
SAMPLING FROM THE BAYESIAN POSTER OF AN AGENT-BASED MODEL GIVEN PARTIAL OBSERVATIONS

Daniel Tang

Leeds Institute for Data Analytics, University of Leeds, UK*
D.Tang@leeds.ac.uk

Nicholas Malleson

Leeds Institute for Data Analytics University of Leeds, UK

October 6, 2021

ABSTRACT

The discipline of data assimilation (DA), sometimes known as data fusion, addresses the problem of how to make use of noisy, incomplete experimental observations to provide information about the time evolution of unobserved properties of a dynamical system. DA has developed rapidly in applications such as weather forecasting but relatively little progress has been made in developing techniques that are applicable when the dynamical system is an agent based model. ABMs consist of ‘agents’ which often make discrete choices from a number of possible actions, meaning the space of model trajectories is not continuous and we cannot use techniques based on gradient ascent, such as 4D-VAR, that require the gradient of the posterior in this space. In addition, a set of observations will typically refute the vast majority of model trajectories, making it difficult to even identify trajectories that could have given rise to the observations, and so making it challenging to use algorithms that rely on perturbing the current solution such as non-gradient optimisation or most sampling algorithms.

Here we present an algorithm that generates samples of the time evolution of an agent based model, given a set of noisy, incomplete experimental observations of the system. The algorithm approximates the set of possible trajectories as a linear program and uses an extension of the simplex algorithm to provide a proposal function for Markov-Chain-Monte-Carlo sampling.

We demonstrate the algorithm by performing data assimilation in an agent-based, spatial predator-prey model.

Keywords Data assimilation, Bayesian inference, Agent based model, Integer linear programming, predator prey model

1 Introduction

1.1 Background

Data assimilation (DA) is a technique that has been widely used in the physical sciences such as meteorology (Kalnay, 2003), oceanography (Bertino, Evensen, & Wackernagel, 2003) and the earth sciences more broadly (Reichle, 2008). Specific applications of DA to ABMs are much rarer. Examples of data assimilation using well known DA methods such as Particle Filters and variants of the Kalman Filter applied to ABMs include models of crime (Lloyd, Santitissadeekorn, & Short, 2016), bus routes (Kieu, Malleson, & Heppenstall, 2020), pedestrian dynamics (Wang & Hu, 2015; Ward, Evans, & Malleson, 2016; Clay, Kieu, Ward, Heppenstall, & Malleson, 2020; Malleson et al., 2020); and population movement (Lueck, Rife, Swarup, & Uddin, 2019). Many DA algorithms rely on the differentiability of the model (Lewis, Lakshmivaran, & Dhall, 2006).

*This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 757455)

2 Formulation of the problem

Suppose we have a timestepping ABM where agents have a finite number of possible internal states and a finite number of ways of acting on their world. Given this, we can define an ABM as:

- A vector of agent actions $\mathcal{A} = \langle A_0 \dots A_n \rangle$
- A vector of agent states $\mathcal{S} = \langle S_0 \dots S_m \rangle$
- An *agent timestep*, $\pi : \mathbb{Z} \times \mathbb{Z}^{m+1} \times \mathbb{Z} \rightarrow \mathbb{R}$, which defines the probability that an agent will act in a particular way such that $\pi(\psi, \Phi, a)$ gives the probability that an agent in state S_ψ , surrounded by agents, Φ , will perform action A_a (where Φ is a vector whose i^{th} element is the number of agents in state S_i at the start of the timestep).
- An *action function*, $F : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}^{m+1}$, which defines the effect of an action on the world such that $F(\psi, a)$ returns a vector, Φ , whose i^{th} element gives the number of agents in state S_i that result from an agent in state S_ψ performing act A_a (including the final state of the acting agent).

As a simple illustration, consider the simple “cat and mouse” ABM, which consists of just two gridsquares, left and right, within which can roam cats and mice, so the vector of agent states is, $\mathcal{S} = \langle \text{left cat, right cat, left mouse, right mouse} \rangle$. In any given timestep, agents can either move or stay still, so the vector of actions is $\mathcal{A} = \langle \text{move, stay still} \rangle$.

