Surrogate Bayesian Networks for Approximating Evolutionary Games

Vincent Hsiao
Univ. of Maryland, College Park

Dana Nau
Univ. of Maryland, College Park

Rina Dechter
Univ. of California, Irvine

Abstract

Spatial evolutionary games are used to model large systems of interacting agents. In earlier work, a method was developed using Bayesian Networks to approximate the population dynamics in these games. One advantage of that approach is that one can smoothly adjust the size of the network to get more accurate approximations. However, scaling the method up can be intractable if the number of strategies in the evolutionary game increases. In this paper, we propose a new method for computing more accurate approximations by using surrogate Bayesian Networks. Instead of doing inference on larger networks directly, we do it on a much smaller surrogate network extended with parameters that exploit the symmetry inherent to the domain. We learn the parameters on the surrogate network using KL-divergence as the loss function. We illustrate the value of this method empirically through a comparison on several evolutionary games.

1 Introduction

Spatial evolutionary game models have been widely used to model both biological and cultural evolution, e.g., (Durrett and Levin, 1994; Fu et al., 2010; Young, 2001; Sandholm, 2009; De et al., 2017) and a variety of multi-agent systems topics (Axtell, 2002; Tuyls and Parsons, 2007; Ponsen et al., 2009; Phelps et al., 2010; Morales et al., 2018). Most work on evaluating these games involves computationally expensive simulations, an approach that often does not scale well for spatial structures (Shakarian et al., 2012). Other limitations on validation (Herd et al., 2013) and variability (Manzo and Matthews, 2014) make it difficult to apply simulations towards the analysis of many evolutionary games. To address these limitations, moment closure methods such as pair approximation (Hauert and Szabó, 2005; Fu et al., 2010; Kuehn, 2016; Overton et al., 2019; Kuga et al., 2021) have been developed to approximate the dynamics of these games.

Symmetric Dynamic Bayesian Network Approximation In (Hsiao et al., 2021), a novel generalization of pair approximation using Bayesian networks was introduced to approximate the dynamics of spatial evolutionary games. That method, which we will call Symmetric Dynamic Bayesian Network Approximation (SD-BNA), takes advantage of the inherent symmetry in spatial Markov processes. Like pair approximation, the population dynamics of a spatial evolutionary game are analyzed by computing the transition probabilities of a local neighborhood of agents around an arbitrary focal agent. This computation is represented as a probabilistic inference problem on individual Bayesian networks that represent a single time-step of the local neighborhood. This allows for much more tractable approximation than a direct computation on the full population.

One of the advantages of the SD-BNA approach is that it is possible to adjust the size of the Bayesian network representing the local neighborhood to get more accurate approximations of the population dynamics. However, scaling up the method for more accurate results can still be intractable, as the induced width of the network depends directly on the size of the local neighborhood represented and the number of strategies in the evolutionary game. To address this issue, we propose a new method for computing approximate inference on larger Bayesian networks in SD-BNAs using what we will call *surrogate* Bayesian networks.

Contributions We propose a novel method for performing approximate inference on evolutionary game Bayesian networks.

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

- We introduce the concept of a surrogate Bayesian network, a subset of an evolutionary game Bayesian network with added parameters that exploit the symmetry inherent in these models.
- We formulate the process of learning the parameters for surrogate Bayesian networks as a KL-divergence minimization problem between the surrogate network and the original larger network.
- We introduce a novel sample-search algorithm for generating high value samples for the KL-divergence minimization problem.
- We provide an empirical comparison of our method with other approximate inference algorithms such as Abstraction Sampling on a variety of evolutionary games.

2 Related Work

Our work is similar to variational inference approaches. One can make the analogy that the smaller surrogate Bayesian network serves as the parameterized distribution in a variational inference approach and our goal is to minimize the KL-divergence between this parameterized distribution and the one we are approximating (the larger network). However, compared to previous work on variational Bayesian network approximation such as the edge-deletion approach in (Choi and Darwiche, 2006; Liu and Ihler, 2011; Ihler et al., 2012), a major difference is that our surrogate networks are always a strict subset of the network we want to approximate. This lets us formulate our optimization problem in a different way, since data points sampled from a larger network are full configurations of the smaller surrogate network. Our surrogate networks also contain parameter sharing by exploiting the symmetry present in SD-BNAs.

3 Background

3.1 Evolutionary Game Theory

Evolutionary Game Theory (EGT) provides a framework for modeling the time evolution of a population of agents that interact through strategic games whose outcome determines each individual's evolutionary fitness. These models disregard any game-theoretic assumptions of rationality and instead assume individuals reproduce or change strategies based on a predetermined population update rule. More concretely, consider a population of agents $\{X_1, ..., X_N\}$ that play an iterated stage game over a finite time horizon $t \in [0, ..., T]$. An evolutionary game is (S, U, F):

- $S = \{s_1, ..., s_M\}$: a set of M strategies where s_i is the i-th strategy in the set of possible strategies S and $\mathbf{s}(t = 0) = (s_1^0, ..., s_n^0)$ denotes the initial strategy profile which is a strategy assignment to each agent in the population.
- $U : S^N \to \mathbb{R}^N$: the payoff (or utility) function that maps the strategy profile to corresponding payoff values for each agent.
- $\Pr(\mathbf{s}^{t+1} = \mathbf{s}' \mid \mathbf{s}^t = \mathbf{s}) = F(\mathbf{s}', \mathbf{s}, U(\mathbf{s}))$: an update rule giving the transition probabilities given the current strategy assignment and payoff values.

Spatial Evolutionary Game Spatial evolutionary games are an extension of evolutionary games to model interactions among agents in a structured population. For example, agents may interact more frequently with agents in a neighborhood around them or the utility received from their interactions may be weighted accordingly to some network structure. These models are useful in examining how network structure influences the evolution of agent behavior. More concretely, a spatial evolutionary game is (S, U, F, G) where

- S, U, and F are analogous components to those in well-mixed evolutionary games, but they can now also depend on a spatial structure G.
- $G \in \{0, 1\}^{N \times N}$ is a spatial or network structure that specifies the strength of interactions between agents. An example payoff function is:

$$U_i(\mathbf{s}) = \sum_{j=1}^{N} G_{ij} \mathbf{Pay}[s_i, s_j] \tag{1}$$

Population Dynamics In non-spatial evolutionary games, it is common to refer to agents as *indistinguishable*, i.e., we don't care exactly which agent is playing which strategy. Instead we are typically interested in population-level aggregated quantities such as the population profile $\mathbf{p} = \{p_{s_1}, ..., p_{s_M}\}$, where p_{s_i} is the proportion of the population that is currently playing strategy s_i. One of the ways to visualize a population's time evolution is to plot these quantities as a function of time as in Fig. 4. In spatial games, the specific distributions of strategies (whether it is uniform or clustered) through the population can make a significant difference. To address this, we define higher-order aggregate quantities such as $p_{s_i s_j}$ which denotes the proportion of pairs of adjacent agents playing strategy s_i and s_j respectively. These second order quantities give an insight to the clustering behavior of different strategies in the population. For analysis on spatial games, it is then common to assume that pairs of agents are *indistinguishable*. While not completely accurate, it is necessary to make such assumptions to make sure the analysis of the population remains tractable.

3.2 Bayesian Networks

A Bayesian Network (Pearl, 1988) is a graphical model (X, D, F), consisting of a discrete variable set $X = \{X_1, X_2, \ldots, X_N\}$, a set of corresponding domains: $D = \{D_{X_1}, D_{X_2}, \ldots, D_{X_N}\}$, and a set of parent functions $F = \{F_1, F_2, \ldots, F_N\}$. Each X_i is associated with a parent function $F_i = P(X_i \mid pa_i)$ where pa_i is the set of parent variables of X_i.

