Master in Intelligent Interactive Systems

Universitat Pompeu Fabra

Optimal Control Using Sparse-Matrix Belief Propagation

Albert Iribarne

Supervisor: Vicenç Gómez Cerdà

July 2019

Universitat Pompeu Fabra Barcelona

Contents

1 Introduction **1**

1.1 Objectives 4

1.2 Structure of the Report 4

2 Methods **6**

2.1 Kullback-Leibler Optimal Control 6

2.1.1 Kullback-Leibler Divergence 6

2.1.2 Kullback-Leibler Minimisation 7

2.1.3 Optimal Control 7

2.1.4 Optimal Control as a Kullback-Leibler Minimisation 8

2.2 Multi-Agent Cooperative Game 10

2.2.1 Graphical model for the KL stag hunt game 12

2.2.2 Conversion to a Pairwise Markov Random Field 15

2.2.3 Ground truth model 16

2.3 Sparse Matrix Belief Propagation 17

2.3.1 Belief Propagation 17

2.3.2 Exploiting Graph Sparseness for Belief Propagation 19

2.3.3 Stag Hunt Sparse Matrix Representation 20

3 Results **22**

3.1 Implementation Specifics 22

3.1.1 The Hunters Trajectories 23

3.1.2 Tests for Correctness 24

3.2 Empirical Evaluation 25

3.2.1 Risk vs Payoff Dominant Strategies 25

3.2.2 Running Time Analysis 26

4 Discussion **31**

4.1 Discussion 31

4.2 Conclusions 33

List of Figures **34**

Bibliography **37**

Abstract

The optimal control framework is a mathematical formulation by means of which many decision making problems can be represented and solved by finding optimal policies or controls. We consider the class of optimal control problems that can be formulated as a probabilistic inference on a graphical model, known as Kullback-Leibler (KL) control problems. In particular, we look at the recent progress on exploiting parallelisation facilitated by the graphics processing units (GPU) to solve such inference tasks, considering the recently introduced sparse-matrix belief propagation framework [1]. The sparse-matrix belief propagation algorithm was reported to deliver significant improvements in performance with respect to traditional loopy belief propagation, when tested on grid Markov random fields.

We develop our approach in the context of the KL-*stag hunt* game, a multi-agent, grid-like game which shows two different behavior regimes [2]. We first describe how to transform the original problem into a pairwise Markov random field, amenable to inference using sparse-matrix belief propagation and, second, we perform an experimental evaluation. Our results show that the use of GPUs can bring notable performance improvements to the optimal control computations in the class of KL control problems. However, our results also suggest that the improvements of sparse-matrix belief propagation may be limited by the concrete form of the Markov random field factors, specially on models with high sparsity *within* a factor, and variables with high cardinality.

Keywords optimal control · graphical model · approximate inference · sparse matrix · belief propagation · GPU

Chapter 1

Introduction

Many decision making problems can be represented as a control problem that can be solved by finding an optimal policy (a mapping from states to actions) using dynamic programming. Dynamic programming is based on estimating some measures of the value of state-action, also known as action-value function, through the Bellman equation [3]. However, for large-scale high-dimensional problems characterised by sparse and delayed costs, computing the value function by dynamic programming is intractable [4]. This is the case, for example, of many applications in robotics, operations research and process control.

In these cases, the standard approach is to approximate the value function and/or the optimal policy using function-approximation and Monte-Carlo sampling [5, 6]. The general algorithm is to interleave a *sampling step*, which executes trajectories on the system, with a *policy update step*, that uses gradient information on the policy and/or value function parameters. This approach has lead to most of the recent breakthroughs in reinforcement learning and optimal control, partly because it can be applied in a model-free setting enabling the possibility of end-to-end policy learning through deep learning techniques [7, 8, 9, 10, 11, 12]. However, current model-free, deep reinforcement learning methods are faced with the impossibility of performing efficient exploration of the state-action space, and require enormous amounts of data to learn controllers that reach human-level performance.

In this thesis, we explore an alternative approach: the probabilistic graphical model inference approach [13, 14]. Here, we start from a given, possibly simplified, model of the environment and actions that is represented using a probabilistic graphical model, which defines a joint probability distribution over the state and the environment variables. Such a representation allows the direct use of approximate inference algorithms that exploit the structure of the graph in some way or another, to compute the optimal control. Of particular interest, because of its wide applicability, is the belief propagation algorithm [13, 15, 16]. Belief propagation performs message passing between the nodes in the graphical model until it reaches a stable condition. After convergence, one can easily obtain estimated posterior marginals of the variables of interest. While belief propagation can fail to converge in general graphs, in practice it does converge in many problems, and very often the estimated marginals are highly accurate [17, 18].

We use the framework of optimal control as probabilistic graphical model inference [2, 19, 20, 21], for which finding an optimal policy corresponds to computing the posterior probability over trajectories that follow the optimally controlled dynamics. We consider the formulation introduced in [2], which known as Kullback-Leibler control and directly expresses our task as a probabilistic inference on a graph.

Kullback-Leibler control problems, also known as linearly-solvable Markov decision processes [22], enjoy several computational advantages, such as a linear Bellman equation [23, 24], compositionality of optimal control laws [25, 26, 27], or fast rates in the online learning setting [28]. The continuous-time version of these problems is known as Path Integral control [24], and has exclusively considered the sampling-based approach, e.g., in robotics [29, 30], multi-agent systems [29, 31, 32]. However, applicability of this class of control problems to real-world ones is challenging, mainly because they impose strong restrictions on the system's controllability [22, 33] or due to the curse of dimensionality [34, 35].

One of the main limitations of using probabilistic graphical models for optimal control is that the approach requires to *unfold* (ground) the entire sequence of state variables before performing the inference. Although this grounding benefits from explicit conditional independencies in the graphical structure, it has important limitations for large-scale problems, or for problems with dense costs. For these problems, the tree-width of the resulting graph, a measure the complexity of the inference task, is too high.

On the other hand, if a model is available and one can efficiently solve the associated probabilistic graphical model, there is no need for efficient exploration, thus circumventing the main workhorse of current sample-based methods.

In this thesis, we claim that many interesting problems can still be formulated using Kullback-Leibler control on a probabilistic graphical model. Motivated on how novel hardware architectures have enabled the widespread use of deep learning, we consider how recent advances in accelerating message passing algorithms for probabilistic inference can be used to scale up Kullback-Leibler control problems. In particular, we look at the recent progress on exploiting the parallelisation facilitated by the graphics processing units (GPUs) to solve the inference tasks on a graphical model. In this sense, we consider the recently introduced sparse matrix belief propagation framework [1].

We develop our methodology for a particular multi-agent game. Games are often used as a model for different decision-making situations, such as the *prisoner's dilemma* [36, 37]. In our case, however, we focus on the *stag hunt* [38]. The *stag hunt* is a cooperative game in which a set of hunters can choose between hunting a hare or hunting a stag. Hunters can catch a hare by themselves and receive a small reward R_h, but it requires at least the coordination of two hunters to hunt a stag, which provides them with a much larger reward R_s. Hence, if only one hunter decides to chase the stag, it will not be able to get it, thus receiving a null reward.

The nature of the problem presents a conflict between personal risk and mutual benefit, provided that the successful coordination to hunt a stag awards great benefits for the implicated players, yet a synchronisation failure can lead to null profit for all of them. This allows for the study of cooperation within social structures [39] and collaborative behavoiour in multi-agent systems [40], and can also be used to understand cooperativity directly in humans [41, 42].

We follow the version of the game introduced in [2], which is played in a grid with a fixed position for the preys: stags and hares, and an initial position for the hunters, who can freely roam around it for a fixed amount of time T. Performing exact inference can be too costly for complex instances and approximate inference methods are needed to obtain approximate optimal controls [43, 44, 45]. We make explicit use of the graphical model structure, and take advantage of certain conditional independences in the agents dynamics, namely, that the agent dynamics decouples in the absence of control, and that the reward (payoff) function can be factorised exactly as a product of smaller functions, involving a very small group of variables.

1.1 Objectives

The objectives of the present master thesis are the following:

- To integrate the recent results on GPU acceleration for message passing in optimal control tasks.
- To formulate the stag hunt probabilistic graphical model as a pairwise Markov random field so that it can be express under the sparse-matrix belief propagation framework.
- To recover by means of such framework the existing approximate inference results on the stag hunt model.
- To measure and compare the performance of different configurations of belief propagation on the stag hunt model, both in CPU and GPU hardware.

1.2 Structure of the Report

In Chapter 2 we review how a class of stochastic optimal control problems can be reformulated as Kullback-Leibler minimisation problems, reducing the optimal control computation to an inference computation, and enabling the use of approximate inference methods to efficiently compute approximate optimal controls. We then introduce the *stag hunt* and its graphical model structure, and provide the methods that allow its formulation as a pairwise Markov random field. Lastly, we present the sparse-matrix belief propagation algorithm and describe how the stag hunt can be expressed under such framework, allowing us to efficiently compute the approximate optimal controls for the game.

In the first section of Chapter 3 we give the details of our implementation of the stag hunt under the sparse-matrix belief propagation framework, and of the variety of tests we designed in order to assess such implementation. Then the second and last section of this chapter contains a description of the the obtained experimental results.

We conclude this thesis in Chapter 4, where we discuss our results and outline future lines of research.

Chapter 2

Methods

2.1 Kullback-Leibler Optimal Control

In this section we review how a general class of optimal control problems can be reformulated as a Kullback-Leibler minimisation problem [2], allowing the use of approximate inference methods such as Belief Propagation to solve the optimal control computation. We introduce the generalised stag hunt game model as it was presented also in [2], and transform it into a pairwise Markov Random Field, in order to be able to exploit the potential of the sparse-matrix modification of the belief propagation algorithm [1] to perform the approximate inference.

