian, J., Paz, A., and Pearl, J. *Finding minimal d-separators*. 1998.

van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Varakantham, P., Marecki, J., Yabu, Y., Tambe, M., and Yokoo, M. Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, 2007.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wang, J., Xu, W., Gu, Y., Song, W., and Green, T. Multi-agent reinforcement learning for active voltage control on power distribution networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Williams, R. J. and Peng, J. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

Witwicki, S. J. and Durfee, E. H. Towards a unifying characterization for quantifying weak coupling in Dec-POMDPs. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems*, pp. 29–36, 2011.

Wu, C., Kreidieh, A., Parvate, K., Vinitsky, E., and Bayen, A. M. Flow: A modular learning framework for autonomy in traffic. *arXiv preprint arXiv:1710.05465*, 2017.

A. Proofs

Theorem 1. *Let $l_{t-k:t}$ be a truncated version of l_t containing only the last k action-local-state pairs and let $a_{0:t-k}$ be the sequence of past actions from time 0 up to k. Let the agent's policy $\bar{\pi}$ be a function of $h_{t-k:t}$ $\bar{\pi}(a_t|h_{t-k:t})$, where $h_{t-k:t}$ is also a truncated version of h_t. If for all u_t we have $P(u_t|l_{t-k:t}, a_{0:t-k}) = P(u_t|l_{t-k:t})$, then an influence predictor that conditions only on $l_{t-k:t}$, $\bar{I}(u_t|l_{t-k:t})$, is sufficient to guarantee no loss in value.*[8]

Proof. We will prove that, when conditioning on the last k elements of the AOH, $h_{t-k:t}$, the action-value function $\bar{Q}^{\bar{\pi}}(h_{t-k:t}, a_t)$, can be expressed in terms of an influence predictor that conditions on $l_{t-k:t}$, $\bar{I}(u_t|l_{t-k:t})$. If this is true, then Q estimates can be obtained by sampling from the finite memory IALS (with influence predictor $\bar{I}(u_t|l_{t-k:t})$) and the agent's optimal policy (in the class of finite memory policies of the form $\bar{\pi}(a_t|h_{t-k:t})$) can be found via policy iteration (Sutton & Barto, 1998). Note that, finite memory policies have been shown to perform arbitrarily bad compared to their full memory counterparts (Singh et al., 1994). Moreover, empirical results have revealed that standard RL methods for MDPs may fail when state representations are not Markovian (Littman, 1994). Nonetheless, here we are only concerned with whether or not we can find the same finite memory policies when using influence predictors that condition only on $h_{t-k:t}$ as those found with influence predictors that condition on the full AOH. That is, independently of the effectiveness of the RL method we use or the absolute performance of the policies we find.

Given the full AOH, the action-value function $Q^{\pi}(h_t, a_t)$ of a local-FPOMDP satisfies the Bellman equation and can be expressed as

$$Q^{\pi}(h_t, a_t) = \sum_{x_t} P(x_t|h_t)\dot{R}(x_t, a_t) + \sum_{l_t} P(l_t|h_t) \sum_{x_{t+1}} P(o_{t+1}|l_t, a_t) \sum_{a_{t+1}} \pi(a_{t+1}|h_{t+1}) Q^{\pi}(\langle h_t, a_t, o_{t+1}\rangle, a_{t+1}), \quad (7)$$

where and $P(x_t|h_t)$ is the probability of the local states x_t given the previous AOH, and from (2)

$$P(o_{t+1}|l_t, a_t) = \sum_{x_{t+1}} \dot{O}(o_{t+1}|x_{t+1}) \sum_{u_t} \dot{T}(x_{t+1}|x_t, a_t, u_t) I(u_t|l_t) \quad (8)$$

If, instead, the agent's policy $\bar{\pi}$, and in turn the $\bar{Q}^{\bar{\pi}}$ function, can condition only on the last k elements in the AOH $h_{t-k:t}$, we have

$$\bar{Q}^{\bar{\pi}}(h_{t-k:t}, a_t) = \sum_{h_{0:t-k}} P^{\bar{\pi}}(h_{0:t-k}|h_{t-k:t}) Q^{\bar{\pi}}(h_t = \langle h_{0:t-k}, h_{t-k:t}\rangle, a_t),$$

where $h_{0:t-k}$ is the AOH from 0 to k and the above expression is just the expected Q value given $h_{t-k:t}$. Then, using (7)