Given a graphical model with variables $\{X_1, ..., X_N\}$, a common task is to compute the marginals, $P(X_i = s), \forall s$ in its domain for some variables X_i. A special case is computing a normalization constant known as the partition function:

$$Z = \sum_x \prod_i F_i(x) \tag{2}$$

where the $\sum_x$ refers to a summation over all possible configurations of the variables in the model. We can also define the partition function conditioned on certain variable assignments:

$$Z(X_1 = s) = \sum_{x|X_i=s} \prod_i F_i(x) \tag{3}$$

where the summation is over all configurations where $X_i = s$. In inference benchmarks, conditioned variables are termed *evidence*. To compute the marginal probability of a variable we can simply compute $Z(X_i = s)/Z$. One of the benefits of working with Bayesian networks is that we know that the partition function $Z = 1$ and so we can just compute $P(X_i = s) = Z(X_i = s)$.

Exact inference (e.g. computing the partition function) on a Bayesian Network is typically done using bucket elimination (aka variable elimination) or jointree algorithm (Darwiche, 2009; Dechter, 1999a, 2013) where messages of the form λ_{B_i}:

$$\lambda_{B_i} = \sum_{X_i} \prod_{F_j \in B_i} F_j \tag{4}$$

are computed from functions F_j placed in individual buckets B_i which are processed to eliminate X_i along a given variable ordering.

However, those algorithms can run into complexity issues where computing a message can take too much time/space. In these cases, many approximate inference techniques exist. One scheme is (Weighted) Mini-Bucket Elimination (WMBE) (Dechter and Rish, 2003; Liu and Ihler, 2011), an extension of bucket elimination, where the message in bucket elimination is replaced with a weighted geometric mean across smaller internal mini-bucket MB_i partitions in each bucket:

$$\lambda_{B_i} = \left(\sum_{X_i} \prod_{F_j \in MB_i} F_j^{\frac{1}{p_j}} \right)^{p_j} \tag{5}$$

The partition function can also be computed in a stochastic manner using sampling algorithms (Darwiche, 2009; Koller and Friedman, 2009). In this work, we will compare with more recent sampling algorithms such as Abstraction Sampling (Broka et al., 2018; Kask et al., 2020) which have been shown to be competitive on inference benchmarks. Other algorithms for evaluating Bayesian Networks optimization tasks can also be relevant (Marinescu and Dechter, 2009; Gogate and Dechter, 2011; Mateescu and Dechter, 2008).

3.3 Symmetric Dynamic Bayesian Network Approximations

A Symmetric Dynamic Bayesian Network Approximation (SD-BNA) (Hsiao et al., 2021) is an iterative method for approximating the forward dynamics of a spatially structured multi-agent population. Given a population profile $\mathbf{p}(t)$ at time t, each iteration of a SD-BNA takes the population profile to the next timestep $\mathbf{p}(t+1)$. It can be shown that a special case of this approximation is computationally equivalent to a pair approximation model, a common approximation technique in the evolutionary game literature (Hsiao et al., 2021). A primary advantage of a SD-BNA is that it allows for the exploration of higher order approximations beyond pair approximation which can yield better accuracy with respect to the underlying stochastic model (Hsiao et al., 2021).

SD-BNA for spatial Markov processes Consider a multivariate Markov process defined over N indistinguishable agents situated on a network G that take a strategy in the set S. We assume that the transition probability of an agent X_i depends locally on its neighborhood $Nb(X_i)$ for each iteration:

$$\begin{aligned} &P(X_i(t+1) = s \mid \mathbf{X}(1), ...\mathbf{X}(t-1)) \\ &= P(X_i(t+1) = s \mid X_i(t), X_j(t), \forall j \in Nb(X_i)). \end{aligned} \tag{6}$$

Following the conventions in (Hsiao et al., 2021), we can construct a Dynamic Bayesian Network (DBN) (Murphy, 2002) that completely captures the above Markov process. Each node in the DBN corresponds to an agent in the spatial evolutionary game. Because the DBN is highly symmetric, we get an additional benefit from the indistinguishability property: the marginal distributions $P(X = s) = P(Y = s)$ are identical for any two arbitrary agents X and Y. This property also applies to any pairs of adjacent agents:

$P(Nb(X)|X) = P(Nb(Y)|Y)$ where $Nb(X)$ is a neighboring adjacent agent of X and $Nb(Y)$ is a neighboring adjacent agent of Y. To simplify notation, we will refer to a single unindexed agent X (we can denote this as the focal agent, and the location of this agent is marked as a dark circle in Figures 3 and 5). We define the following population level aggregate terms:

$$p_{s_i} = P(X = s_i)$$
$$p_{s_i s_j} = P(X = s_i, Nb(X) = s_j) \quad (7)$$

At $t = 0$, we are given $p_{s_i}(0)$ and $p_{s_i s_j}(0)$ which is an initial distribution of the population. By *Symmetric Dynamic Bayesian network approximation* we refer to the following iterative method for approximating the time evolution of p_{s_i} and $p_{s_i s_j}$:

- Choose a representative subset of agents to be the *input neighborhood* $I \subset [1, ..., M]$ and a subset $O \subset I$ to be the *output neighborhood*. It is necessary for every node in O to have a fully defined transition probability conditioned on the nodes in I as in Eq. 6. For example, in the simplified diagram of Fig. 1, $I(T-1)$ consists of the three nodes at $T-1$ within the highlighted area and $O(T)$ consists of the the single node at time T within the highlighted area.
- Define the distribution of $I(t)$ using some function of $P(X_t)$ and $P(X_t, Nb(X)_t)$.
- Using the Markov process defined in Eq. 6, construct a Bayesian network B that transform the strategies of each agent in $I(t)$ to the agents at $O(t+1)$.
- Use a probabilistic inference algorithm (Darwiche, 2009; Dechter, 2013) such as Bucket Elimination (Dechter, 1999b) to query $p_{s_i}(t+1)$, $p_{s_i s_j}(t+1)$ for time $t+1$. These distributions are defined over the agents in the output $O(t+1)$.

To summarize, a separate 2 timestep Bayesian network is constructed for each iteration that takes the strategies of each agent from t to $t+1$. Using a probabilistic inference algorithm, we query the probabilities $P(X_{t+1})$ and $P(X_{t+1}, Nb(X)_{t+1})$ and we repeat the process at the next timestep by defining the distribution of the next input set $I(t+1)$ using the quantities we queried. More information is described in an earlier paper (Hsiao et al., 2021).

Symmetric Dynamic Bayesian Network Approximation as a truncation The SD-BNA can be considered to be a truncation approximation of the exact Dynamic Bayesian Network (DBN) for Eq. 6. We truncate the DBN temporally and spatially (blue and red respectively in Fig. 1). We approximate the distribution of $I(t+1)$ using quantities queried from $O(t+1)$ in the previous network. This can be thought of as a type of moment closure approximation (Kuehn, 2016). In previous work (Hsiao et al., 2021), the distributions are closed at the pair level and a tree approximation is used to model the distribution of input nodes at $t+1$. For example, if the input consists of 8 nodes $I = (A_1, ..., A_8)$ with A_1 being the focal node in the left network of Fig. 3, we can specify a tree approximation as in Fig. 2 where:

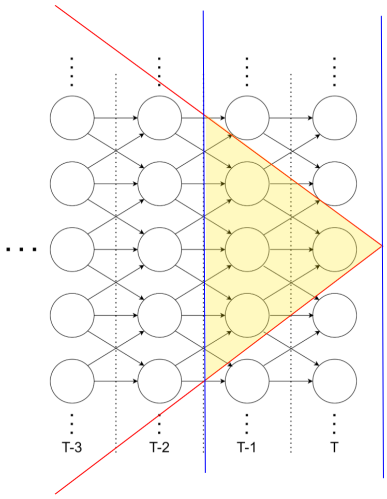

Figure 1: SD-BNA as a DBN truncation

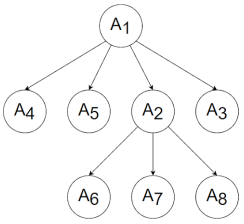

Figure 2: Tree approximation for input neighborhood definition

$$P(A_1 = s_i) = p_{s_i}$$
$$P(A_2 = s_i, A_1 = s_j) = p_{s_i s_j} \quad (8)$$

and so forth for each node in the tree.