2.1.1 Kullback-Leibler Divergence

The Kullback-Leibler divergence, also known as the relative entropy between two probability distributions $p(x)$ and $q(x)$, that are defined over the same state space, is defined as:

$$D_{\mathrm{KL}}(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \ . \tag{2.1}$$

It is a measure of how a probability distribution p is different from a second reference probability distribution q, satisfying $D_{\mathrm{KL}}(p \parallel q) \geq 0$, which is known as the *Gibbs' inequality*, and being exactly equal to 0 only if $p = q$. Note also that its expression can be written as the expectation of the logarithmic difference between p and q, with respect to p:

$$D_{\text{KL}}(p \parallel q) = \mathbb{E}_{x \sim p} \left[\log \frac{p(x)}{q(x)} \right] \tag{2.2}$$

2.1.2 Kullback-Leibler Minimisation

Consider the Kullback-Leibler divergence between a probability distribution $p(x)$ and a positive function $\psi(x)$, as a function of the probability distribution p:

$$C(p) = \sum_x p(x) \log \frac{p(x)}{\psi(x)} \tag{2.3}$$

The distribution that minimises C with respect to p subject to the normalisation $\sum_x p(x) = 1$, can be found by adding a Lagrange multiplier:

$$L(p) = C(p) + \beta \left(\sum_x p(x) - 1 \right), \quad \frac{\mathrm{d}L}{\mathrm{d}p} = \log \frac{p(x)}{\psi(x)} + 1 + \beta \tag{2.4}$$

And setting the derivative $\frac{\mathrm{d}L}{\mathrm{d}p}$ equal to zero, giving $p(x) = \psi(x)\exp(-(1+\beta))$. If we define $Z = \exp(1+\beta)$ we have $p(x) = \psi(x)/Z$, and by means of the normalisation condition, $Z = \sum_x \psi(x)$. Finally, substituting the solution that minimises the Kullback-Leibler divergence in $C(p)$, one obtains $C = -\log Z$.

2.1.3 Optimal Control

Let $t = 0, 1, \ldots, T$, $T \geq 0$, denote time, and consider a Markov chain $(x_t)_t$ on a finite state space $\mathcal{S} = \{1, 2, \ldots, N\}$, $N \geq 1$, so that $x^t \in S, \forall t$. Denote by $p^t(x^{t+1} \mid x^t, u^t)$ the Markov transition probability at time t under control u^t, from state x^t to state x^{t+1}. Finally, let $p(x^{1:T} \mid x^0, u^{0:T-1})$ denote the probability of the trajectory $x^{1:T}$ given the initial state x^0 and the control trajectory $u^{0:T-1}$.

For any $t < T$, let us define as $\hat{R}(x^t, u^t, x^{t+1}, t)$ the cost associated to applying control -action- u^t to the system, changing its state from x^t to x^{t+1}. Let us further convene that at the horizon T, this cost becomes only state dependant: $\hat{R}(x^T, u^T, x^{T+1}, T) = R(x^T, T)$. The control problem is then to find the action sequence $u^{0:T-1}$ that minimises the expected future cost:

$$C(x^0, u^{0:T-1}) = \mathbb{E}_p \left[\sum_{t=0}^{T} \hat{R}(x^t, u^t, x^{t+1}, t) \right] \tag{2.5}$$

Or equivalently:

$$C(x^0, u^{0:T-1}) = \sum_{x^{1:T}} p(x^{1:T} \mid x^0, u^{0:T-1}) \sum_{t=0}^{T} \hat{R}(x^t, u^t, x^{t+1}, t) \tag{2.6}$$

2.1.4 Optimal Control as a Kullback-Leibler Minimisation

Consider the restricted class of control problems for which C in Equation (2.6) can be written as a Kullback-Leibler divergence.

Let us assume, in one hand, the existence of some uncontrolled transition probability matrix $q^t(x^{t+1} \mid x^t)$ over the same state space $\mathcal{S}$, which can be any first order Markov process that assigns zero probability to physically impossible state transitions. On the other hand, assume that the cost $\hat{R}$ can be expressed as the sum of a control dependent term and a state dependent term.

Since control u acts directly on the transition probabilities p, one can express the control dependent cost in terms of the logarithmic difference between the controlled and uncontrolled transition probabilities. For any $t = 0, 1 \ldots, T-1$:

$$\hat{R}(x^t, u^t, x^{t+1}, t) = \log \frac{p^t(x^{t+1} \mid x^t, u^t)}{q^t(x^{t+1} \mid x^t)} + R(x^t, t) \tag{2.7}$$

With R an arbitrary state dependent cost. Intuitively, the uncontrolled dynamics q would represent the *rules of nature*, and the first term on the right hand of the equation would encode the cost of deviating from those rules.

Now, one can conveniently define a function ψ as:

$$\psi(x^{1:T} \mid x^0) = q(x^{1:T} \mid x^0) \exp \left(- \sum_{t=0}^{T} R(x^t, t) \right) \tag{2.8}$$

That allows us to express the action cost $\hat{R}$ defined in (2.7) as:

$$\begin{aligned}\sum_{t=0}^{T} \hat{R}(x^t, u^t, x^{t+1}, t) &= \log \frac{p(x^{1:T} \mid x^0, u^{0:T-1})}{q(x^{1:T} \mid x^0) \exp\{-\sum_{t=0}^{t} R(x^t, t)\}} \\ &= \log \frac{p(x^{1:T} \mid x^0, u^{0:T-1})}{\psi(x^{1:T} \mid x^0)}\end{aligned} \tag{2.9}$$

By means of which, taking into account the definition in (2.5), we can write the expected future cost as a KL-divergence:

$$C(x^0, u^{0:T-1}) = \mathbb{E}_p \left[\log \frac{p(x^{1:T} \mid x^0, u^{0:T-1})}{\psi(x^{1:T} \mid x^0)} \right] = D_{\mathrm{KL}}(p \parallel \psi) \tag{2.10}$$

Where in the second equality we have used the expression of the Kullback-Leibler divergence given in (2.2).

Since the expected future cost C depends on the control only through p one can write $\frac{\mathrm{d}C}{\mathrm{d}u} = \frac{\mathrm{d}C}{\mathrm{d}p}\frac{\mathrm{d}u}{\mathrm{d}p}$, and therefore a sufficient condition to find the optimal control is to set $\frac{\mathrm{d}C}{\mathrm{d}p} = 0$. Then, as seen in Section 2.1.2, this minimisation yields:

$$p(x^{1:T} \mid x^0) = \frac{1}{Z(x^0)} \psi(x^{1:T} \mid x^0) \tag{2.11}$$

With the following optimal cost:

$$C(x^0, p) = -\log Z(x^0) = -\log \sum_{x^{1:T}} q(x^{1:T} \mid x^0) \exp\left(-\sum_{t=0}^{T} R(x^t, t) \right) \tag{2.12}$$

The optimal control in the current state x^0 at the current time $t = 0$ is given by the marginal probability:

$$p(x^1 \mid x^0) = \sum_{x^{2:T}} p(x^{1:T} \mid x^0) \tag{2.13}$$

Which defines a standard graphical model inference problem, where the distribution p is given by (2.11). As illustrated in [2] this problem can be, under general assumptions, solved by many exact or approximate inference methods such as the Junction Tree method or Belief Propagation, respectively.

2.2 Multi-Agent Cooperative Game

Consider the following generalisation of the stag hunt game: there are M hunters in a grid of N locations, and each of them can choose between hunting a hare or hunting a stag, without knowing in advance the choice of the others. Hares can be hunted individually, giving the hunter a *small* reward R_h. On the other hand, catching a stag yields a much larger reward R_s, but it requires at least two agents to cooperate in the hunt.

Let $x_i^t \in \{1, \dots, N\}$, $i = 1, \dots, M$, $t = 1, \dots, T$ be the position of agent i at time t in a grid of N locations. Also, let $s_j \in \{1, \dots, N\}$, $j = 1, \dots, S$, and $h_k \in \{1, \dots, N\}$, $k = 1, \dots, H$, be the positions of the jth stag and the kth hare respectively. Consider the following state-dependent reward:

$$R(x^t) = R_h \sum_{k=1}^{H} \sum_{i=1}^{M} \delta_{x_i^t h_k} + R_s \sum_{j=1}^{S} \mathcal{I} \left\{ \sum_{i=1}^{M} \delta_{x_i^t s_j} > 1 \right\} \tag{2.14}$$

Where $\mathcal{I}$ denotes the function that takes the value 1 when the condition inside the brackets is satisfied and 0 otherwise, and δ_{ij} denotes the Kronecker delta over the indexes i and j. The first term on the right hand of the equation captures the number of agents located at a position of a hare, while the second one accounts for the rewards obtained in hunting stags. Note that, while the reward for a hare is proportional to the number of agents at its position, the reward corresponding to a stag remains the same, independent of the number of agents at its position, provided that it is greater or equal than two.