$$\begin{aligned}\bar{Q}^{\bar{\pi}}(h_{t-k:t}, a_t) = &\sum_{h_{0:t-k}} P^{\bar{\pi}}(h_{0:t-k}|h_{t-k:t}) \sum_{x_t} P(x_t|h_{0:t-k}, h_{t-k:t}) \dot{R}(x_t, a_t) \\ &+ \sum_{h_{0:t-k}} P^{\bar{\pi}}(h_{0:t-k}|h_{t-k:t}) \sum_{l_t} P(l_t|h_{0:t-k}, h_{t-k:t}) \sum_{o_{t+1}} P(o_{t+1}|l_t, a_t) \sum_{a_{t+1}} \bar{\pi}(a_{t+1}|h_{-k+1:t+1}) \bar{Q}^{\bar{\pi}}(h_{-k+1:t+1}, a_{t+1})\end{aligned}$$

and from (8) we have

$$\begin{aligned}&\sum_{h_{0:t-k}} P^{\bar{\pi}}(h_{0:t-k}|h_{t-k:t}) \sum_{l_t} P(l_t|h_{0:t-k}, h_{t-k:t}) \sum_{o_{t+1}} P(o_{t+1}|l_t, a_t) \\ &\qquad = \sum_{o_{t+1}} \sum_{x_{t+1}} \dot{O}(o_{t+1}|x_{t+1}) \sum_{u_t} \sum_{l_t} \dot{T}(x_{t+1}|x_t, a_t, u_t) I(u_t|l_t) \sum_{h_{0:t-k}} P^{\bar{\pi}}(h_{0:t-k}|h_{t-k:t}) P(l_t|h_{0:t-k}, h_{t-k:t}).\end{aligned}$$

Then, using the rule of conditional probability, the last summation can be simplified as

$$\sum_{h_{0:t-k}} P^{\bar{\pi}}(h_{0:t-k}|h_{t-k:t}) P(l_t|h_{0:t-k}, h_{t-k:t}) = \sum_{h_{0:t-k}} P^{\bar{\pi}}(l_t, h_{0:t-k}|h_{t-k:t}) = P^{\bar{\pi}}(l_t|h_{t-k:t})$$

[8] Note that this refers only to the value of polices of the form $\bar{\pi}(a_t|h_{t-k:t})$, which might perform arbitrarily worse than policies that condition on the full AOH, $\pi(a_t|h_t)$ (Singh et al., 1994).

and thus

$$\sum_{l_t} \dot{T}(x_{t+1}|x_t,a_t,u_t)I(u_t|l_t)P^{\bar{\pi}}(l_t|h_{t-k:t}) = \sum_{l_{0:t-k},l_{t-k:t}} \dot{T}(x_{t+1}|x_t,a_t,u_t)I(u_t|l_{0:t-k},l_{t-k:t})P^{\bar{\pi}}(l_{0:t-k},l_{t-k:t}|h_{t-k:t})$$

$$= \sum_{l_{0:t-k},l_{t-k:t}} \dot{T}(x_{t+1}|x_t,a_t,u_t)I(u_t|l_{0:t-k},l_{t-k:t})P^{\bar{\pi}}(l_{0:t-k}|l_{t-k:t})P^{\bar{\pi}}(l_{t-k:t}|h_{t-k:t})$$

since $P^{\bar{\pi}}(l_{0:t-k}|l_{t-k:t},h_{t-k:t}) = P^{\bar{\pi}}(l_{0:t-k}|l_{t-k:t})$ because $l_{t-k:t}$ is the only parent of $h_{t-k:t}$

$$= \sum_{l_{t-k:t}} \dot{T}(x_{t+1}|x_t,a_t,u_t) \sum_{l_{0:t-k}} P^{\bar{\pi}}(u_t,l_{0:t-k}|l_{t-k:t})P^{\bar{\pi}}(l_{t-k:t}|h_{t-k:t})$$

$$= \sum_{l_{t-k:t}} \dot{T}(x_{t+1}|x_t,a_t,u_t)\bar{I}^{\bar{\pi}}(u_t|l_{t-k:t})P^{\bar{\pi}}(l_{t-k:t}|h_{t-k:t})$$

Hence, putting all together,

$$\bar{Q}^{\bar{\pi}}(h_{t-k:t},a_t) = \sum_{h_{0:t-k}} P^{\bar{\pi}}(h_{0:t-k}|h_{t-k:t}) \sum_{x_t} P(x_t|h_{0:t-k},h_{t-k:t})\dot{R}(x_t,a_t)$$

$$+ \sum_{o_{t+1}} \sum_{x_{t+1}} \dot{O}(o_{t+1}|x_{t+1}) \sum_{u_t} \sum_{l_{t-k:t}} \dot{T}(x_{t+1}|x_t,a_t,u_t)\bar{I}^{\bar{\pi}}(u_t|l_{t-k:t})P^{\bar{\pi}}(l_{t-k:t}|h_{t-k:t}) \sum_{a_{t+1}} \bar{\pi}(a_{t+1}|h_{-k+1:t+1})\bar{Q}^{\bar{\pi}}(h_{-k+1:t+1},a_{t+1})$$

Intuitively, we see that, because the action-value function $\bar{Q}^{\bar{\pi}}(a_t,h_{t-k:t})$ is just an expectation over all possible h_t given $h_{t-k:t}$, the distribution of u_t given the full ALSH l_t, $I(u_t|l_t)$, is effectively washed away by the same expectation. Hence, we may as well condition the influence predictor directly on $l_{t-k:t}$, $\bar{I}^{\bar{\pi}}(u_t|l_{t-k:t})$.