4 Problem Statement

In Hsiao et al. (2021), it was shown that increasing the number of nodes in the input and output sets produce SD-BNAs that give more accurate approximations of the underlying dynamics. However, it is computationally expensive to evaluate large SD-BNAs. The induced width of the resulting Bayesian Network is proportional to the number of nodes in the input and

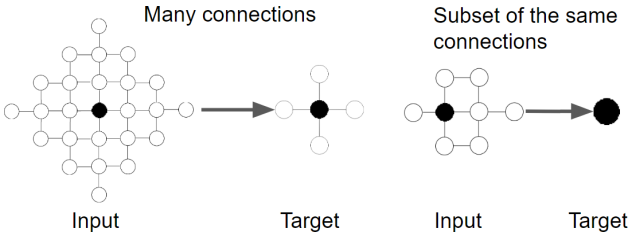

Figure 3: The input and output sets for a 25 node SD-BNA (left) and 8 node SD-BNA (right). Focal node is blackened.

output sets and the number of strategies in the evolutionary game is the base of the exponent (analagous to the domain size of each variable in the network).

One might consider a straightforward approach of applying approximate inference techniques to the larger SD-BNA models. An example of a state-of-the-art approximate inference technique is Abstraction Sampling (AS) (Broka et al., 2018; Kask et al., 2020). We can apply AS instead of Bucket Elimination as our inference routine to compute $p_{s_i s_j}(t+1)$.

However, as can be seen in the right most graph in Fig. 4 on the Deadlock evolutionary game, simply replacing exact inference with Abstraction Sampling runs into several limitations. Specifically:

- Accuracy: It can take many samples for AS on a 25 node network to produce better results compared to exact inference on an 8 node network.
- Time: Running exact inference on 8 node network is much faster than AS on a 25 node network.
- Noise: Without sufficient sample convergence in AS, it is difficult to determine certain quantities such as the location of inflection points, which can impact qualitative results.

To address these issues, we propose a method that combines the estimate from the smaller network with estimates from the larger network.

5 Surrogate Maximum Likelihood Models

Consider an 8 node SD-BNA and a 25 node SD-BNA for approximating a spatial evolutionary game located on a grid structure (each agent has four neighbors). For the purpose of this paper, we will only focus on 8 and 25 node SD-BNAs (the number of nodes referring to the size of the input set including the focal node as in Fig. 3), but in principle this work can also be extended to other pairs of SD-BNAs with the smaller network serving as the surrogate of the larger network.

A key observation we can make is that the smaller 8 node network is a subset of the larger 25 node network. Our idea is to extend the 8 node SD-BNA with additional parameterized nodes/connections so it can approximate the 25 node network. We first give an initial explanation of the basic idea of our approach (termed the basic MLE algorithm) in Section 5.1. In Section 6, we develop an improved algorithm termed KL-search for inference by devising a sample/search method for generating informative samples for the basic MLE algorithm.

5.1 Parameterizing the new edges

In the current setting of the 8 node network we approximate the query $p_{s_i s_j}$ as $\approx P(X_{t+1}, Nb(X)_t)$, when we really want $P(X_{t+1}, Nb(X)_{t+1})$. In order to increase accuracy we propose to add a new set of 4 dummy nodes $Nb(X)_{t+1}$ that capture the output neighborhood. We add new connections (edges) from the input neighborhood to the output neighborhood as seen in Fig. 5. For these new connections, we define these new connections as a simple conditional probability table (CPT) conditioned on two nodes in the input neighborhood. In essence, we add new CPTs of the form $\theta_{x,y,z} = p_{x|yz}$. Since we inherit the assumption that all neighboring nodes are indistinguishable from 25 node network, all CPTs are identical and we can share parameters across them.

Given a 25 node SD-BNA model and an 8 node extended model, the **basic MLE algorithm** for learning the extended 8-node network is as follows:

1. Forward sample the 25 node SD-BNA model for a full configuration of every variable in the 25 node network. Denote each full configuration as an individual data point $\mathbf{x}^j$.
2. The 8-node network is fully observed for each data since its variables are subsumed in the large network we can derive the likelihood expression:

$$\begin{aligned}\mathcal{L}(\theta) &= \prod_{j=1}^{L} P_\theta(\mathbf{x}^j) \\ &= \prod_{j=1}^{L} P_\theta(Nb(X^{(j)})_{t+1} | X_t^{(j)}, Nb(X^{(j)})_t) \cdot C \\ &= \prod_{j=1}^{L} \theta(Nb(X^{(j)})_{t+1}, X_t^{(j)}, Nb(X^{(j)})_t) \cdot C \end{aligned} \tag{9}$$

where $X_t^{(j)}$ denotes the assignment to X for the j-th data point and at time t. C is some constant that doesn't depend on θ and can be ignored when

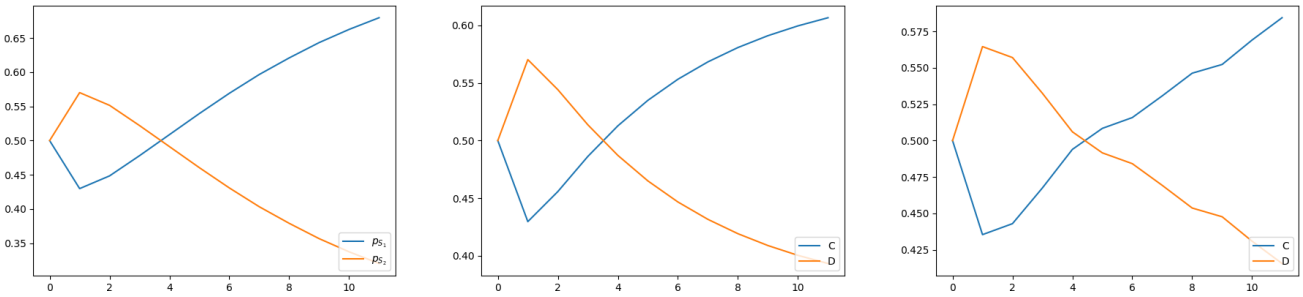

Figure 4: Simulation ground truth (left), compared with exact 8 node SD-BNA result (center), and approximate Abstraction Sampling result on 25 node SD-BNA (right) on the Deadlock evolutionary game.

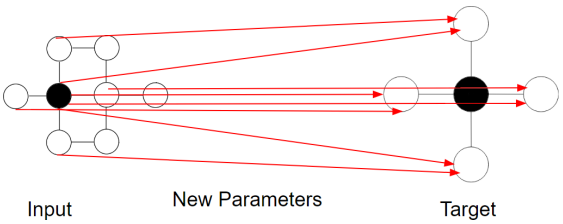

Figure 5: New parameters (edges) to add to 8 node network for MLE estimation.

maximizing the corresponding sum of log likelihoods.

3. Maximize the log likelihood $\log \mathcal{L}(\theta)$ subject to symmetry constraints:

$$p_{s_i}(t+1) = \sum_{s_j, s_k \in S} \theta_{s_i,s_j,s_k} \cdot p_{s_j s_k}(t), \quad \forall s_i \in S \tag{10}$$

We can pre-compute $p_{s_i}(t+1)$ on a non-parameter extended 8 node network and use this to constrain our log likelihood maximization derived from Eq. 9:

$$\theta^* = \arg\max_{\theta} \sum_{j=1}^{L} \log \theta(Nb(X^{(j)})_{t+1}, X_t^{(j)}, Nb(X^{(j)})_t)$$

subject to

$$p_{s_i}(t+1) = \sum_{s_j, s_k \in S} \theta_{s_i,s_j,s_k} \cdot p_{s_j s_k}(t), \quad \forall s_i \in S \tag{11}$$

This can be solved with existing constrained maximization solvers such as sequential least squares Kraft (1988) or trust-region algorithms which can be found in the SciPy python package (Virtanen et al., 2020).