Assume that the game takes place in a square grid of $N = n^2$ locations, so that one can write the positions in cartesian coordinates. Let $x_i^t = (l, m)$, $x_i^{t+1} = (l', m')$, $l, m, l', m' \in \{1, \dots, n\}$ be the positions of agent i at times t and $t + 1$ respectively,

$i \in \{1, \ldots, M\}$, $t \in \{0, \ldots, T-1\}$, and consider the following function:

$$\begin{aligned}\psi_q\left(x_i^{t+1}, x_i^t\right) := \mathcal{I}\{ \; & ((l' = l) \wedge (m' = m)) \vee \\ & ((l' = l-1) \wedge (m' = m) \wedge (l > 0)) \vee \\ & ((l' = l) \wedge (m' = m-1) \wedge (l < n)) \vee \\ & ((l' = l) \wedge (m' = m+1) \wedge (m < n)) \; \}\end{aligned} \tag{2.15}$$

Which evaluates to one if either the agent does not move, or it moves one position left, right, up or down inside the grid boundaries. This allows us to define the uncontrolled dynamics of the system as conditional probabilities in the following manner:

$$q(x_i^{t+1} \mid x_i^t) = \frac{\psi_q(x_i^{t+1}, x_i^t)}{\sum_{x_i^{t+1}} \psi_q(x_i^{t+1}, x_i^t)} \; . \tag{2.16}$$

This is, without control, the probability is distributed uniformly among the possible actions the agent can take. Note that the uncontrolled dynamics factorises among the agents, and the joint probability distribution can be written as:

$$q(x^{t+1} \mid x^t) = \prod_{i=1}^{M} q(x_i^{t+1} \mid x_i^t) \; . \tag{2.17}$$

On the other hand, since we are interested in the final configuration at the time horizon, x^T, let us set the state dependent cost to zero for $t = 1, \ldots, T-1$ and to $\psi_R(x)$ for $t = T$, where ψ_R is defined as:

$$\psi_R(x^T) = \exp\left(-\frac{1}{\lambda} R(x^T)\right) \; . \tag{2.18}$$

So that we recover the control problem as it was defined in Section 2.1. Finally, note that in this case we can decompose the factor in terms of the kind of pray that is giving the reward:

$$\begin{aligned}\psi_R(x^T) &= \psi_{R_H}(x^T)\psi_{R_S}(x^T) \\ &= \exp\left(R_h \sum_{k=1}^{H} \sum_{i=1}^{M} \delta_{x_i^T h_k}\right) \exp\left(R_s \sum_{j=1}^{S} \mathcal{I}\left\{\sum_{i=1}^{M} \delta_{x_i^t s_j} > 1\right\}\right)\end{aligned} \tag{2.19}$$

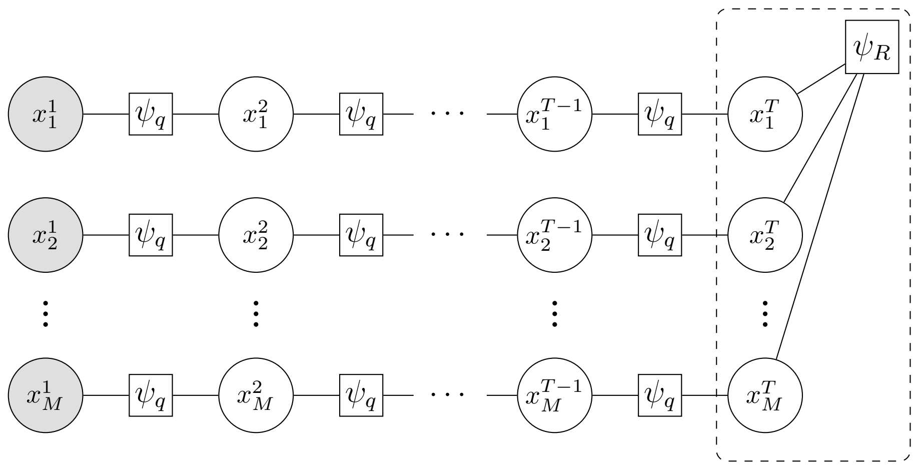


Figure 1: Factor graph representation of the generalised stag hunt game. Circles denote variable nodes, and squares denote factor nodes. Variables nodes in grey correspond to the *clamped* initial configuration.

2.2.1 Graphical model for the KL stag hunt game

Consider the graphical model that arises from factors ψ_q and ψ_R described in equations (2.15) and (2.18) respectively. In Figure 1 one can see how the uncontrolled dynamics ψ_q factorises over the agents and can be drawn as pairwise factors, while the factor node associated to the state cost ψ_R involves all agent state nodes at the time horizon. Despite being a tree, exact and even approximate inference approaches become intractable in this model, due to the complex factor ψ_R.

Let us then consider the transformation of the problem proposed in [2], which decomposes the structure of ψ_R in smaller factors defined over at most three auxiliary variables of small cardinality:

1. M factors $\psi_h(x_i^T)$ defined for each agent location at the time horizon $x_i^T \in \{1, \ldots, N\}$, $i = 1, \ldots, M$, which account for the hare costs:

$$\psi_h(x_i^T) := \begin{cases} \exp\left(-\frac{1}{\lambda} R_h\right) & \text{if } \sum_{k=1}^{H} \delta_{x_i^T h_k} > 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.20}$$

2. $S \times M$ factors $\psi_{s_j}(x_i^T, d_{ij})$ defined for each stag location $s_j \in \{1, \ldots, N\}$,

$j = 1, \dots, S$, and each agent location at the time horizon x_i^T. New auxiliary variables $d_{ij} \in \{0, 1\}$, defined for each stag and each agent positions, encode whether agent i and stag j are in the same position or not, and the factor is defined as:

$$\psi_{s_j}(x_i^T, d_{ij}) = \mathcal{I}\left\{ d_{ij} = \delta_{x_i^T s_j} \right\} \ . \tag{2.21}$$

3. $S \times M$ factors ψ_{r_i} involving the previously defined binary variables d_{ij} and additional auxiliary variables $u_{ij} \in \{0, 1, 2\}$, $i = 2, \dots, M$, $j = 1, \dots, S$, which encode whether the stag j has zero, one or more agents at its position after considering the ith agent. These factors are defined as:

$$\begin{aligned}
\psi_{r_1}(d_{1j}, d_{2j}, u_{1j}) :=& \mathcal{I}\{\ ((d_{1j} = 0) \wedge (d_{2j} = 0) \wedge (u_{1j} = 0)) \vee \\
& ((d_{1j} = 1) \wedge (d_{2j} = 1) \wedge (u_{1j} = 2)) \vee \\
& ((d_{1j} \neq d_{2j}) \wedge (u_{1j} = 1))\ \} \\
\psi_{r_{i-1}}(u_{(i-1)j}, d_{ij}, u_{ij}) :=& \mathcal{I}\{\ \big((d_{ij} = 0) \wedge (u_{(i-1)j} = u_{ij})\big) \vee \\
& \big((d_{ij} = 1) \wedge (u_{(i-1)j} = 0) \wedge (u_{ij} = 1)\big) \vee \\
& \big((d_{ij} = 1) \wedge (u_{(i-1)j} = 1) \wedge (u_{ij} = 2)\big) \vee \\
& \big((d_{ij} = 1) \wedge (u_{(i-1)j} = 2) \wedge (u_{ij} = 2)\big)\ \} \\
\psi_{r_M}(u_{(M-1)j}) :=& \begin{cases} \exp\left(-\frac{1}{\lambda} R_s\right) & \text{if } u_{(M-1)j} = 2, \\ 1 & \text{otherwise} \end{cases}
\end{aligned} \tag{2.22}$$

Where i ranges from $i = 3$ to $i = M$ in the second definition. Note that both definitions of ψ_{r_1} and $\psi_{r_{i-1}}$ encode the allowed configurations of d_{ij} and u_{ij}, while ψ_{r_M} weights the allowed configurations of having zero, one or more agents being at each stag position according to the corresponding cost.

Note that every pairwise factor introduced in this section, as well as the uncontrolled dynamics factor q, can take only two values, 0 and 1.

By means of this procedure one obtains a computationally tractable representation of the problem, where the complex factor ψ_R can be rewritten in terms of smaller factors that involve, at most, three variables of the model. On the one hand, ψ_{R_H}

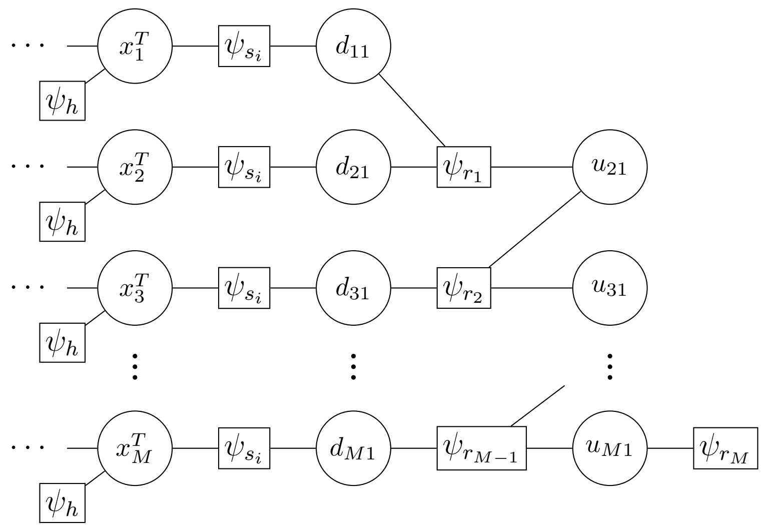


Figure 2: Decomposition of the complex factor ψ_R into simpler factors involving at most three variables of small cardinality. Note that in the figure it is only shown the structure of the factor for stag i, which would be identically reproduced for the rest of the stags.

in (2.19) factorises trivially among the agents:

$$\psi_H(x^T) = \prod_{i=1}^{M} \psi_h(x_i^T) \tag{2.23}$$

On the other hand, ψ_{R_S} can be rewritten marginalising the auxiliary variables d_{ij}, u_{ij} over the product of the factors ψ_{s_j} and ψ_{r_i}:

$$\psi_S(x^T) = \prod_{j=1}^{S} \left[\sum_{\substack{d_{1j},d_{2j}\\u_{2j},u_{Mj}}} \psi_{s_j}(x_1^T, d_{1j})\psi_{s_j}(x_2^T, d_{2j})\psi_{r_1}(d_{1j}, d_{2j}, u_{2j})\psi_{r_M}(u_{Mj}) \right. \\ \left. \sum_{\substack{d_{3j},\ldots,d_{Mj}\\u_{2j},\ldots,u_{Mj}}} \prod_{i=3}^{M} \psi_{r_{i-1}}(u_{(i-1)j}, d_{ij}, u_{ij})\psi_{sj}(x_i^T, d_{ij}) \right] \tag{2.24}$$

Obtaining the factor decomposition:

$$\psi_R(x^T) = \exp\left(-\frac{1}{\lambda}R(x^T)\right) = \psi_H(x^T)\psi_S(x^T) \tag{2.25}$$