Finally, using the assumption $P(u_t|l_{t-k:t},a_{0:t-k}) = P(u_t|l_{t-k:t})$ we can drop the superscript $\bar{\pi}$ from $\bar{I}^{\bar{\pi}}(u_t|l_{t-k:t})$. This assumption is important because the distribution $\bar{I}(u_t|l_{t-k:t})$ is generally not well-defined. Hence, if we estimate $\bar{I}^{\bar{\pi}_0}(u_t|l_{t-k:t})$ from samples collected while following $\bar{\pi}_0$ but $P(u_t|l_{t-k:t},a_{0:t-k}) \neq P(u_t|l_{t-k:t})$, then, when we update the policy, the marginal distribution $P^{\bar{\pi}}(a_{0:t-k})$ will change and the old estimates will be biased.

□

Theorem 2. *A subset of variables d_t from l_t is guaranteed to elicit an invariant predictor of u_t, across all $\pi \in \Pi$, if (i) d_t constitutes a d-separating set and (ii) all policies are functions of the local AOH, $\pi(h_t)$.*

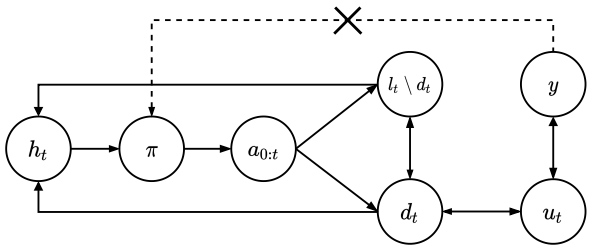

Figure 7. Graphical causal model of a local-FPOMDP. The bidirectional arrows between two sets of variables (e.g. between d_t and u_t) indicate that there can be variables in one set that are affected by other variables in the other set and vice-versa. As depicted by the two diagonal arrows in the middle, actions $a_{0:t}$ may or may not be included in d_t. The top dashed arrow connecting y and π indicates that π cannot be a function of the non-local variables.

Proof. In this proof, we will make use of the graphical causal model in Figure 7. We know that the agent's policy π can only influence the environment through its actions $a_{0:t}$. Moreover, from the definition of d-set (Definition 5), we know that the *direct path* (Pearl, 2000) that connects the past sequence of actions $a_{0:t}$ with u_t must go through d_t. Otherwise, $(u_t \not\perp\!\!\!\perp d_t \setminus l_t | d_t)$ and d_t would not constitute a d-set. Note that, as shown by the diagonal arrows in the middle, actions in

$a_{0:t}$ may or may not be included in d_t. Finally, the second condition in Theorem 2 implies that there cannot be *backdoor paths* (Pearl, 2000) between π and u_t. Hence, if we control for d_t, π and u_t become conditionally independent,

$$P(u_t|d_t) = P^{\pi}(u_t|d_t) \quad \text{for all} \quad \pi \in \Pi.$$

Moreover, for the above to hold, the d-set does not necessarily need to be minimal, $d_t \subseteq d_t^*$. That is, as long as the path between $l_t \setminus d_t$ and u_t remains closed, d_t may include as many additional (irrelevant) variables as there are in l_t. Note that this path is only open when d_t is not a d-set. □

Proposition 1. *Let π_0 be the exploratory policy used to collect the dataset. Then, for all $\pi \in \Pi$*

$$KL(P^{\pi_0}(d_t, u_t)||P^{\pi}(d_t, u_t)) \leq KL(P^{\pi_0}(l_t, u_t)||P^{\pi}(l_t, u_t))$$

where KL is the Kullback–Leibler divergence.

Proof.

$$\begin{aligned}
KL(P^{\pi_0}(d_t, u_t)||P^{\pi}(d_t, u_t)) &= \sum_{d_t, u_t} P^{\pi_0}(d_t, u_t) \log\left(\frac{P^{\pi_0}(d_t, u_t)}{P^{\pi}(d_t, u_t)}\right) \\
&= \sum_{d_t, u_t} P^{\pi_0}(u_t, d_t) \log\left(\frac{P^{\pi_0}(u_t|d_t)P^{\pi_0}(d_t)}{P^{\pi}(u_t|d_t)P^{\pi}(d_t)}\right) \\
&= \sum_{d_t} \log\left(\frac{P^{\pi_0}(d_t)}{P^{\pi}(d_t)}\right) \sum_{u_t} P^{\pi_0}(u_t, d_t) \\
&\quad \text{since, from Theorem 2, we know that } P(u_t|d_t) \text{ is invariant across all } \pi \in \Pi \\
&= KL(P^{\pi_0}(d_t)||P^{\pi}(d_t))
\end{aligned}$$