Once θ^* is learned, we can evaluate the forward dynamics (compute $p_{s_i s_j}(t+1)$) on the optimal extended network B_{θ^*}.

6 KL-based Search Tree Exploration

In the previous section we used Monte Carlo forward sampling to generate the samples. **Key idea: instead of drawing random samples, we will combine sampling with search aiming for diverse samples that are likely to reduce the KL-divergence between our surrogate model (the 8 node model) and the model we are approximating (the 25 node model).**

From this point onward, we'll refer to the the 25 node model as model A and the 8 node model as model B for simplicity. Furthermore, we will assume bi-valued variables and use P_A, and P_B to denote distributions on model A and model B, respectively, and use P_{B_θ} for the distribution of the extended model B parameterized by θ.

Informative sample generation In order to guarantee that all the samples are different and aim towards meaningful samples we generate a fixed number L of partial configurations from the large network A using best-first search along some variable ordering guided by a heuristic function. For example, the first two nodes that are generated are associated with the root variable X_1 and represent the partial configurations $(X_1 = 0)$ and $(X_1 = 1)$. In the best-first search, we want a heuristic that will explore branches of high difference between the models. To do this, we define a KL heuristic for the partial configuration $(X_1 = 0)$ as:

$$h_{kl}(X_1{=}0) = [\log(P_A(X_1{=}0)) - \log(P_B(X_1{=}0))] \cdot P_B(X_1{=}0) \tag{12}$$

Once we have L leaf nodes (representing a partial configuration) we extend each to a full configuration by forward sampling the rest of the variables using the 25 node network yielding a data point $\mathbf{x}^k$ conditioned on the node's partial configuration.

Since it is difficult to directly compute $P_A(X_1 = 0)$ and $P_B(X_1 = 0)$ in the KL heuristic, we approximate

these terms using a weighted mini-bucket elimination (WMBE) heuristic (Liu and Ihler, 2011) with an i-bound of 10 (this is just an estimate of our Bayesian network's partition function conditioned on a node's partial configuration).

Once we have L full samples, we solve for θ^* that minimizes the following loss function:

$$\theta^* = \arg\max_{\theta} \sum_{j=1}^{L} \log P_{B_\theta}(\mathbf{x}^j) \cdot P_A(\mathbf{x}^j)$$

subject to

$$p_{s_i}(t+1) = \sum_{s_j, s_k \in S} \theta_{s_i, s_j, s_k} \cdot p_{s_j s_k}(t), \quad \forall s_i \in S \quad (13)$$

with an additional term $P_A(\mathbf{x}^k)$ where each θ expression is also multiplied with the probability of the sample $P(\mathbf{x}^k)$. We can justify this as the loss function because at the limit of large L, this loss function is equivalent to minimizing KL divergence. This is formally stated in the following Theorem. The full **KL-Search** algorithm is shown in Algorithm 1.

Theorem 6.0.1 (Asymptotic Convergence of KL-Search Minimization)**.** *Let θ_L be the result of KL-Search Minimization [Algorithm 1] given L samples. Then given a family of extended networks B_θ parameterized by θ:*

$$\lim_{L \to \infty} \theta_L = \arg\min_{\theta} D_{KL}(P_A || P_{B_\theta}) \quad (14)$$

Proof [See Supplemental]

6.1 Scaling up to more strategies

To scale up the method to handle evolutionary games with more strategies, the major computational bottleneck is in computing $P_A(X_1 = 0)$ in Eq. 12 using our WMBE approximation. While we can feasibly evalute this quantity in 25 node networks having 3 or less strategies, WMBE heuristics for approximating $P_A(X_1 = 0)$ become difficult to evaluate directly in 25 node networks with 4 or more strategies. Therefore, instead of computing $P_A(X_1 = 0) \approx WMB(X_1 = 0)$, we propose an easier to compute heuristic such as:

$$P_A(X_1 = 0) \approx \hat{Z}_A(X_1 = 0) \quad (15)$$

where $\hat{Z}_A$ is a stochastic estimate of the partition function conditioned on $(X_1 = 0)$:

$$\hat{Z}_A(X_1 = 0) = \frac{1}{N} \sum_{i=1}^{N} \hat{Z}_A(x^{(i)}), \; x^{(i)} \sim P_A(x | X_1 = 0) \quad (16)$$

We propose a quick estimate for the partition function to be evaluated from a single sampled configuration ($N = 1$). We'll call this new method **Fast KL-search**.

Algorithm 1: KL-Search Minimization

Input: Two Bayesian networks: a large network A, a smaller network B, and a parameterized extended network B_θ such that all nodes in B are in A ($B \subset A$), a variable ordering o over A, initial distribution $p_{yz}(t)$, and pre-computed output distribution $p_x(t+1)$

Parameters: Number of samples L

Output: θ, estimated parameters that minimize difference between A and B_θ

$T \leftarrow$ the OR-search tree on A using ordering o;
$OPEN \leftarrow \{\langle root(T), 0 \rangle\}$;
`// frontier nodes are ordered by the 2nd value`
for $i = 1 \rightarrow L$ **do**
  $v \leftarrow OPEN$.dequeue() ; `// remove the node of highest priority`
  for $u \in children(v)$ **do**
    $h_{kl}(u) \leftarrow [\log(P_A(u)) - \log(P_B(u))] \cdot P_B(u)$;
    Append $< u, h_{kl}(u) >$ to $OPEN$;
  end
end
Let X be an empty list;
for $v \in OPEN$ **do** `// leaf nodes`
  Forward sample x, a full configuration of A conditioned on the partial configuration represented by v;
  Append x to X;
end
Solve $\theta^* = \arg\max_\theta \sum_{j=1}^{L} \log P_{B_\theta}(\mathbf{x}^j) \cdot P_A(\mathbf{x}^j)$, subject to
$p_{s_i}(t+1) = \sum_{s_j, s_k \in S} \theta_{s_i, s_j, s_k} \cdot p_{s_j s_k}(t), \quad \forall s_i \in S$;
Return θ^*;

6.2 Learning and Inference

KL-Search (Algorithm 1) and Fast KL-search (Algorithm 1 with a simplified heuristic) both work over one timestep of the evolutionary game. The θ^* obtained as the return value from these algorithms minimizes the kl divergence between models for a single timestep. The optimal θ value can change each time iteration and is dependent on the current values of $p_{s_i}(t)$ and $p_{s_i s_j}(t)$. Thus the full algorithm for our approach has two parts: learning (KL-Search) followed by inference (SD-BNA). Our goal in the empirical results section is to determine what methods reduce the single iteration error the most, so we will use the measure $D_{KL}(P_{sim}(X_1 | Nb(X)_1) || P_{method}(X_1 | Nb(X)_1))$ evaluated at $t = 1$ where P_{sim} is the probability estimated using the average of many simulations and P_{method} is the probability estimated using a given method. Finding the most effective intervals to interleave learning with inference will be future work.