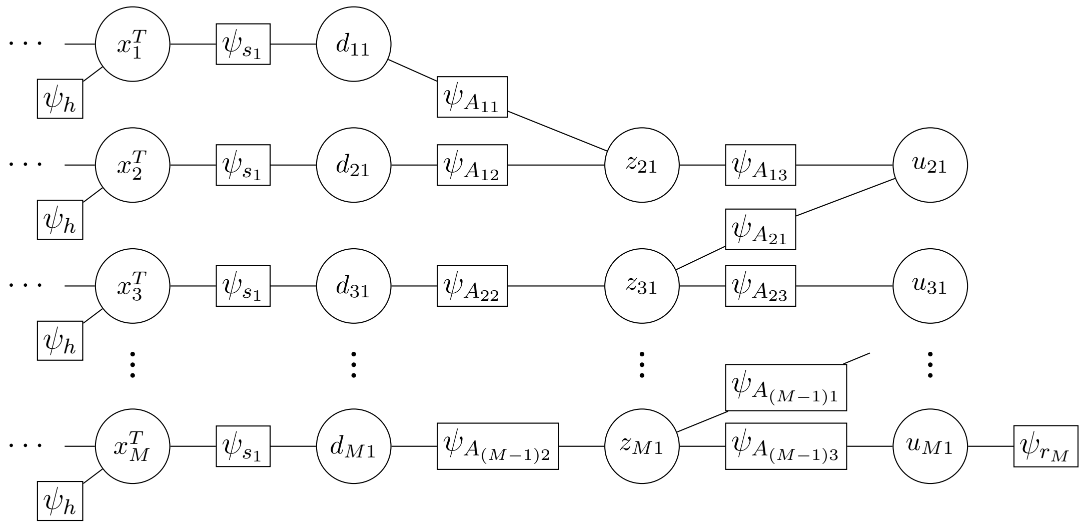


Figure 3: Pairwise factor graph representation of the generalised stag hunt game. The auxiliary variables z_{ij}, $i = 2, \ldots, M$, $j = 1, \ldots, S$ allow us to replace the ternary factors $\psi_{r_{i-1}}$ for the binary factors $\psi_{A_{(i-1)s}}$, $s = 1, 2, 3$.

2.2.2 Conversion to a Pairwise Markov Random Field

Consider the graphical model obtained in Section 2.2.1, composed of factors ψ_q, ψ_h, ψ_{s_j}, $j = 1, \ldots, S$ and ψ_{r_i}, $i = 1, \ldots, M$, among which only ψ_{r_i}, for $i = 1, \ldots, M-1$, involve three variables, while the rest are pairwise or unary factors. Then, if one converts these factors into a pairwise form, the model becomes a Pairwise Markov Random Field, which is in turn the simplest graph that can encode the problem structure.

Let us introduce, following the method proposed in [17], another set of auxiliary variables z_{ij}, $i = 1, \ldots, M-1$, $j = 1, \ldots, S$. On the one hand, for $j = 1, \ldots, S$, $z_{2j} \in \{0,1\} \times \{0,1\} \times \{0,1,2\}$ will be connected to d_{1j}, d_{2j} and u_{2j} with pairwise factors that will substitute ψ_{r_1}. On the other hand, for $i = 2, \ldots, M-1$ and $j = 1, \ldots, M$, $z_{ij} \in \{0,1,2\} \times \{0,1\} \times \{0,1,2\}$ will be connected to u_{ij}, $d_{(i+1)j}$ and $u_{(i+1)j}$ substituting the factors ψ_{r_i}.

1. Let $\mathcal{Z} = \{0,1\} \times \{0,1\} \times \{0,1,2\}$ be the state space of $z_{2j} = (z_{2j}^1, z_{2j}^2, z_{2j}^3) \in \mathcal{Z}$, where we have explicitly written the components of the new variable. Then,

we define the following potentials between d_{1j}, d_{2j} and u_{2j}, and z_{2j}:

$$\begin{aligned}\psi_{A_{11}} &:= \psi_{r_1}(z^1_{2j}, z^2_{2j}, z^3_{2j})\mathcal{I}\left\{z^1_{2j} = d_{1j}\right\}\\ \psi_{A_{12}} &:= \psi_{r_1}(z^1_{2j}, z^2_{2j}, z^3_{2j})\mathcal{I}\left\{z^2_{2j} = d_{2j}\right\}\\ \psi_{A_{13}} &:= \psi_{r_1}(z^1_{2j}, z^2_{2j}, z^3_{2j})\mathcal{I}\left\{z^3_{2j} = u_{2j}\right\}\end{aligned} \tag{2.26}$$

2. Analogously, let $\mathcal{Z} = \{0,1,2\} \times \{0,1\} \times \{0,1,2\}$ be the state space of $z_{ij} = (z^1_{ij}, z^2_{ij}, z^3_{ij}) \in \mathcal{Z}$, $i = 3, \ldots, M$. Now, we define the potentials between $u_{(i-1)j}$, d_{ij} and u_{ij}, and z_{ij} as:

$$\begin{aligned}\psi_{A_{(i-1)1}} &:= \psi_{r_{i-1}}(z^1_{ij}, z^2_{ij}, z^3_{ij})\mathcal{I}\left\{z^1_{ij} = u_{(i-1)j}\right\}\\ \psi_{A_{(i-1)2}} &:= \psi_{r_{i-1}}(z^1_{ij}, z^2_{ij}, z^3_{ij})\mathcal{I}\left\{z^2_{ij} = d_{ij}\right\}\\ \psi_{A_{(i-1)3}} &:= \psi_{r_{i-1}}(z^1_{ij}, z^2_{ij}, z^3_{ij})\mathcal{I}\left\{z^3_{ij} = u_{ij}\right\}\end{aligned} \tag{2.27}$$

We have then substituted the $S \times (M-1)$ ψ_{r_i} ternary factors, $i = 1, \ldots, M-1$, for $3 \times S \times (M-1)$ pairwise factors, and converted the graphical model into a Pairwise Markov Random field, as illustrated in Figure. This will allow us to take advantage of the sparse-matrix belief propagation algorithm to solve the inference problem on this model.

2.2.3 Ground truth model

Finally, let us recover the graphical structure illustrated in Figure 1. When the number of agents is $M = 2$ and there is only $S = 1$ stag, the model can be understood as a pairwise Markov Random Field without further modifications. As illustrated in Figure 4, every factor in the model involves at most two variables: ψ_q and ψ_h as defined in (2.15) and (2.20) respectively, and ψ'_R defined as:

$$\psi'_R(x^T_1, x^T_2) := \begin{cases} \exp\left(-\frac{1}{\lambda}R_s\right) & \text{if } x^T_1 = x^T_2 = s_1 \\ 0 & \text{otherwise}\end{cases} \tag{2.28}$$

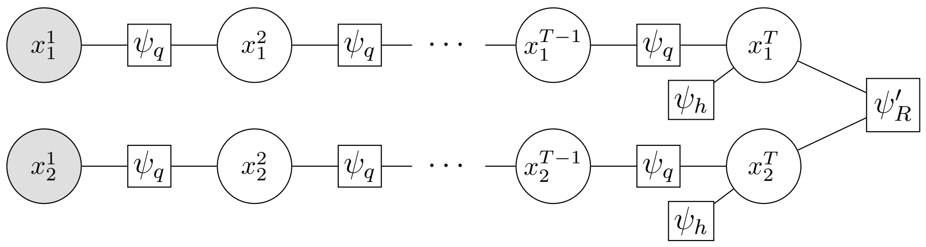


Figure 4: Graphical structure of the stag-hunt game with $M = 2$ agents. This particular game structure can be understood as a pairwise Markov Random Field without further modifications, and can serve as a ground truth model to assess the correctness of the proposed decomposition.

This MRF structure will then be greatly useful to computationally assess the correctness of the factor decompositions proposed in Sections 2.2.1 and 2.2.2.

2.3 Sparse Matrix Belief Propagation

In this section we introduce the sparse-matrix belief propagation algorithm as it was proposed in [1], which has been proven to significantly improve the performance of loopy belief propagation on pairwise Markov random fields. After reviewing the key concepts on traditional belief propagation and its sparse-matrix modified form, we describe how the latest can be used to perform inference over the model introduced in Section 2.2.

2.3.1 Belief Propagation

In the context of computational inference on probabilistic graphical models, the sum-product algorithm, also known as belief propagation, is a message passing algorithm that provides exact inference on tree-shaped graphs. However, a slight modification of the algorithm, often referred as *loopy* belief propagation, can perform approximate inference in graphs with cycles. Furthermore, since in the original algorithm messages are computed from the product of previous ones, the algorithm is often moved to the logarithmic domain, where the product becomes a sum and the floating point operations can remain stable.

Consider the pairwise Markov random field induced by the following factorised probability distribution:

$$p(x) = \frac{1}{Z} \exp \left(\sum_{i \in V} \phi_i(x_i) + \sum_{(i,j) \in E} \phi_{ij}(x_i, x_j) \right) \tag{2.29}$$

Where $x = (x_1, \ldots, x_n) \in \mathbf{X}$ is the state vector of the model, Z is a normalisation constant such that $\sum_x p(x) = 1$, and $\{V, E\}$ is the graph representation of the Markov random field.