Similarly, because $P(u_t|l_t)$ is also invariant across all $\pi \in \Pi$, $P^{\pi_0}(u_t|l_t) = P^{\pi}(u_t|l_t)$,

$$KL(P^{\pi_0}(l_t, u_t)||P^{\pi}(l_t, u_t)) = KL(P^{\pi_0}(l_t)||P^{\pi}(l_t)).$$

Then, since $d_t \subseteq l_t$, we can write

$$P^{\pi}(l_t) = P^{\pi}(d_t, l_t \setminus d_t)$$

and, using the chain rule

$$\begin{aligned}
KL(P^{\pi_0}(l_t)||P^{\pi}(l_t)) &= KL(P^{\pi_0}(l_t \setminus d_t|d_t)||P^{\pi}(l_t \setminus d_t|d_t)) + KL(P^{\pi_0}(d_t)||P^{\pi}(d_t)) \\
&\geq KL(P^{\pi_0}(d_t)||P^{\pi}(d_t))
\end{aligned}$$

where the last inequality holds because the KL divergence is always non-negative. □

Corollary 1. *Let $a_{0:t}$ be the past sequence of actions from 0 to t. If $P(d_t|do(a_{o:t})) = P(d_t)$ for all d_t and $a_{0:t}$. Then, for any $\pi_1 \neq \pi_2 : \pi_1, \pi_2 \in \Pi$ we have $P^{\pi_1}(d_t, u_t) = P^{\pi_2}(d_t, u_t)$ even though $P^{\pi_1}(l_t, u_t) \neq P^{\pi_2}(l_t, u_t)$.*

Proof. Follows from Proposition 1. We know that π can only influence the environment through the actions $a_{0:t}$. Hence,

$$\forall \pi_1 \neq \pi_2, d_t, a_{0:t} \quad P(d_t|do(a_{0:t})) = P(d_t) \Rightarrow P^{\pi_1}(d_t) = P^{\pi_2}(d_t) \iff KL(P^{\pi_1}(d_t)||P^{\pi_2}(d_t)) = 0$$

and from the proof of Proposition 1

$$KL(P^{\pi_1}(d_t, u_t)||P^{\pi_2}(d_t, u_t)) = KL(P^{\pi_1}(d_t)||P^{\pi_2}(d_t))$$

which means

$$\forall \pi_1 \neq \pi_2, d_t, a_{0:t}, u_t \quad P(d_t|do(a_{0:t})) = P(d_t) \Rightarrow KL(P^{\pi_1}(d_t, u_t)||P^{\pi_2}(d_t, u_t)) = 0 \iff P^{\pi_1}(d_t, u_t) = P^{\pi_2}(d_t, u_t)$$

and, because $l_t \subseteq a_{0:t}$ then

$$\forall \pi_1 \neq \pi_2, l_t, a_{0:t}, u_t \quad KL(P^{\pi_1}(l_t)||P^{\pi_2}(l_t)) > 0 \iff KL(P^{\pi_1}(l_t, u_t)||P^{\pi_2}(l_t, u_t)) > 0 \iff P^{\pi_1}(l_t, u_t) \neq P^{\pi_2}(l_t, u_t)$$

□

B. Example: Spurious Correlations in the Traffic Domain

Figure 8 shows four screenshots taken from the SUMO simulator (Lopez et al., 2018). These capture a sequence consecutive states in the traffic domain (see Section 5.2 for more details about this environment). The agent's local region is depicted by the red dashed box. The small cyan box on the right of every screenshot indicates the location of an influence source. As shown in the screenshots, when traffic is dense, lines of cars are formed along the incoming lanes in front of the traffic lights. This situation leads to the appearance of spurious correlations between the traffic lights and the influence sources. In particular, Figure 8 reveals that three timesteps after the traffic lights on the horizontal lanes are switched to green a new car appears at the influence source. Although there is clearly no direct relationship between these two events, if this pattern occurs often enough, as it is the case under a uniform random policy, the AIP might pick it up and use it as a shortcut. That is, the AIP might learn to predict that a car will show at the influence source exactly three timesteps after the lights are switched to green. This would indeed be effective at the beginning of training, since traffic jams are common under poor performing policies, but may result in highly inaccurate predictions when policies are further improved. As explained in Section 4.2 the problem above can be prevented if d-sets, rather than full AOHs, are fed into the AIP.