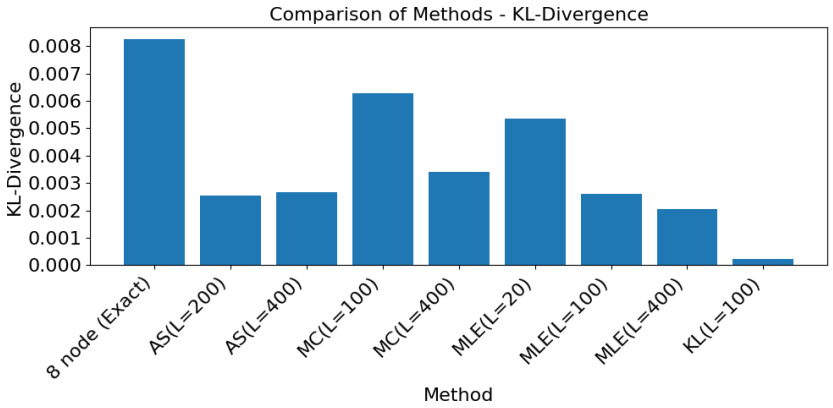

Figure 6: Sample efficiency comparison (measured using average KL divergence) on the Deadlock evolutionary game. Results are averaged across 10 separate runs for each method

7 Empirical Results

The central task of our empirical evaluation is estimating $P(X_{t+1}|Nb(X)_{t+1})$ to compute $p_{s_i s_j}$ at each time step in a SD-BNA. Therefore, we focus on evaluating the accuracy of different approaches for estimating these quantities. For this, we will use the divergence measure $D_{KL}(P_{sim}(X_1|Nb(X)_1)||P_{method}(X_1|Nb(X)_1))$ (as mentioned in Section 6.2) to evaluate the effectiveness of different approximate inference algorithms.

For our comparisons in Fig. 6, 7, and 8, we test our algorithms on a variety of SD-BNAs constructed from different evolutionary games with KL-divergence results. In each of these experiments, the goal is to minimize the KL-divergence with respect to the ground truth estimate from the simulation.

For an initial comparison in Fig. 6, we use the Deadlock game (see Table 2) from Fig. 4. The methods used in the initial comparison include:

- 8 node (exact): the estimate obtained from exact inference on the 8 node network.
- AS: Abstraction Sampling on a 25 node SD-BNA with (200, 400) samples.
- MC: Forward sampling on a 25 node SD-BNA with (100, 400) samples.
- MLE: The surrogate network approach from Section 5 with (20, 100, 400 data points) without doing KL-search to generate the data points.
- KL: The estimate obtained using θ^* from Algorithm 1.

In the initial tests on the DLK game, the KL-search approach outperforms all of the other methods in terms of KL-divergence.

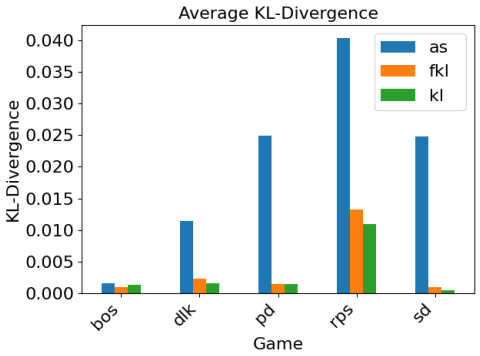

Figure 7: Average KL divergence between simulation and [abstraction sampling (as), KL-search using WMBE (kl), and KL-search using a single configuration sample (fkl)].

7.1 Comparison of Methods

We compare KL-search (Algorithm 1), Fast KL-search (Algorithm 1 with Eq. 15), and Abstraction Sampling using the same average KL divergence metric as in Fig. 6. For these set of tests, we average the results of 30 separate runs for each game and allow each method to generate 100 samples for each run.

The main results are displayed in Fig. 7, which compares the three methods across 5 different evolutionary games (Table 2). Both the Fast KL-search (denoted fkl in the figure) and the regular KL-search with WMBE heuristic perform much better than Abstraction Sampling in all games.

Larger Games The only method that can run on games with more than 3 strategies is Fast KL-Search. As mentioned earlier, with 4 or more strategies, the WMBE heuristic is hard to compute because of its inherent table representation of the CPTs which become

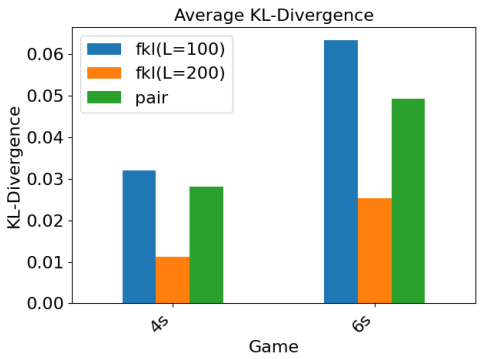

Figure 8: Average KL divergence between simulation and [pair approximation (pair), and KL-search using a single configuration sample (Fast KL-search) at 100 and 200 samples (fkl(L=100), fkl(L=200))] on games with a high number of strategies.

too large. in Fig. 8, we show a comparison with the method known as pair approximation (Fu et al., 2010). We observed that in these larger games, more than 100 samples are needed for Fast KL-Search to outperform pair approximation.

Time Comparison A comparison of time per probe/sample for the experiments from Fig. 7 and 8 is shown in Table 1. For example, for $|S| = 2$ on KL-Search, we average the time taken for the experiment on the 2 strategy (bos, dlk, pd, and sd) games and divide by 4 (games)·30 (runs)·100 (samples).

Fast KL-Search actually takes more time than normal KL-Search, as the time for the latter is frontloaded in computing the initial WMBE tree. For individual samples, the WMBE heuristic takes less time than evaluating an entire sample as in Fast KL-Search. As expected, both methods take much less time than Abstraction Sampling.

One sample in AS is a full probe that expands $O(d \cdot h)$ nodes in a search tree where d is the number of abstract states and h is the tree's depth. This takes $O(d \cdot h \cdot L)$ time where L is the number of samples. For KL-Search and Fast KL-Search, only L nodes in the search tree need to expanded to get L samples, so the final time is just $O(L)$. The following table summarizes this:

AS	KL-Search	Fast KL-Search
$O(d \cdot h \cdot L) \cdot O(1)$	$O(L) \cdot O(1)$	$O(h) \cdot O(L)$

8 Conclusion

We have introduced a novel approach for performing approximate inference on evolutionary game Bayesian networks. Our method, based on optimizing parameterized surrogate Bayesian Networks, leverages the symmetry present in SD-BNAs to combine the efficient computation of smaller SD-BNAs with the more accurate results of larger SD-BNAs. We introduce a novel sample-search approach to generate high value samples for this optimization problem, and empirically demonstrate its effectiveness when compared to existing approximate inference techniques such as Abstraction Sampling.

For future work, it may be interesting to extend this technique to domains beyond spatial evolutionary games. It should be possible to apply SD-BNAs to any general spatial Markov process with a high degree of symmetry. Other spatial Markov processes such as voter models and SIS models could be a valid area of application. It may also be interesting to see if this technique could be applied to general Bayesian network inference on networks with a high degree of structure.

Table 1: Time per sample (s) with amortized initial (WMBE for AS/KL) and post-processing (KL minimization for KL/FKL) time (each time is computed by averaging over at least 3000 samples from the experiments for Fig. 7 and 8).

$\|S\|$	AS	KL-Search	Fast KL-Search
2	0.1175	0.01419	0.01624
3	0.2203	0.03063	0.03725
4	-	-	0.1630
6	-	-	0.7299

Table 2: Evolutionary Game Payoff Matrices

Game Name	Payoff Matrix
Prisoner's Dilemma (pd)	$\begin{pmatrix} 2 & -1 \\ 3 & 0 \end{pmatrix}$
Snowdrift (sd)	$\begin{pmatrix} 2 & 1 \\ 3 & 0 \end{pmatrix}$
Battle of the Sexes (bos), symmetric version, (Cooper et al., 1989)	$\begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}$
Deadlock (dlk)	$\begin{pmatrix} 5 & -4 \\ 3 & -5 \end{pmatrix}$
Rock Paper Scissors (rps)	$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

Acknowledgements

This work supported in part by NSF grant IIS-2008516, NSF-2321786 and AFOSR grant 1010GWA357.

References

Axtell, R. L. (2002). Non-cooperative dynamics of multi-agent teams. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1082–1089.