A cat moves with probability 0.5, irrespective of other agents, while a mouse will move if there is one or more cats on the same gridsquare, otherwise it will stay still. So, the agent timestep function is

$$\begin{aligned} \pi(\psi, \Phi, 0) &= \begin{cases} 0.5 & \text{if } \psi \in \{0, 1\} \\ 1 & \text{if } \psi = 2, \Phi_1 > 0 \text{ or } \psi = 3, \Phi_2 > 0 \\ 0 & \text{otherwise} \end{cases} \\ \pi(\psi, \Phi, 1) &= \begin{cases} 0.5 & \text{if } \psi \in \{0, 1\} \\ 1 & \text{if } \psi = 2, \Phi_1 = 0 \text{ or } \psi = 3, \Phi_2 = 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (1)$$

The action function expresses the movement of the agents

$$\begin{aligned} F(0, 0) &= \{0, 1, 0, 0\} \\ F(1, 0) &= \{1, 0, 0, 0\} \\ F(2, 0) &= \{0, 0, 0, 1\} \\ F(3, 0) &= \{0, 0, 1, 0\} \\ F(0, 1) &= \{1, 0, 0, 0\} \\ F(1, 1) &= \{0, 1, 0, 0\} \\ F(2, 1) &= \{0, 0, 1, 0\} \\ F(3, 1) &= \{0, 0, 0, 1\} \end{aligned} \quad (2)$$

2.1 Trajectories

Let a model timestep consist of a matrix E whose elements $e_{\psi a}$ are the number of agents in state ψ that perform act a in this timestep. For example, the timestep shown in Figure 1 for the cat and mouse example would be

$$E = \begin{matrix} & \begin{matrix} A_0 & A_1 \end{matrix} \\ \begin{matrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{matrix} & \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

where all elements are zero except those representing agent S_1 performing action A_0 and agent S_2 performing action A_1 .

Finally, let a model trajectory be a tensor, T , consisting of a number of model timesteps. We use the notation T^t to denote the t^{th} timestep matrix, T_ψ^t to denote the ψ^{th} row of the t^{th} timestep matrix and $T_{\psi a}^t$ to denote the a^{th} element of the ψ^{th} row of the t^{th} timestep. By convention, we’ll let indices begin at 0. Note that this tensor will generally be very large for more realistic models, but also very sparse, so it can be dealt with computationally using a sparse representation.

It will occasionally be useful to refer to the set of all tensors of a given shape. For this we’ll use \mathbb{R} adorned with the number of elements in each index position. For example, a trajectory representing N timesteps of a model with S agent states and A actions must be a member of the set of tensors \mathbb{R}_{SA}^N .

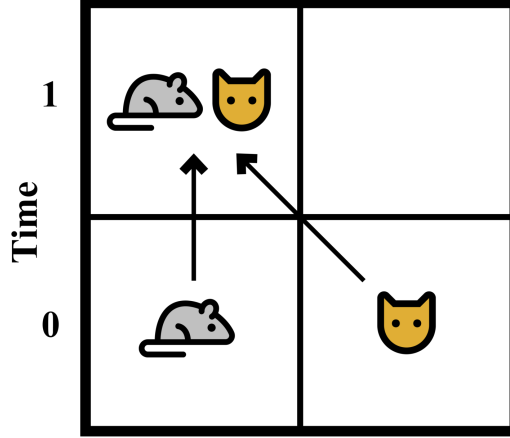


Figure 1: A simple cat and mouse model.

A tensor must satisfy a number of constraints in order to be a valid trajectory of an ABM. Since the elements of a trajectory are counts, they must all be non-negative integers, we'll call this the *non-negative integer constraint* and define the set of all non-negative integer tensors

$$\mathcal{I}_{SA}^N = \{T \in \mathbb{R}_{SA}^N : \forall t, \psi, a : T_{\psi a}^t \geq 0, T_{\psi a}^t \in \mathbb{Z}\} \quad (3)$$

A trajectory must also be *continuous* by which we mean that the number of agents in each state at the end of timestep $t - 1$ must be the number of agents in each state at the beginning of timestep t . We call this the *continuity constraint* and define the set of continuous tensors:

$$\mathcal{C}_{SA}^N(F) = \left\{ T \in \mathbb{R}_{SA}^N : \forall t \in 1 \dots N - 1 : \forall \phi : \sum_{\psi, a} F(\psi, a)_{\phi} T_{\psi a}^{t-1} - \sum_a T_{\phi a}^t = 0 \right\} \quad (4)$$

So, the set of valid trajectories, $\mathcal{T}_{SA}^N(F)$, is given by the set of tensors that satisfy 3 and 4.

$$\mathcal{T}_{SA}^N(F) = \mathcal{I}_{SA}^N \cap \mathcal{C}_{SA}^N(F) \quad (5)$$