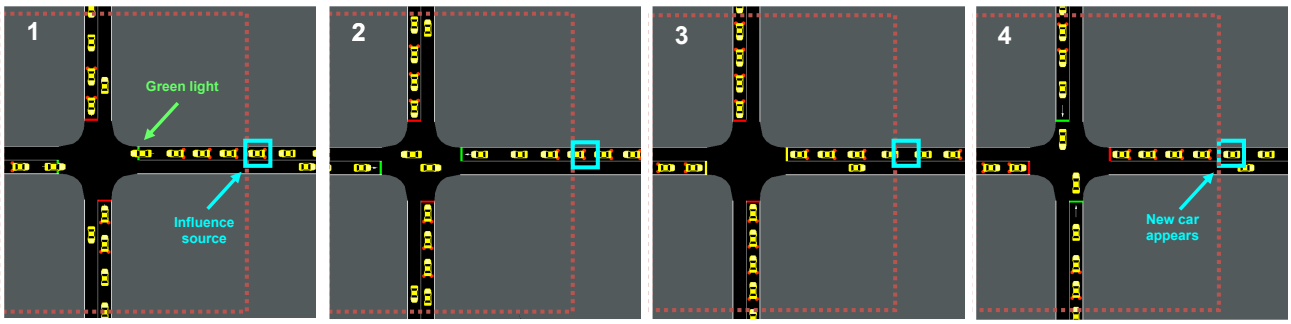

Figure 8. Four screenshots capturing a sequence of states that may occur under a uniform random policy. The agent's local region is depicted by the red dashed box. The small cyan box on the right of every screenshot indicates the location of an influence source. The screenshots reveal that, when the traffic is dense, three timesteps after the traffic lights on the horizontal lanes are switched to green a new car appears at the influence source. This is clearly a spurious correlation as there is not direct relationship between the traffic lights and the influence sources.

C. Runtimes

The table below shows a breakdown of the runtime for the two environments and the three simulators. These were measured on an Intel i7-8650U CPU with 8 cores.

	Simulator	Agents training (h)	Data collection (h)	AIP training (h)	Total (h)
Traffic	GS	6.16	-	-	6.16
	IALS	2.13	0.03	0.01	2.17
	untrained-IALS	2.13	-	-	2.13
Warehouse	GS	13.12	-	-	13.12
	IALS	3.90	0.03	0.06	3.99
	untrained-IALS	3.90	-	-	3.90

D. Local Simulators

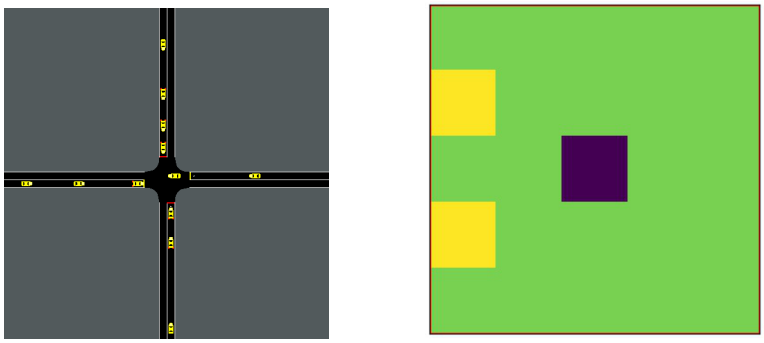

Figure 9. A screenshot of the local simulators for the traffic control (left) and warehouse (right) environments

E. Results

E.1. Traffic control intersection 2

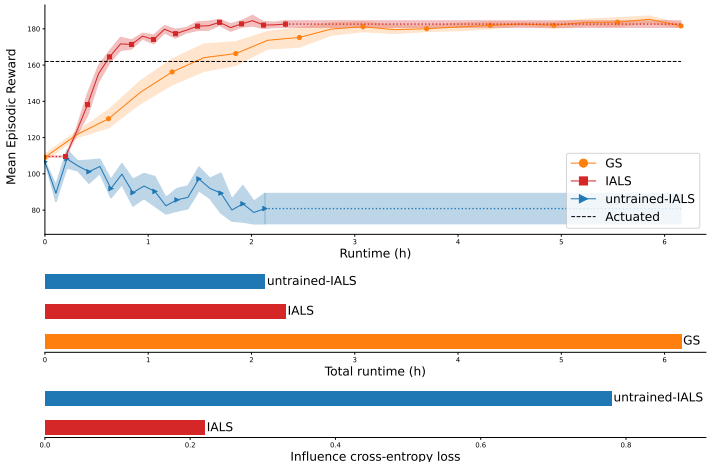

Figure 10. **Top:** Learning curves of agents trained with the GS, the IALS, and the untrained-IALS on intersection 2 (Figure 2) as a function of wall-clock time. The dotted horizontal lines show the final performance of the agents after 2M timesteps of training. The dotted horizontal line at the begining of the red curve corresponds to the AIP training time. **Middle:** Total runtime of training for 2M training steps on the three simulators. **Bottom:** Cross entropy loss for the trained and untrained AIPs.