Broka, F., Dechter, R., Ihler, A., and Kask, K. (2018). Abstraction sampling in graphical models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Choi, A. and Darwiche, A. (2006). A variational approach for approximating bayesian networks by edge deletion. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 80–89.

Cooper, R., DeJong, D. V., Forsythe, R., and Ross, T. W. (1989). Communication in the battle of the sexes game: some experimental results. *The RAND Journal of Economics*, pages 568–587.

Darwiche, A. (2009). *Modeling and reasoning with Bayesian networks*. Cambridge university press.

De, S., Nau, D. S., and Gelfand, M. J. (2017). Understanding norm change: An evolutionary game-theoretic approach. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1433–1441.

Dechter, R. (1999a). Bucket elimination: A unifying framework for reasoning. *Artif. Intell.*, 113(1-2):41–85.

Dechter, R. (1999b). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85.

Dechter, R. (2013). Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191.

Dechter, R. and Rish, I. (2003). Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153.

Durrett, R. and Levin, S. (1994). The importance of being discrete (and spatial). *Theoretical population biology*, 46(3):363–394.

Fu, F., Nowak, M. A., and Hauert, C. (2010). Invasion and expansion of cooperators in lattice populations: Prisoner's dilemma vs. snowdrift games. *Journal of theoretical biology*, 266(3):358–366.

Gogate, V. and Dechter, R. (2011). Samplesearch: Importance sampling in presence of determinism. *Artif. Intell.*, 175(2):694–729.

Hadjichrysanthou, C., Broom, M., and Kiss, I. Z. (2012). Approximating evolutionary dynamics on networks using a neighbourhood configuration model. *Journal of theoretical biology*, 312:13–21.

Hart, S. and Mas-Colell, A. (2001). A general class of adaptive strategies. *Journal of Economic Theory*, 98(1):26–54.

Hauert, C. and Szabó, G. (2005). Game theory and physics. *American Journal of Physics*, 73(5):405–414.

Herd, B., Miles, S., McBurney, P., and Luck, M. (2013). Verification and validation of agent-based simulations using approximate model checking. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 53–70. Springer.

Hsiao, V., Pan, X., Nau, D., and Dechter, R. (2021). Approximating spatial evolutionary games using bayesian networks. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1533–1535.

Ihler, A., Flerova, N., Dechter, R., and Otten, L. (2012). Join-graph based cost-shifting schemes. In de Freitas, N. and Murphy, K. P., editors, *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pages 397–406. AUAI Press.

Kask, K., Pezeshki, B., Broka, F., Ihler, A., and Dechter, R. (2020). Scaling up and/or abstraction sampling. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,{IJCAI} 2020*.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Kraft, D. (1988). A software package for sequential quadratic programming. Technical report, DLR German Aerospace Center – Institute for Flight Mechanics.

Kuehn, C. (2016). Moment closure—a brief review. In *Control of self-organizing nonlinear systems*, pages 253–271. Springer.

Kuga, K., Tanaka, M., and Tanimoto, J. (2021). Pair approximation model for the vaccination game: predicting the dynamic process of epidemic spread and individual actions against contagion. *Proceedings of the Royal Society A*, 477(2246):20200769.

Liu, Q. and Ihler, A. T. (2011). Bounding the partition function using holder's inequality. In *ICML*.

Manzo, G. and Matthews, T. (2014). Potentialities and limitations of agent-based simulations. *Revue française de sociologie*, 55(4):653–688.

Marinescu, R. and Dechter, R. (2009). Memory intensive AND/OR search for combinatorial optimization

in graphical models. *Artif. Intell.*, 173(16-17):1492–1524.

Mateescu, R. and Dechter, R. (2008). Mixed deterministic and probabilistic networks. *Ann. Math. Artif. Intell.*, 54(1-3):3–51.

Morales, J., Wooldridge, M., Rodríguez-Aguilar, J. A., and López-Sánchez, M. (2018). Off-line synthesis of evolutionarily stable normative systems. *Autonomous agents and multi-agent systems*, 32(5):635–671.

Murphy, K. P. (2002). *Dynamic bayesian networks: representation, inference and learning*. University of California, Berkeley.

Overton, C. E., Broom, M., Hadjichrysanthou, C., and Sharkey, K. J. (2019). Methods for approximating stochastic evolutionary dynamics on graphs. *Journal of Theoretical Biology*, 468:45–59.

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

Phelps, S., McBurney, P., and Parsons, S. (2010). Evolutionary mechanism design: a review. *Autonomous agents and multi-agent systems*, 21(2):237–264.

Ponsen, M., Tuyls, K., Kaisers, M., and Ramon, J. (2009). An evolutionary game-theoretic analysis of poker strategies. *Entertainment Computing*, 1(1):39–45.

Sandholm, W. H. (2009). Evolutionary game theory. In *Encyclopedia of Complexity and Systems Science*, pages 3176–3205. Springer.

Shakarian, P., Roos, P., and Johnson, A. (2012). A review of evolutionary graph theory with applications to game theory. *Biosystems*.

Traulsen, A. and Hauert, C. (2009). Stochastic evolutionary game dynamics. *Reviews of nonlinear dynamics and complexity*, 2:25–61.

Tuyls, K. and Parsons, S. (2007). What evolutionary game theory tells us about multiagent learning. *Artificial Intelligence*, 171(7):406–416.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Young, H. P. (2001). *Individual strategy and social structure: An evolutionary theory of institutions*. Princeton University Press.

Surrogate Bayesian Networks for Approximating Evolutionary Games: Supplementary Materials

S1 Markov Process to Game dynamics

Given an evolutionary game, we use the *indistinguishable* property of agents to define a Markov process over the population profile. For example, for a system of N agents and 2 strategies $\{A, B\}$, each state in the Markov process is defined as the number of agents playing strategy A (with the number of agents playing strategy $B = N - A$). Using the master equation on this population profile Markov process (Traulsen and Hauert, 2009), we can derive a set of differential equations that represent the time evolution of the population profile.

In spatial evolutionary games, we can follow a similar process to define a Markov process over higher order population profile terms. However, in a spatial model the time evolution of a single node will depend on its neighbor or the pair distribution and the time evolution of the pair distribution depends on the triplet distribution and higher. Therefore, when you apply the master equation, what we get is a set of differential equations defined with terms defined up to the size of the total population:

$$\begin{aligned} p_i(t+1) &= p_i(t) + F(p_i(t), p_{ij}(t)) \\ p_{ij}(t+1) &= p_{ij}(t) + G(p_i(t), p_{ij}(t), p_{ijk}(t)) \\ p_{ijk}(t+1) &= p_{ijk}(t) + H(p_i(t), p_{ij}(t), p_{ijk}(t), p_{ijkl}(t)) \\ &\vdots \end{aligned} \tag{S1}$$

Clearly this is not tractable for large enough population sizes, and thus many approximation methods such as pair approximation have been developed where all equations above a certain order are ignored. For example in pair approximation, the equations are limited to 2nd order equations (p_{ij} or below) and p_{ijk} would be approximated with some combination of lower order terms (for example $p_{ijk} \approx \frac{p_{ij}p_{jk}}{p_j}$).