With the described configuration, the message sent from variable x_i to variable x_j is:

$$m_{i \to j}[x_j] = \log \sum_{x_i} \exp \left(\phi_{ij}(x_i, x_j) + \phi_i(x_i) + \sum_{k \in N_i - j} m_{k \to i}[x_i] - d_{ij} \right) \tag{2.30}$$

Where $N_i = \{k \mid (i,k) \in E\}$, d_{ij} is any constant value that is eventually cancelled out by normalisation, and the bracket notation $m_{i \to j}[x_j]$ represents the indexing by the state of x_j of the message vector. Then, the estimated unary marginals or beliefs, are $p(x_j) \approx e^{b_j[x_j]}$, where:

$$b_j[x_j] = \phi_j(x_j) + \sum_{i \in N_j} m_{i \to j}[x_j] - z_j \tag{2.31}$$

And z_j is the normalisation constant:

$$z_j = \log \sum_{x_j} \exp \left(\phi_j(x_j) + \sum_{i \in N_j} m_{i \to j}[x_j] \right) \tag{2.32}$$

Again, the bracket notation is used for indexing into the beliefs, since they are interpreted as vectors for discrete variables. Finally, note that since the expression for $b_i[x_i]$ and the argument of the exponential function in (2.30) differ only in the term $m_{j \to i}[x_i]$ plus constants, the constants d_{ij} in (2.30) can be omitted and the message update expression becomes:

$$m_{i \to j}[x_j] = \log \sum \exp \left(\phi_{ij}(x_i, x_j) + b_i[x_i] - m_{j \to i}[x_i] \right) \tag{2.33}$$

2.3.2 Exploiting Graph Sparseness for Belief Propagation

Let n_v be the number of variables of the Markov random field, and c the maximum number of states of any variable. For simplicity, and without loss of generality, one can assume that all the variables have the same cardinality c. Let $\{(s_1, t_1), \ldots, (s_{|E|}, t_{|E|}\}$ be an arbitrary but fixed order in the edge set E, in which the edges are defined redundantly, this is $(i, j) \in E$ if, and only if, $(j, i) \in E$.

Let $\mathbf{B}$ be a $c \times n$ belief matrix whose columns are the stacked belief vectors, $B_{ij} = b_j[x_j = i]$, and consider the $c \times n$ unary factors tensor $\mathbf{\Phi}$ such that the jth column represents the jth unary factor: $\Phi_{ij} = \phi_j(x_j = i)$. Finally, consider the three-dimensional $c \times c \times |E|$ pairwise factors tensor $\mathbf{\Gamma}$ such that the kth slice is a matrix representing the kth pairwise factor as a table: $\Gamma_{ijk} = \phi_{s_k t_k}(x_{s_k} = i, x_{t_k} = j)$.

Now, let us define a $c \times |E|$ message matrix $\mathbf{M}$ such that its ith column is the ith message, this is $M_{ij} = m_{s_j \to t_j}[x_i]$. Finally, let $\mathbf{T}$ be a $|E| \times n$ sparse matrix whose non-zero entries are $T_{ij} = 1$ when $t_i = j$, mapping the ordered messages to the variables that receive those messages.

With this setup one can define both the message and the belief update equations in terms of tensor operations. The beliefs are updated, as in (2.31), by adding the unary factor functions of each variable and the sum of the messages that *arrive* to those variables, and subtracting the normalisation constant:

$$\tilde{\mathbf{B}} \longleftarrow \mathbf{\Phi} + \mathbf{M}^T\mathbf{T}, \qquad \mathbf{B} \longleftarrow \tilde{\mathbf{B}} - \text{logsumexp}(\tilde{\mathbf{B}}) \tag{2.34}$$

On the other hand, the message update equation in (2.33) becomes:

$$\mathbf{M} \longleftarrow \text{logsumexp}\left(\mathbf{\Gamma} + \mathbf{B}[: \mathbf{s}] - \mathbf{M}[: \mathbf{r}]\right) \tag{2.35}$$

Where $\mathbf{s} = [s_1, \ldots, s_{|E|}]^T$ and $\mathbf{r} = [r_1, \ldots, r_{|E|}]^T$ is such that $E[i] = (i, j) \Rightarrow E[r_i] = (j, i)$. It is important to note that $\mathbf{\Gamma}$ is a $c \times c \times |E|$ tensor while $\mathbf{B}[: \mathbf{s}] - \mathbf{M}[: \mathbf{r}]$ is a $c \times |E|$ matrix. The operation is done by stacking together c copies of the latest as rows to form the same shape as $\mathbf{\Gamma}$.

2.3.3 Stag Hunt Sparse Matrix Representation

Sparse-matrix belief propagation encodes both unary and pairwise factors into tensors in the logarithmic domain. Then, the pairwise Markov random field model for the generalised stag hunt game obtained in section 2.2.2 needs to be slightly modified.

On the one hand, instead of using the factors described above, we have to consider the natural logarithm of those functions. On the other hand, the variables of our model have different cardinalities, and therefore both the unary and the pairwise factors, expressed as vectors and matrices respectively, will have different shapes depending on the involved variables. Finally, a 1-1 map between all the possible values of any variable x and the index set of the cardinality c_x of x, $\{0, \dots, c_x\}$, has to be established and fixed. From now on those maps are assumed, and so that all the variables involved in the model take values in the index set of their cardinality.

The unary factors matrix $\mathbf{\Phi}$ for the generalised stag hunt game is a $c \times n_v$ matrix, where:

$$\begin{aligned} c &= N \\ n_v &= MT + S\,(M + 2(M-1)) \end{aligned} \tag{2.36}$$

Let $i \in \{0, \dots, c-1\}$ and $j \in \{0, \dots, n_v - 1\}$. Let $\{x_j\}_j$ be an arbitrary but fixed order over the set of variables of the model. Note that during Section 2.2 we have only specified unary factors for some variables. Since in the sparse-matrix setup for belief propagation every variable should have a unary factor made explicit, we set for the variables that do not have a unary factor defined, the neutral factor $\psi = 1$, that does not change the factorised probability distribution. Then, the unary factors matrix is defined as:

$$\mathbf{\Phi}_{ij} = \begin{cases} \log \psi_j(x_j = i) & \text{if } i < c_j \\ -\infty & \text{otherwise} \end{cases} \tag{2.37}$$

Where $c_j = |\mathcal{X}_j|$ and $\mathcal{X}_j$ is the domain of ψ_j, and where we have solved in turn the difference of cardinalities within the variables of the model.

On the other hand, the pairwise factors matrix $\mathbf{\Gamma}$ is a $c \times |E| \times |E|$ tensor, where:

$$|E| = 2[M(T-1) + MS + 3(M-1)S] \tag{2.38}$$

And $\{(s_1, t_1), \ldots, (s_{|E|}, t_{|E|})\}$ is an arbitrary fixed order on the edge set E of the model. Then, $\mathbf{\Gamma}$ is defined as:

$$\Gamma_{ijk} = \begin{cases} \log \psi_{s_k t_k}(x_{s_k} = i, x_{t_k} = j) & \text{if } i < c_{s_k} \text{ and } j < c_{t_k} \\ -\infty & \text{otherwise} \end{cases} \tag{2.39}$$

Where $c_{s_k} = |\mathcal{X}_{s_k}|$, $c_{t_k} = |\mathcal{X}_{t_k}|$, and $\mathcal{X}_{s_k}$ and $\mathcal{X}_{t_k}$ are the domains of ψ_{s_k} and ψ_{t_k} respectively.

Chapter 3

Results

3.1 Implementation Specifics

We have implemented the stag hunt Markov random field model and its sparse-matrix representation in the `staghunt` Python module, based on the `mrftools` framework [46] developed by the authors of [1].

The interface of our module has three main classes. The StagHuntGame class is a representation of the game and its parameters. Then, the MatrixStagHuntModel class encodes the Markov random field model of the game as `mrftools` MarkovNet and BeliefPropagator -or MatrixBeliefPropagator- objects. This representation uses `numpy` and `scipy.sparse` objects for computation, and so its usage is limited to the CPU. Finally, the TorchStagHuntModel class is a *translation* of the mentioned MatrixStagHuntModel class into `pytorch`, using the `mrftools` objects TorchMarkovNet and TorchMatrixBeliefPropagator, in which the sparse-matrix belief propagation can be ran both in CPU or in GPU hardware.

Lastly, it is worth mentioning that, while both MatrixBeliefPropagator and TorchMatrixBeliefPropagator `mrftools` objects accept Python `-float('inf')` special value and can perform their operations normally, the BeliefPropagator object does not. Therefore, we introduced in the uses of such object, a `MIN` constant set to an extremely negative value, that takes the role of $-\infty$.

3.1.1 The Hunters Trajectories

It is important to note how the hunters trajectories are obtained from the results of belief propagation.

The first step, before executing the algorithm, is to *clamp* the variables $x_1^1, \dots, x_M^1$ according to the initial (known) positions of the agents. This is, for every $i = 1, \dots, M$, if $a_i \in \{0, \dots, N-1\}$ is the initial position of agent i and s_i is the index of x_i^1 in the fixed order of the variable set:

$$\Phi_{is_i} = \begin{cases} 0 & \text{if } i = a_i \\ -\infty & \text{otherwise} \end{cases} \tag{3.1}$$

Now, as we have seen in Sections 2.3.1 and 2.3.2, the output of belief propagation is a belief matrix $\mathbf{B}$ whose columns are the belief vectors, $B_{ij} = b_j[i]$, which represent (single variable) marginal estimates, i.e., $b_j[i] \approx p(x_j = i)$. Furthermore, one can also compute the pairwise beliefs for each pair of connected variables, which serve in turn as estimates for the joint probabilities $p(x_i, x_j)$ for i, j such that $(i, j) \in E$. Therefore, after a run of belief propagation, one can compute every conditional probability $p(x_i^{t+1} \mid x_i^t)$ for $t = 1, \dots, T-1$ and $i = 1, \dots, M$.

In this work, unlike in [2], we compute the agent trajectories by sequentially clamping the position of the agents. This is, at each time index $t = 1, \dots, T-1$, we run belief propagation on the MRF model and decide the next position of every agent $i = 1, \dots, M$ based on the maximum value of $p(x_i^{t+1} \mid x_i^t)$, breaking ties randomly. Once we have the agent position at time $t+1$, we clamp the variable x_i^{t+1} to this position as done in (3.1), and repeat until the end of the trajectory. Such a sequence of *interventions* in the graphical model ensures that generated trajectories are fully consistent, with the obvious drawback of requiring $T-1$ executions of belief propagation in order to solve the optimal control computation.

3.1.2 Tests for Correctness

In order to assess the correctness of our implementation we defined a wide variety of tests available within the `staghunt` module. All the tests start generating two identical instances of the StagHuntGame randomly.