E.2. Comparison on the number of timesteps

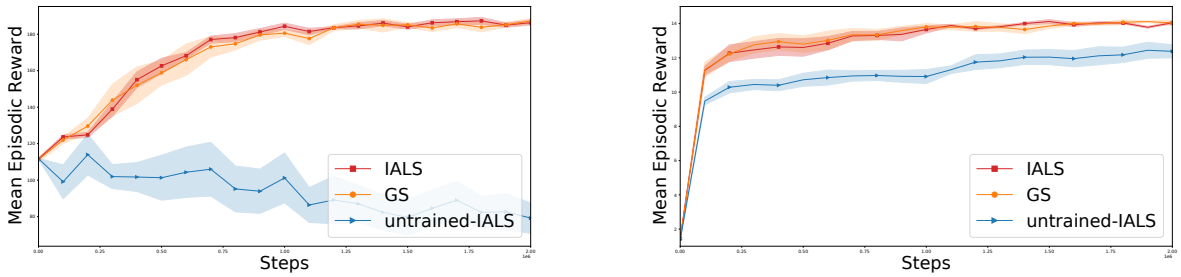

Figure 11. Learning curves as a function of the number of timesteps for the traffic (left) and warehouse (right) environments.

E.3. Sufficiently similar training conditions

Here we further explore our research question in Section 4.3. To what extent can agents trained on inaccurate IALS achieve similar performance to those trained on the GS? To shed some light on this this complicated question, we introduce a new type of AIP which represents the influence sources by a fixed marginal probability $P(u_t)$ independent of the ALSH. We call the resulting simulator F-IALS (F for fixed).

Traffic results: Figure 3 shows the performance of agents trained on the F-IALS with $P(u_t) = 0.1$ (F-IALS 0.1) and $P(u_t) = 0.5$ (F-IALS 0.5). The bar plot at the bottom of Figure 3 shows that the cross-entropy loss is much higher when modeling the influence sources with the marginal distribution $P(u_t)$ (F-IALS 0.1 and F-IALS 0.5) than when modeling them with the learned influence predictor $P(u_t|l_t)$ (IALS). This suggests that there is a non-trivial relationship between ALSH and influence sources, and in principle

$$KL(I(u_t|d_t)||\hat{I}_\theta(u_t|d_t)) < KL(I(u_t|d_t)||P(u_t) = 0.1) < KL(I(u_t|d_t)||P(u_t) = 0.5) \tag{9}$$

Nonetheless, agents trained on the F-IALS 0.1 reach the same (intersection 2) or similar (intersection 1) level of performance as those trained on the IALS. This is in line with our hypothesis in section 4.3. If real trajectories are somewhat likely under a random influence predictor, an agent with sufficient capacity will be able to learn from them and perform well on the true environment. In fact, given that the probability used for the inflow of vehicles entering the GS is also 0.1, the chances of generating traffic patterns with the F-IALS (0.1) similar to those of the GS are very high. In contrast, when setting the probability of cars entering the LS to a less sensible $P(u_t) = 0.5$ agents trained on the F-IALS (0.5) can no longer match the same performance level.

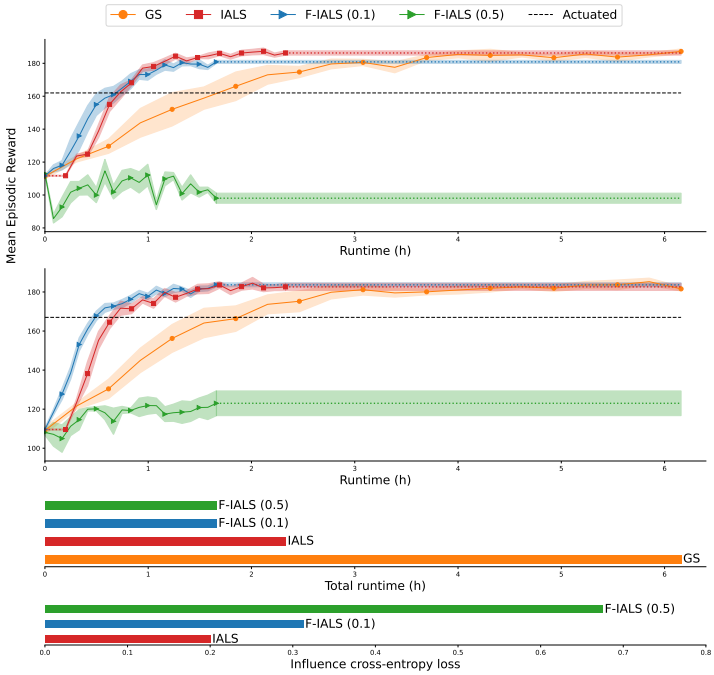

Figure 12. **Top:** Learning curves of agents trained with the GS, the IALS and the F-IALS on the the two intersections highlighted in Figure 2 as a function of wall-clock time. The dotted horizontal lines show the final performance of the agents after 2M timesteps of training. **Second from the bottom:** Total runtime of training for 2M training steps on the three simulators. **Bottom:** Cross entropy loss evaluated at intersection 1 for the three AIPs (values are very similar for intersection 2).