Table S1: Algorithm Complexities

Algorithm	Time Complexity	Space Complexity
Abstraction Sampling	$O(d \cdot h \cdot L) \cdot O(1) + O(WMBE)$	$O(\|D\|^{ibound})$
KL-Search	$O(L) \cdot O(1) + O(WMBE)$	$O(\|D\|^{ibound})$
Fast KL-Search	$O(h) \cdot O(L)$	$O(h)$
Weighted Mini-Bucket Elimination (WMBE)	$O(\|D\|^{ibound})$	$O(\|D\|^{ibound})$

S2 Additional Algorithm and Model Details

The time/space complexities of each algorithm are listed in Table S1. For constants in the table:

- d: number of abstract states in the Abstraction Sampling algorithm.
- h: maximum height of the search tree, corresponds to number of variables in the Bayesian network
- L: number of samples
- $|D|$: size of variable domains in the Bayesian network
- *ibound*: parameter used in Weighted Mini-Bucket Elimination

The settings used for the 8 and 25 node Symmetric Dynamic Bayesian Network Approximations are listed in Table S2. We make the following assumptions for the evolutionary games we are modeling:

- We assume that the spatial evolutionary game takes place on a square lattice, but in practice you can define a SD-BNA for any spatial network. In the case of more complex networks such as heterogeneous networks, it is necessary to use a different base approximation (the tree approximation is based on pair approximation for example) for the input neighborhood such as a neighborhood configuration approximation (Hadjichrysanthou et al., 2012), but this lies outside the scope of this work.
- We assume that the update rule is the Fermi rule (Hart and Mas-Colell, 2001). Each agent obtains a payoff by summing the payoff received from playing a normal form game with each of its neighbors. The probability that an agent will switch to a neighbors strategy would be:

$$P(X_{t+1} = s' | X_t, N_t = s', Pay(N)_t = \pi', Pay(X)_t = \pi) = \frac{1}{(1 + e^{-s(\pi' - \pi)})} \tag{S2}$$

  where $Pay(X)$ is the payoff of the agent and $Pay(N)$ is the payoff of a randomly chosen neighbor $N \in \{Nb_1(X), ..., Nb_4(X)\}$. In an 8 node SD-BNA, the input neighborhood is only large enough to compute the payoff of one neighbor. To solve this we assume, without loss of generality, that the focal agent X would always learn from the first neighbor $Nb_1(X)$. This is an assumption made in many pair approximations models ((Fu et al., 2010)).

To define the input neighborhood for each model, we first start with a focal node and four neighboring nodes $Nb_1(X)...Nb_4(X)$. The +3 and +20 quantifiers refer to 3 or 20 surrounding nodes that can be seen in Fig. 3 of the main paper. The Bayesian network for a 8 node SD-BNA can be seen in Fig. S2 which corresponds to an input neighborhood labeled in S1.

To query for the next iteration in the 8 node SD-BNA, we compute $p_{ij} = P(X'|Nb(X))$ where $Nb(X)$ is a randomly selected neighbor at time t. In actual implementation, we can add additional selection/dummy nodes to simulate the random selection and query the joint distribution of the focal node with the dummy node. A 25 node SD-BNA would have a similar construction with far more Y nodes representing 2nd degree and 3rd degree neighbors. For the exact model and CPT values, please refer to the full version of (Hsiao et al., 2021).

Table S2: Symmetric Dynamic Bayesian Network Approximation settings

	8 node SD-BNA	25 node SD-BNA
Input	$X, Nb_1(X), ...Nb_4(X), +3$	$X, Nb_1(X), ...Nb_4(X), +20$
Output	X'	$X', Nb_1(X)', ...Nb_4(X)'$
Query	$p_i = P(X')$ $p_{ij} = P(X'\|Nb(X))$	$p_i = P(X')$ $p_{ij} = P(X'\|Nb(X)')$

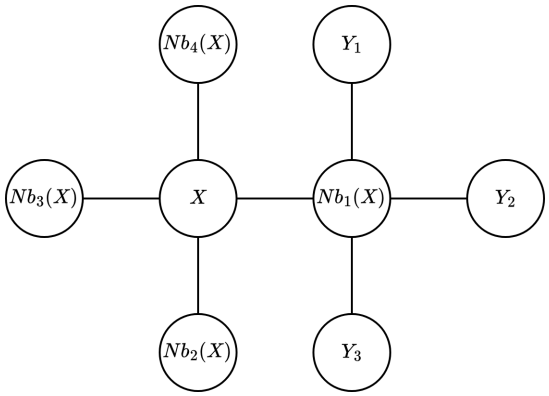

Figure S1: Labeled input neighborhood for 8 node SD-BNA

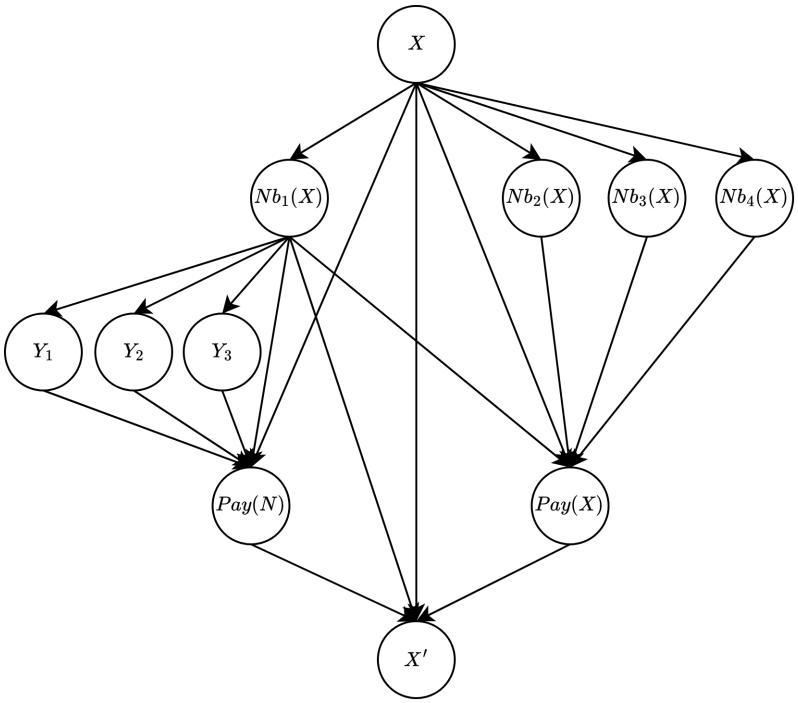

Figure S2: Example (simplified) Bayesian network corresponding to an 8 node SD-BNA for an evolutionary game

S3 Proof of Theorem 6.0.1

Theorem 6.0.1 (Asymptotic Convergence of KL-Search Minimization)**.** *Let θ_L be the result of KL-Search Minimization [Algorithm 1 in the main paper] given L samples. Then for a given family of extended networks B_θ parameterized by θ:*

$$\lim_{L\to\infty} \theta_L = \arg\min_\theta D_{KL}(P_A||P_{B_\theta}) \tag{S3}$$

Proof Since samples are generated from leaf nodes who are guaranteed to be different, it implies that no samples are identical. If the search tree is fully expanded, our optimization problem becomes:

$$\begin{aligned}\theta^* &= \arg\max_\theta \sum_{X\in A} \log P_{B_\theta}(X)\cdot P_A(X)\\ &= \arg\max_\theta \mathbb{E}_{X\sim P_A}[\log P_{B_\theta}(X)]\end{aligned} \tag{S4}$$

which is just maximizing the log likelihood. With some additional derivation:

$$\begin{aligned}\theta^* &= \arg\max_\theta \sum_{X\in A} \log P_{B_\theta}(X)\cdot P_A(X)\\ &= \arg\max_\theta \sum_{X\in A} \log P_{B_\theta}(X)\cdot P_A(X) - \log P_A(X)\cdot P_A(X)\\ &= \arg\max_\theta \sum_{X\in A} \log \frac{P_{B_\theta}(X)}{P_A(X)}\cdot P_A(X)\\ &= \arg\min_\theta \sum_{X\in A} -\log \frac{P_{B_\theta}(X)}{P_A(X)}\cdot P_A(X)\\ &= \arg\min_\theta \sum_{X\in A} \log \frac{P_A(X)}{P_{B_\theta}(X)}\cdot P_A(X)\\ &= \arg\min_\theta D_{KL}(P_A(X)||P_{B_\theta}(X))\end{aligned} \tag{S5}$$

it can be shown that maximizing the expectation $\mathbb{E}_{X\sim P_A}[\log P_{B_\theta}(X)]$ is equivalent to minimizing the KL-divergence $D_{KL}[P_A(X)||P_B(X)]$, so our search procedure is guaranteed to eventually find the 8 node parameterized model having minimum KL-divergence as the search tree approaches full expansion.