Ground vs. Pairwise In one of them we build the ground truth Markov random field model described in Section 2.2.3, and in the other we build the pairwise model described in Section 2.2.2. The game is ran in both models as described in Section 3.1.1, and at each time step we assert that both the marginals and the pairwise beliefs coincide for every variable and every pair of connected variables respectively.

Loopy vs. Matrix In both instances we build the pairwise model described in Section 2.2.2. The game is ran in one of the models using the loopy belief propagation implementation of the `mrftools` library, and in the other using the sparse-matrix implementation. At each time step we assert that both the marginals and the pairwise beliefs coincide for every variable and every pair of connected variables respectively.

Matrix vs Torch The pairwise model described in Section 2.2.2 is built in one of the two instances using the MatrixStagHuntModel class, based on `numpy`, and using the TorchStagHuntModel class in the other one, based on `pytorch`. The game is ran in both models and at each time step we assert that both the marginals and the pairwise beliefs coincide for every variable and every pair of connected variables respectively.

CPU vs. GPU In both instances we build the pairwise model described in Section 2.2.2. The game is ran in both models using the sparse-matrix `pytorch` implementation of the `mrftools` library, in one of them using CPU hardware and in the other one using GPU hardware. At each time step we assert that both the marginals and the pairwise beliefs coincide for every variable and every pair of connected variables respectively.

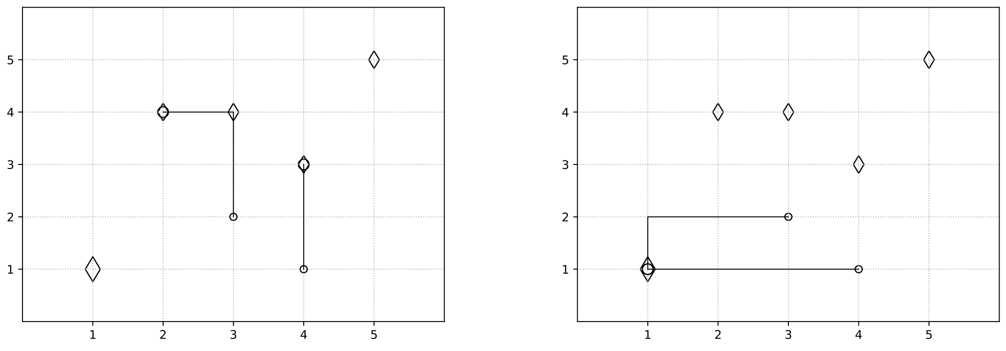
3.2 Empirical Evaluation

All the experiments described in this section share the following setup: for a fixed number M of agents, the number of stags and hares are set to $S = \lfloor M/2 \rfloor$ and $H = 2M$ respectively. Their locations, as well as the initial states x_i^1, $i = 1, \ldots, M$, are chosen randomly and non-overlapping, and the rewards for hunting a stag and for hunting a hare are $R_s = -10$ and $R_h = -2$ respectively.

3.2.1 Risk vs Payoff Dominant Strategies

Let us first test our implementation in a small instance of the game. Figure 5 shows both the initial and final positions of the agents as well as their trajectories, after running the stag hunt on a 5×5 grid with two agents, one stag and four hares. We observe how we recover the duality of optimal strategies, from payoff to risk dominant, when the parameter λ increases from $\lambda = 0.1$ to $\lambda = 10$, as in [2]. For high values of λ (Figure 5a), each hunter catches one of the hares. On the other hand, for small enough values of λ (Figure 5b), both hunters cooperate to catch the stag. Note that the game was ran using the ground truth Markov random field model described in Section 2.2.3, which is a tree, and therefore the belief propagation is exact.

Figure 6 shows two solutions observed on a larger instance of the game. In this case, we run sparse-matrix belief propagation on a GPU using our `pytorch` implementation of the pairwise Markov random field model described in Section 2.2.2. Note how, for $\lambda = 10$ all the agents hunt hares and no one cooperates, being the strategy risk dominant. Conversely, for $\lambda = 0.1$ almost every agent cooperates with others to hunt a stag, in a payoff dominant strategy. Note that there is one agent in the top-right corner hunting a hare, even that there is one stag in a reachable position. Reaching that stag position, though, would not increase the final game reward.



(a) $\lambda = 10$, risk dominant. (b) $\lambda = 0.1$, payoff dominant.

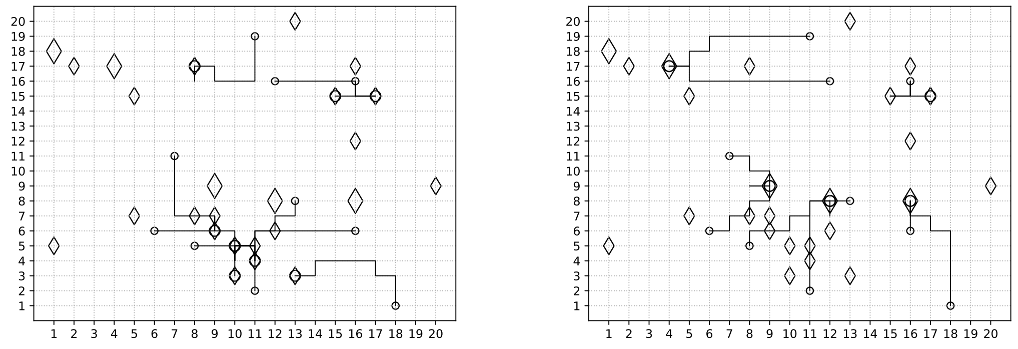
Figure 5: Exact inference on the stag hunt: optimal control obtained using loopy belief propagation on the tree-shaped ground truth MRF model of the game, for $M = 2$ hunters (circles) in a 5×5 grid with $S = 1$ stag (big diamond) and $H = 4$ hares (small diamonds), within a time horizon of $T = 4$. (**a**) For $\lambda = 10$, the optimal control is risk dominant and the hunters go for the hares. (**b**) For $\lambda = 0.1$ the optimal control is payoff dominant and the hunters cooperate to hunt the stag.

3.2.2 Running Time Analysis

Let us now analyse the cpu-time required to run the game for different game, algorithm and hardware configurations.

Figure 7 shows the cpu-time required for running belief propagation as a function of the number agents for the risk-dominant regime (left) and the payoff dominant regime (right). We observe that the `pytorch` implementation on GPU hardware provides a notable speedup with respect to the rest of the configurations, in agreement with [1]. Such a performance improvement becomes especially remarkable as we increase the model size, either by increasing N or by increasing M, which can be better perceived in Figure 8a.

However, in the same figure one can see that, despite sparse-matrix belief propagation is reported to be faster than loopy belief propagation by its authors, we find the opposite behaviour in our model running times. Indeed, both in Figures 7a and 8a we see how loopy belief propagation becomes eventually faster than both `numpy` and `pytorch` based sparse-matrix belief propagation. This is due to the `MIN` constant that is used on the loopy belief propagation implementation of the game



(a) $\lambda = 10$, risk dominant

(b) $\lambda = 0.1$, payoff dominant

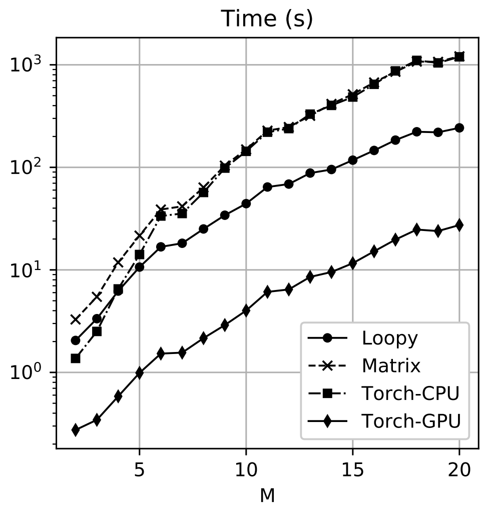
Figure 6: Approximate inference on the stag hunt: optimal control obtained using GPU sparse-matrix belief propagation on the pairwise MRF model of the game, for $M = 10$ hunters (circles) in a 20×20 grid with $S = \lfloor M/2 \rfloor$ stags (big diamonds) and $H = 2M$ hares (small diamonds), $R_s = -10$, $R_h = -2$, within a time horizon of $T = 10$. (**a**) For $\lambda = 10$, the optimal control is risk dominant and the hunters go for the hares. (**b**) For $\lambda = 0.1$ the optimal control is payoff dominant and the hunters cooperate to hunt the stag.

which, as mentioned in Section 3.1 takes the role of $-\infty$ in the belief propagation computation.

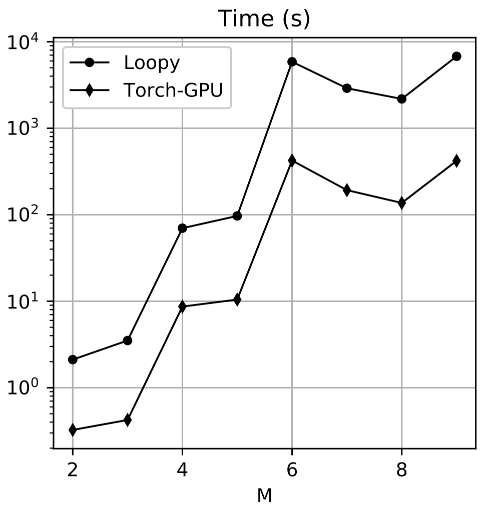
Then, the game configuration parameters that change the shape of the factors of the Markov random field obviously affect the running time of the game. As the factors grow in size so it does the running time, since more computations need to be performed in order to run both loopy and sparse-matrix belief propagation.

On the other hand, though, looking at Figures 7a, 7b and 8b, one can see how λ also affects the running time of the stag hunt. On Figure 7b the running time of the payoff dominant ($\lambda = 0.1$) game solved with GPU sparse-matrix belief propagation overcomes the 100 seconds for $M = 6$ agents, while it barely surpasses the 10 seconds for $M = 20$ agents in the risk dominant regime ($\lambda = 10$) in Figure 7a. Similarly, on Figure 8b it is clearly seen how different values of lambda result in large performance variations starting at a given value of M.