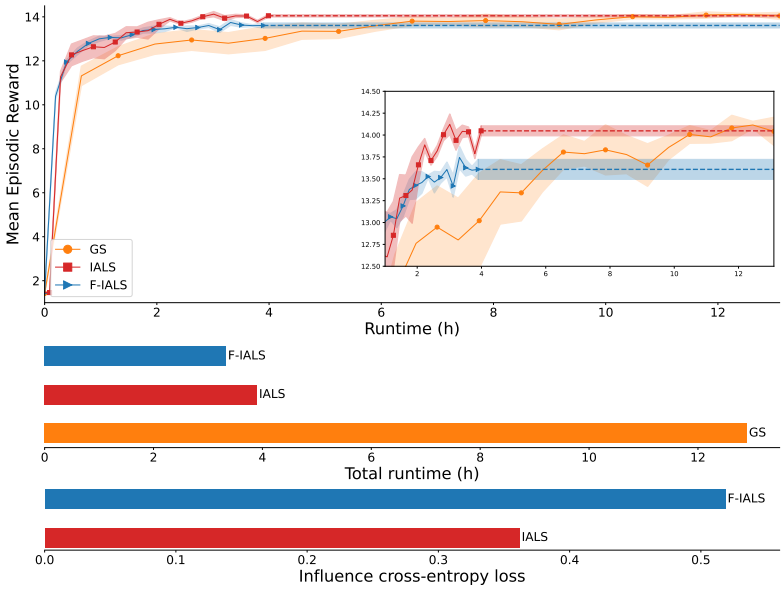

Figure 13. **Top:** Learning curves of agents trained with the GS, the IALS and the F-IALS on the the warehouse commissioning task as a function of wall-clock time. Zoom in version of the same chart showing the performance difference between IALS and F-IALS. The dotted horizontal lines show the final performance of the agents after 2M timesteps of training. The dotted horizontal line at the beginning of the red curve corresponds to the AIP training time. **Second from the bottom:** Total runtime of training for 2M steps on the three simulators. **Bottom:** Cross entropy loss for the two AIPs.

Warehouse results: In this environment the fixed influence source probabilities is set to an estimate $\hat{P}(u_t)$ of true value $P^{\pi_0}(u_t)$, which we approximated empirically from $N = 10K$ samples collected from the GS while following π_0. Even so, the cross-entropy of the F-IALS is again higher than that of the learned influence predictor $\hat{I}_\theta(u_t|l_t)$, indicating again that influence sources u_t and ALSHs are strongly coupled with one another and that there is a non-trivial relationship between them,

$$KL(I(u_t|d_t)||\hat{I}_\theta(u_t|d_t)) < KL(I(u_t|d_t)||\hat{P}(u_t)) \tag{10}$$

Despite being less accurate, agents trained on the F-IALS can also perform well on the true environment. Yet, they do not reach the same level of performance as those obtained with the GS and the IALS (see zoom in version in Figure 5). Even though, the basic strategy on how to collect items can be learned from the F-IALS, the simulator does not provide the extra level of detail needed to learn better policies. That is, a consistent pattern that is present in both the GS and the IALS by which items that have been active the longest are more likely to disappear next. With this, agents can learn to not go for an item when the chances that a neighbor robot will get there first are high.

All in all we see that, in certain situations, inaccurate influence predictors can still provide a fair amount of useful experiences for the agent to perform well on the true environment. However, in domains with complicated dynamics, such as our warehouse environment, the best policies can only be obtained when simulations provide the extra level of detail that only accurate influence predictors are able to deliver.

F. Implementation Details

Approximate influence predictor: Due to the sequential nature of the problem, rather than feeding the full past history every time we make a prediction, we use a recurrent neural network (RNN) (Hochreiter & Schmidhuber, 1997; Cho et al., 2014) and process observations one at a time,

$$P(u_t|l_t) \approx \hat{I}_\theta(u_t|\hat{h}_{t-1}, o_t) = F_{\text{rnn}}(\hat{h}_{t-1}, o_t, u_t), \tag{11}$$

where we use $\hat{h}$ to indicate that the history h is embedded in the RNN's internal memory.