S4 Additional Computational Details

Optimization In practice it is difficult for black-box solvers to satisfy the probability constraint in Eq. 9 in the main paper. This is even more of an issue in games with a larger number of strategies. To address this, we turn the hard constraint into a soft constraint:

$$\begin{aligned}\theta^* = \arg\max_\theta \sum_k^L \log P_{B_\theta}(\mathbf{x}^k)\cdot P_A(\mathbf{x}^k) +\\ C\cdot\sum_x \left[p_x(t+1) - (\sum_{y,z\in S}\theta_{x,y,z}\cdot p_{yz}(t).\right]^2\end{aligned} \tag{S6}$$

For our empirical tests, we solve this optimization problem with Sequential Least Squares Programming (SLSQP) through the minimize routine in the SciPy python library (Kraft, 1988; Virtanen et al., 2020).

Computation In the KL-Search algorithm, we can reduce the $P_{B_\theta}(X)$ to just the θ expression. Since model B is a Bayesian network and θ only controls the CPTs that we added, we can decompose $P_{B_\theta}(X) = \theta(X)\cdot P'_B(X)$,

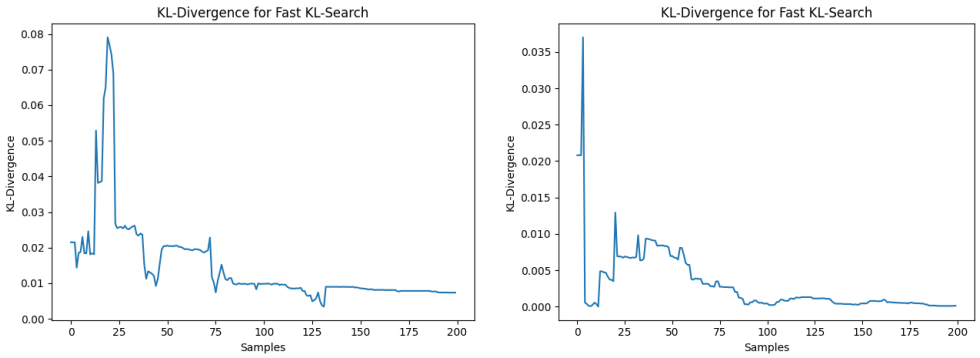

Figure S3: KL divergence for KL-search using a single configuration (Fast KL-search) on RPS (left) and Deadlock (right)

such that $P'_B(X)$ does not depend on θ.

$$\begin{aligned}\theta^* &= \arg\max_\theta \sum_k^N \log P_{B_\theta}(\mathbf{x}^j) \cdot P_A(\mathbf{x}^j) \\ &= \arg\max_\theta \sum_j^N (\log \theta(\mathbf{x}^j) + \log P'_B(\mathbf{x}^j)) \cdot P_A(\mathbf{x}^j) \\ &= \arg\max_\theta \sum_k^N \log \theta(\mathbf{x}^j) P_A(\mathbf{x}^j) \end{aligned} \tag{S7}$$

yielding a a simple constrained optimization problem. The intuition for using a best-first search guided by the KL-heuristic is that we are greedily generating samples in regions of high KL-distance in order to get informative samples.

Computational Resources All experiments in this work are evaluated on a 64-bit machine with an Intel i7-10870H 2.2 GHz CPU and 32 GB of RAM. No GPU resources are used in our experiments.

Abstraction Sampling In the main paper we compare against the use of Abstraction Sampling to estimate $P(X_{t+1}|Nb(X)_{t+1})$. Since Abstraction Sampling is an algorithm for estimating the partition function, we use it query the partition function conditioned on the partial configuration $(X_{t+1} = s_i, Nb(X)_{t+1} = s_j)$ for every pair of values $s_i, s_j \in S$. This necessitates splitting up the sampling procedure across $O(|S|^2)$ different sample sets, one set for each value pair. To improve the quality (e.g. reduce the variance) of our final result, after doing an initial 5 samples for each configuration of $(X_{t+1}, Nb(X)_{t+1})$, we perform adaptive variance sampling. To do this we calculate the variance of the samples currently taken for each configuration of $(X_{t+1}, Nb(X)_{t+1})$ and the next sample taken will be for the configuration with the highest sample variance. We stop this process once we reach L total samples. In this process each configuration is not guaranteed an even number of samples, but we found that this reduces the variance of the estimated joint probability (and thus the estimated conditional probability) compared to an even split across $O(|S|^2)$ different sample sets.

Individual Runs In Fig. S3, we show two of the divergence curves for the Fast KL-Search method. The behavior of our method is somewhat different from standard sampling algorithms as the final step of the method is an optimization problem. As a result, there are certain points in the graph where the optimization problem switches between different local maxima. This becomes less frequent as more samples are obtained.

S5 Extended Comparison

For a lengthier comparison, we test Abstraction Sampling vs KL-search on Rock-Paper-Scissors for approximately 2.5 and 5 minutes in Fig. S4. Each test is performed over an average of 30 independent runs. We also provide a

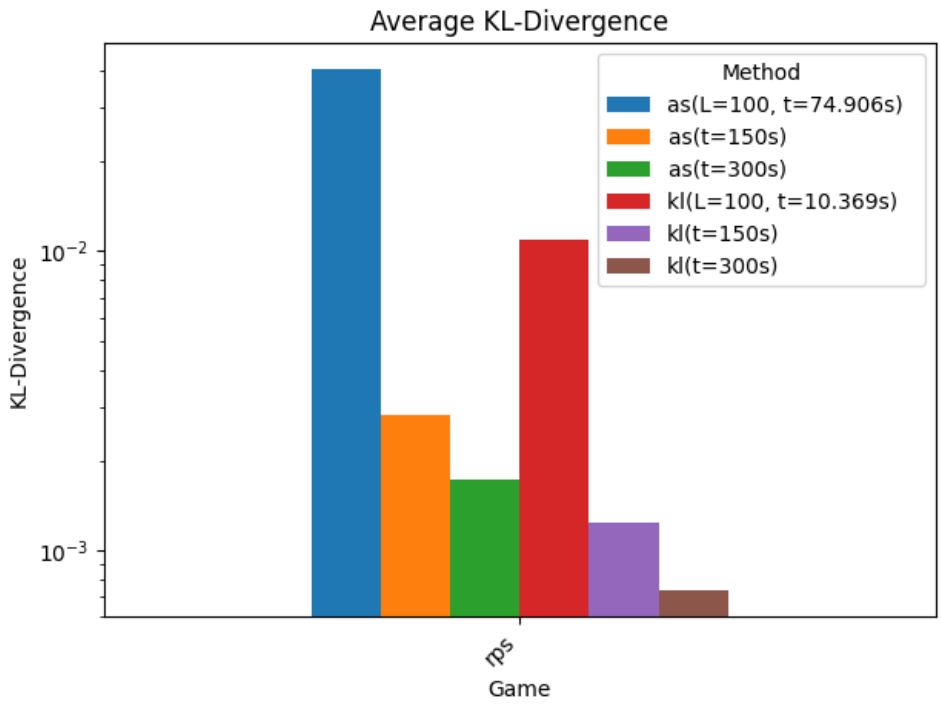

Figure S4: KL divergence for KL-search (left 3 bars) vs Abstraction Sampling (right 3 bars) for longer sample time

comparison to the 100 sample results from the main paper. The time taken for KL-search is not exactly 2.5/5 minutes due to the stopping criteria used. Since we need to solve an optimization problem and an inference problem for $P(X_{t+1}|Nb(X)_{t+1})$ after sampling, we add an early stopping criteria of 10s before 2.5/5 minutes for post-sampling computation. The KL-divergence continues to converge towards 0 for both methods even beyond the initial 100-200 samples.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [No]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [No]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]