Given that the hardest computational task involved in the generation of optimal agent trajectories is belief propagation, it is obvious that both increasing the model size and decreasing the parameter λ affect the running time of this algorithm. Such



(a) $\lambda = 10$, risk dominant

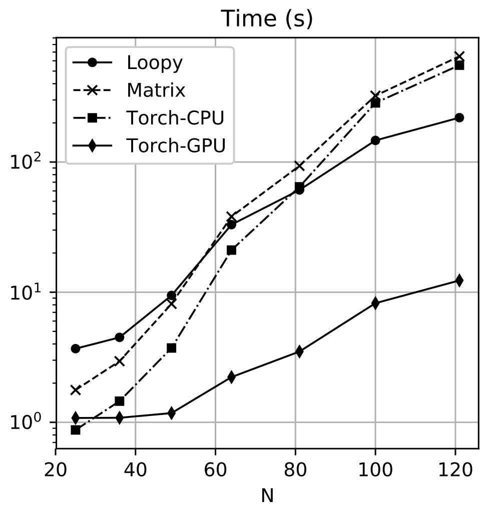


(b) $\lambda = 0.1$, payoff dominant

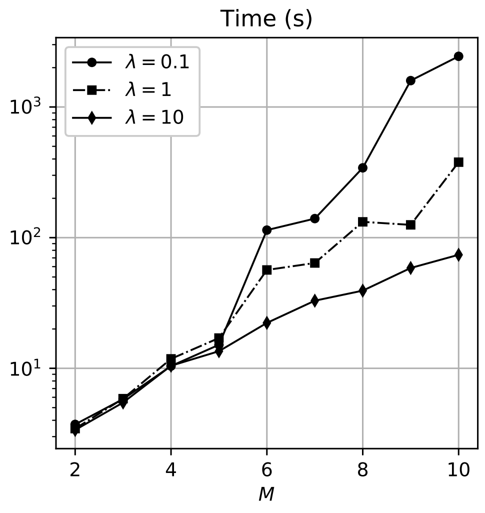
Figure 7: Semi-log plots of the generalised stag hunt running times for two different values of λ, as functions of the number of agents M, with $N = 100$ and $T = 10$. Results are averages over 5 runs with random initial states. *Loopy* refers to the direct Python implementation of belief propagation, while *Matrix* refers to the sparse-matrix `numpy`-based implementation, and *Torch-CPU*, *Torch-GPU* refer to the `pytorch` implementation ran in CPU and GPU hardware respectively. Since the running times of the payoff dominant regime increase much faster than those of the risk dominant regime, in (**b**) we have only been able to compute the running times for the loopy and Torch-GPU configurations.

running time is in turn influenced by two factors: the running time of one iteration of belief propagation and the number of iterations required for the algorithm to converge. To analyse the latter, Figure 9a shows the dependence of the number of iterations as a function of λ and the size of the model N. We observe that larger instances and lower values of λ correspond to more difficult tasks, which require more iterations of BP until convergence.

Finally, despite the GPU sparse-matrix belief propagation provides significant improvements in the running time of game with respect to any CPU method, we have encountered a memory limitation that has prevented us from scaling up beyond a 20×20 grid with 10 to 12 agents and the corresponding number of hares and stags. For instance, a 30×30 grid with $M = 20$ agents -and thus $S = 10$ stags and $H = 40$ hares- implies a $\mathbf{\Gamma}$ tensor of $1.863 \cdot 10^9$ elements, which needs almost 14GB of memory at double precision, currently unsupported by the GPUs.



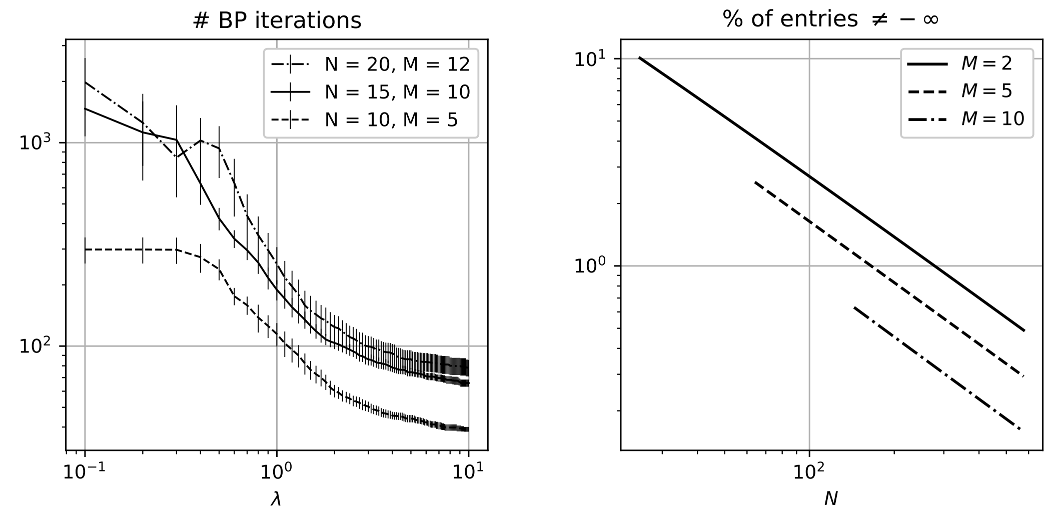
(a) $\lambda = 10$



(b) $N = 400$

Figure 8: Semi-log plots of the generalised stag hunt running times. Results are averages over 5 runs with random initial states. (**a**) The number of agents is increased with N as $M = 0.28(2N/7)$, such that the first game configuration is a 5×5 grid with two agents, and the density of objects in the grid remains nearly constant. The parameter λ is set to $\lambda = 10$. (**b**) The experiments are ran on a large $N = 400$ grid, and we observe the running times for three different values of λ.

Furthermore, since the methods in the `mrftools` library require this tensor to be allocated twice at some point, the resulting memory need becomes intractable for any GPU. One way to better scale up this problem would be precisely to take advantage of the sparsity, not only at the level of edges in the factors graph, but also within the factors. Note that a sparse matrix representation in this case requires sparsity with respect to the value $-\infty$. We believe this additional improvement is feasible and would notable boost the scalability of this approach, both by reducing the memory needs of the algorithm and the cpu-time required by belief propagation. Figure 9b shows the proportion of values different from $-\infty$ in the pairwise factors tensor $\mathbf{\Gamma}$, as a function of the grid size N, for three different values of M. We observe how the number of values different than $-\infty$ in fact decreases exponentially as N increases, and becomes also smaller for higher values of M.



(a) BP iterations (b) Sparsity

Figure 9: (**a**) Number of sparse-matrix BP iterations required for convergence as a function of λ for two different pairs of values of N and M, $T = 10$. Results are averages over 5 runs with random initial states and the error bars indicate the standard error of the mean. (**b**) Proportion of entries different from $-\infty$ in the pairwise factors tensor $\mathbf{\Gamma}$ as a function of N for three different values of M.

Chapter 4

Discussion

We have reviewed how a class of stochastic optimal control problems can be expressed as graphical inference problems [2], enabling approximate inference methods such as sparse-matrix belief propagation [1] to be applied to the intractable stochastic control computation. We have focused on an interesting special case of such class of problems, namely the generalised stag hunt, a multi-agent cooperative game that has allowed us to show how this modification of the widely known belief propagation algorithm can bring notable performance improvements to the optimal control computations in the class of Kullback-Leibler control problems.

As a particular result, we have shown how the graphical model of the generalised stag hunt can be transformed into a pairwise Markov random field, which can be fed into the `mrftools` [46] Python library to do inference on discrete Markov random fields, allowing us to solve in short time the optimal control problem for relatively large instances of this cooperative game.

4.1 Discussion

Sparse-matrix belief propagation was reported to deliver significant improvements in performance with respect to traditional loopy belief propagation, when tested on grid Markov random fields. However, we do not observe such speedup in the stag hunt model. Both models are, though, considerably different between each other. The grid Markov random fields considered in [1] have a large number of variables, up to 2^{18}, giving up to $523,264$ edges. On the other hand, these models variable cardinality is set, at most, of 64 different values. Then, while its sizes could be similar, shapes of these pairwise factor tensors and the ones from the stag hunt pairwise MRF model are completely different. In the latest, variable cardinalities are an order of magnitude larger, while the graph is way more sparsely connected, giving a much lower number of edges. Furthermore, in the grid models used tho evaluate the sparse-matrix belief propagation algorithm the log factors were chosen to be normally distributed with variance one, without any value at $-\infty$, which completely differs from our game representation, in which the pairwise factor tensor proportion of $-\infty$ values can go up to 99.99%.

On the other hand, though, the sparse-matrix form of belief propagation has an immediate advantage: since all the computations are encoded in tensor operations, it can be executed on GPUs. The straight off resulting parallelisation benefits overcome to a greater extent the loose of performance of sparse-matrix belief propagation with respect to the loopy implementation.

Finally, the observed growth of time and memory needs as the stag hunt model grows in size, either by increasing the size of the grid or the number of agents, could be addressed by taking advantage of the sparsity of the factors, specially the pairwise factor tensor $\mathbf{\Gamma}$. This work results open a path for a modification of sparse-matrix belief propagation in which not only the sparsity of the indexing operations but also the sparsity of the factors, which would be present in a wide variety of real-world Markov random field models, such as the stag hunt.

Particularly, addressing the memory limitation that we have encountered in this work would let much bigger instances of the stag hunt optimal control problem to be solved using sparse-matrix belief propagation, which would pave in turn the way to a better analysis of the phase transition between payoff and risk dominant regimes observed in [2].