Given that we generally have multiple influence sources $u_t = \langle u_t^1 \dots u_t^M \rangle$, we need to fit M separate models $\hat{I}_{\theta_m}$ to predict each of the M influence sources. In practice, to reduce the computational cost, we can have a single network with a common representation module for all influence sources and output their probability distributions using M separate heads. This representation assumes that the influence sources are independent of one another,

$$I(u_t|l_t) = \prod_{m=0}^{M} P(u_t^m|l_t), \tag{12}$$

which is true for the two domains we study in this paper.

Practical implications of Theorem 1: One important consideration when training RNNs via backpropagation through time (BPTT) is to choose the right length for the input sequences (Williams & Peng, 1990). This determines the number of steps the network is backpropagated, and thus for how long past information can be retained. On the one hand, longer sequences will often provide more information to predict the influence sources, on the other, they will also make it harder to optimize the network. Ideally, we would like to choose just the right sequence length such that the agent cannot perceive any distribution shift in the local transitions. Assuming that in our environment $P(u_t|l_{t-k:t}, a_{0:t-k}) = P(u_t|l_{t-k:t})$, from Theorem 1 we know that the sequence length should be (at least) as long as that of the agent's (if these are also modelled by RNNs or as long as the number of stacked observations fed to feedforward neural networks; FNN). If $P(u_t|l_{t-k:t}, a_{0:t-k}) \neq P(u_t|l_{t-k:t})$ we can still condition the AIP only on $l_{t-k:t}$ but we might need to retrain the AIP every certain number of policy updates to prevent it from becoming stale (see last paragraph in the proof of theorem 1; Appendix A).

Policies: We model policies by FNNs. In the warehouse environment, since the agent needs memory to perform well, policies are fed with a stack of the last 8 observations. This architecture performed better than GRUs. In the traffic control task, on the other hand, policies are fed only with the current observation as this seems to be sufficient to predict the influence sources.

G. Algorithms

Algorithm 1 Collect dataset with GS

Input: T, O, π_0 ▷ Global simulator, global observation function, and exploratory policy
for $n \in \langle 0, ..., N/T \rangle$ **do**
  $s_0 \leftarrow$ reset ▷ Reset initial state
  $x_0 \leftarrow s_0$ ▷ Extract local state from global state
  $l_0 \leftarrow x_0$ ▷ Initialize ALSH with initial local state
  $o_0 \sim O(\cdot \mid s_0)$ ▷ Sample observation from O
  $h_0 \leftarrow o_0$ ▷ Initialize AOH with initial observation
  for $t \in \langle 0, ..., T \rangle$ **do**
    $\langle u_t^0, ..., u_t^k \rangle \leftarrow s_t$ ▷ Extract influence sources from global state
    $D \leftarrow \{l_t, \langle u_t^0, ..., u_t^k \rangle\}$ ▷ Append ALSH-influence-source pair to the dataset
    $a_t \sim \pi(\cdot \mid h_t)$ ▷ Sample action
    $s_{t+1} \sim T(\cdot \mid s_t, a_t)$ ▷ Sample next state from GS
    $x_{t+1} \leftarrow s_{t+1}$ ▷ Extract local state from global state
    $l_{t+1} \leftarrow \langle a_t, x_{t+1} \rangle$ ▷ Append action-local-state pair to ALSH
    $o_{t+1} \sim O(\cdot \mid s_{t+1})$ ▷ Sample observation from O
    $h_{t+1} \leftarrow \langle a_t, o_{t+1} \rangle$ ▷ Append actions-observation pair to the AOH
  end for
end for

Algorithm 2 Simulate trajectory with IALS

Input: $\dot{T}, \dot{R}, \dot{O}, \pi, \hat{I}_\theta$ ▷ local simulator, local reward and observation functions, policy, AIP
$x_0 \leftarrow$ reset ▷ Reset initial state
$o_0 \sim \dot{O}(\cdot|x_0)$ ▷ Sample observation from $\dot{O}$
$h_0 \leftarrow o_0$ ▷ Initialize AOH with initial observation
for $t \in \langle 0, ..., T \rangle$ **do**
    $a_t \sim \pi(\cdot \mid h_t)$ ▷ Sample action
    $\dot{R}(x_t, a_t)$ ▷ Compute reward
    $u_t \sim \hat{I}_\theta(\cdot \mid l_t)$ ▷ Sample influence sources from AIP
    $x_{t+1} \sim \dot{T}(\cdot \mid x_t, a_t, u_t)$ ▷ Sample next local state from LS
    $l_{t+1} \leftarrow \langle a_t, x_{t+1} \rangle$ ▷ Append action-local-state pair to ALSH
    $o_{t+1} \sim \dot{O}(\cdot \mid x_{t+1})$ ▷ Sample observation from O
    $h_{t+1} \leftarrow \langle a_t, o_{t+1} \rangle$ ▷ Append action-observation pair to AOH
end for