4.2 Conclusions

In this master thesis we have successfully integrated a stochastic optimal control task, namely the computation of optimal trajectories for the stag hunt, in the recently introduced sparse-matrix belief propagation framework [1]. To do so, we have modified the probabilistic graphical model of the stag hunt proposed in [2] in order to formulate it as a Markov random field. This has allowed us, through the sparse-matrix belief propagation implementation, to take advantage of GPU parallelisation and provide a notable performance improvement with respect to traditional loopy belief propagation. Furthermore, with our implementation of the stag hunt model we have recovered the duality of optimal strategies, from payoff to risk dominant, previously observed on this cooperative game. Finally, we have identified that the limitations of our implementation, both in terms of memory and running time needs, could be addressed by a modification of the sparse-matrix belief propagation algorithm that leveraged the sparsity of the factors and not only the sparsity of the graph structure. This opens an interesting line of research that could enable the approximate optimal control computation on much larger models.

List of Figures

1 Factor graph representation of the generalised stag hunt game. Circles denote variable nodes, and squares denote factor nodes. Variables nodes in grey correspond to the *clamped* initial configuration. 12

2 Decomposition of the complex factor ψ_R into simpler factors involving at most three variables of small cardinality. Note that in the figure it is only shown the structure of the factor for stag i, which would be identically reproduced for the rest of the stags. 14

3 Pairwise factor graph representation of the generalised stag hunt game. The auxiliary variables z_{ij}, $i = 2, \ldots, M$, $j = 1, \ldots, S$ allow us to replace the ternary factors $\psi_{r_{i-1}}$ for the binary factors $\psi_{A_{(i-1)s}}$, $s = 1, 2, 3$. 15

4 Graphical structure of the stag-hunt game with $M = 2$ agents. This particular game structure can be understood as a pairwise Markov Random Field without further modifications, and can serve as a ground truth model to assess the correctness of the proposed decomposition. 17

5 Exact inference on the stag hunt: optimal control obtained using loopy belief propagation on the tree-shaped ground truth MRF model of the game, for $M = 2$ hunters (circles) in a 5×5 grid with $S = 1$ stag (big diamond) and $H = 4$ hares (small diamonds), within a time horizon of $T = 4$. (**a**) For $\lambda = 10$, the optimal control is risk dominant and the hunters go for the hares. (**b**) For $\lambda = 0.1$ the optimal control is payoff dominant and the hunters cooperate to hunt the stag. 26

6 Approximate inference on the stag hunt: optimal control obtained using GPU sparse-matrix belief propagation on the pairwise MRF model of the game, for $M = 10$ hunters (circles) in a 20×20 grid with $S = \lfloor M/2 \rfloor$ stags (big diamonds) and $H = 2M$ hares (small diamonds), $R_s = -10$, $R_h = -2$, within a time horizon of $T = 10$. (**a**) For $\lambda = 10$, the optimal control is risk dominant and the hunters go for the hares. (**b**) For $\lambda = 0.1$ the optimal control is payoff dominant and the hunters cooperate to hunt the stag. 27

7 Semi-log plots of the generalised stag hunt running times for two different values of λ, as functions of the number of agents M, with $N = 100$ and $T = 10$. Results are averages over 5 runs with random initial states. *Loopy* refers to the direct Python implementation of belief propagation, while *Matrix* refers to the sparse-matrix `numpy`-based implementation, and *Torch-CPU*, *Torch-GPU* refer to the `pytorch` implementation ran in CPU and GPU hardware respectively. Since the running times of the payoff dominant regime increase much faster than those of the risk dominant regime, in (**b**) we have only been able to compute the running times for the loopy and Torch-GPU configurations. 28

8 Semi-log plots of the generalised stag hunt running times. Results are averages over 5 runs with random initial states. (**a**) The number of agents is increased with N as $M = 0.28(2N/7)$, such that the first game configuration is a 5×5 grid with two agents, and the density of objects in the grid remains nearly constant. The parameter λ is set to $\lambda = 10$. (**b**) The experiments are ran on a large $N = 400$ grid, and we observe the running times for three different values of λ. . . . 29

9 (**a**) Number of sparse-matrix BP iterations required for convergence as a function of λ for two different pairs of values of N and M, $T = 10$. Results are averages over 5 runs with random initial states and the error bars indicate the standard error of the mean. (**b**) Proportion of entries different from $-\infty$ in the pairwise factors tensor $\mathbf{\Gamma}$ as a function of N for three different values of M. 30

Bibliography

[1] Bixler, R. & Huang, B. Sparse-matrix belief propagation. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, 611–620 (2018).

[2] Kappen, H. J., Gómez, V. & Opper, M. Optimal control as a graphical model inference problem. *Machine learning* **87**, 159–182 (2012).

[3] Bellman, R. *Dynamic Programming* (Princeton University Press, Princeton, NJ, USA, 1957), 1 edn.

[4] Bertsekas, D. P. *Dynamic Programming and Optimal Control* (Athena Scientific, 2000), 2nd edn.

[5] Sutton, R. S. & Barto, A. G. *Introduction to Reinforcement Learning* (MIT Press, Cambridge, MA, USA, 1998), 1st edn.

[6] Szepesvari, C. *Algorithms for Reinforcement Learning* (Morgan and Claypool Publishers, 2010).

[7] Jaderberg, M. *et al.* Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science* **364**, 859–865 (2019).

[8] Silver, D. *et al.* A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**, 1140–1144 (2018).

[9] Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).

[10] Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (The MIT Press, 2016).

[11] Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, 1889–1897 (2015).

[12] Jordan, M. I. & Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science* **349**, 255–260 (2015).

[13] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988).

[14] Koller, D. & Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning (MIT Press, 2009).

[15] Yedidia, J. S., Freeman, W. T. & Weiss, Y. Exploring artificial intelligence in the new millennium. chap. Understanding Belief Propagation and Its Generalizations, 239–269 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003).

[16] Kschischang, F. R., Frey, B. J., Loeliger, H.-A. *et al.* Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory* **47**, 498–519 (2001).

[17] Wainwright, M. J. & Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* **1**, 1–305 (2008).

[18] Murphy, K. P., Weiss, Y. & Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, 467–475 (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999).

[19] Ziebart, B. D. Modeling purposeful adaptive behavior with the principle of maximum causal entropy (2010). PhD thesis.

[20] Toussaint, M. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, 1049–1056 (ACM, 2009).

[21] Todorov, E. General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 4286–4292 (IEEE, 2008).

[22] Todorov, E. Efficient computation of optimal actions. *Proceedings of the national academy of sciences* **106**, 11478–11483 (2009).

[23] Todorov, E. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, 1369–1376 (2007).

[24] Kappen, H. J. Linear theory for control of nonlinear stochastic systems. *Physical review letters* **95**, 200201 (2005).

[25] Jonsson, A. & Gómez, V. Hierarchical linearly-solvable markov decision problems. In *26th International Conference on Automated Planning and Scheduling*, ICAPS'16, 193–201 (AAAI Press, 2016).

[26] Saxe, A. M., Earle, A. C. & Rosman, B. Hierarchy through composition with multitask LMDPs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3017–3026 (JMLR. org, 2017).

[27] Todorov, E. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, 1856–1864 (2009).

[28] Neu, G. & Gómez, V. Fast rates for online learning in Linearly Solvable Markov Decision Processes. In *Proceedings of the 2017 Conference on Learning Theory*, vol. 65 of *Proceedings of Machine Learning Research*, 1567–1588 (PMLR, 2017).

[29] Williams, G., Aldrich, A. & Theodorou, E. A. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics* **40**, 344–357 (2017).

[30] Gómez, V., Kappen, H. J., Peters, J. & Neumann, G. Policy search for path integral control. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 482–497 (Springer, 2014).

[31] Gómez, V., Thijssen, S., Symington, A. C., Hailes, S. & Kappen, H. J. Real-time stochastic optimal control for multi-agent quadrotor systems. In *26th International Conference on Automated Planning and Scheduling* (2016).

[32] Van Den Broek, B., Wiegerinck, W. & Kappen, B. Graphical model inference in optimal control of stochastic multi-agent systems. *Journal of Artificial Intelligence Research* **32**, 95–122 (2008).

[33] Matsubara, T., Gómez, V. & Kappen, H. J. Latent Kullback Leibler control for continuous-state systems using probabilistic graphical models. *30th Conference on Uncertainty in Artificial Intelligence (UAI)* (2014).

[34] Abbasi-Yadkori, Y., Bartlett, P., Chen, X. & Malek, A. Large-scale Markov decision problems with KL control cost and its application to crowdsourcing. In *International Conference on Machine Learning*, 1053–1062 (2015).

[35] Thalmeier, D., Gómez, V. & Kappen, H. J. Action selection in growing state spaces: control of network structure growth. *Journal of Physics A: Mathematical and Theoretical* **50**, 034006 (2016).

[36] Helbing, D., Schönhof, M., Stark, H.-U. & Hołyst, J. A. How individuals learn to take turns: Emergence of alternating cooperation in a congestion game and the prisoner's dilemma. *Advances in Complex Systems* **8**, 87–116 (2005).

[37] Friedman, D. & Oprea, R. A continuous dilemma. *American Economic Review* **102**, 337–63 (2012).

[38] Skyrms, B. *The stag hunt and the evolution of social structure* (Cambridge University Press, 2004).

[39] Skyrms, B. *Evolution of the social contract* (Cambridge University Press, 2014).

[40] Yoshida, W., Dolan, R. J. & Friston, K. J. Game theory of mind. *PLoS computational biology* **4**, e1000254 (2008).

[41] Yoshida, W., Seymour, B., Friston, K. J. & Dolan, R. J. Neural mechanisms of belief inference during cooperative games. *Journal of Neuroscience* **30**, 10744–10751 (2010).

[42] Yoshida, W. *et al.* Cooperation and heterogeneity of the autistic mind. *Journal of Neuroscience* **30**, 8815–8818 (2010).

[43] Lauritzen, S. L. & Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)* 157–224 (1988).

[44] Jordan, M. I. *Learning in graphical models*, vol. 89 (Springer Science & Business Media, 1998).

[45] Murphy, K. P., Weiss, Y. & Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 467–475 (Morgan Kaufmann Publishers Inc., 1999).

[46] Bixler, R. & Huang, B. mrftools library. `https://bitbucket.org/berthuang/mrftools/src/master/` (2018).