2.2 The posterior

If we let Ψ^t be the vector whose ψ^{th} element is the number of agents in state S_{ψ} at the beginning of timestep t , then the prior probability of a trajectory is

$$P(T) = \begin{cases} P(\Psi^0 = T^0 \mathbf{1}) \prod_{\psi, t} P(T_{\psi}^t | \Psi^t = T^t \mathbf{1}) & \text{if } T \in \mathcal{T}_{SA}^N(F) \\ 0 & \text{otherwise} \end{cases}$$

where $P(\Psi^0)$ is our prior beliefs about the state at time $t = 0$, and we use $\mathbf{1}$ to denote a vector whose elements are all 1.

The probability that a single agent in a given state will perform an action, given the state of the other agents, Ψ^t , is given by the agent timestep function, $\pi(\psi, \Psi^t, a)$, so the joint probability that Ψ_{ψ}^t agents will perform actions T_{ψ}^t in an environment of other agents Ψ^t , is given by the multinomial distribution

$$P(T_{\psi}^t | \Psi^t) = \Psi_{\psi}^t! \prod_a \frac{\pi(\psi, \Psi^t, a)^{T_{\psi a}^t}}{T_{\psi a}^t!}.$$

So the prior probability of a trajectory is

$$P(T) = \begin{cases} P(\Psi^0 = T^0 \mathbf{1}) \prod_{t, \psi} (T_{\psi}^t \cdot \mathbf{1})! \prod_a \frac{\pi(\psi, T^t \mathbf{1}, a)^{T_{\psi a}^t}}{T_{\psi a}^t!} & \text{if } T \in \mathcal{T}_{SA}^N(F) \\ 0 & \text{otherwise} \end{cases}$$

Suppose now we have a set of noisy, aggregate observations, Ω , that have a likelihood function $P(\Omega|T)$. By Bayes' rule, we have

$$P(T|\Omega) \propto P(\Omega|T) P(T)$$

Without loss of generality, we take Ω to consist of some number of observations that are independent of each other given the trajectory, so that the members $(\omega, v) \in \Omega$ consist of a stochastic observation operator ω and an observed value v (which may be a vector). We write $P(\omega(T) = v)$ to denote the probability of observation operator ω making observation v on trajectory T . So

$$P(\Omega|T) = \prod_{(\omega, v) \in \Omega} P(\omega(T) = v)$$

The posterior can now be written as

$$P(T|\Omega) \propto \begin{cases} P(\Psi^0 = T^0 \mathbf{1}) \prod_{(\omega, v) \in \Omega} P(\omega(T) = v) \prod_{t, \psi, a} \left(T_{\psi}^t \cdot \mathbf{1} \right)!^{\frac{\pi(\psi, T^t \mathbf{1}, a) T_{\psi a}^t}{T_{\psi a}^t}} & \text{if } T \in \mathcal{T}_{SA}^N(F) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The problem we're interested in is how to sample from this distribution. In many practical applications, sampling is difficult because the posterior has zero probability for the vast majority of tensors (i.e. most tensors are not trajectories, contain an impossible action or are refuted by the observations). Even though we can generate trajectories that fit the prior by simply performing a forward execution of the model from an initial state drawn from the prior, if the observations refute the trajectory the probability falls to zero. It doesn't take many observations until the probability of randomly choosing a trajectory that fits the observations becomes very small indeed. So simple techniques such as rejection sampling, for example, are not practical. Techniques based on particle filtering may have more success but, for similar reasons, will likely soon reach a state containing a set of particles, none of which can be fit to the observations.

In this paper we'll show how the Metropolis-Hastings algorithm can be used to generate samples from equation 6. The challenge will be to create a proposal function which randomly generates a proposed next sample given the current one. For example, a common strategy with Metropolis-Hastings is to generate a new sample by perturbing one or more elements of the previous sample at random. However, if we do this with an ABM trajectory it's very unlikely that the perturbed tensor will be a trajectory that contains only possible actions and satisfies the observations. So, the proposed next sample would almost certainly be rejected and we'd probably end up stuck on the first sample until we grew old.

3 Approximating the support of the posterior

We solve this problem by first approximating the support of the posterior, $\text{supp}(P(T_{\psi a}^t|\Omega))$ (i.e. the set of trajectories that have non-zero probability).

From equation 6

$$\begin{aligned} \text{supp}(P(T|\Omega)) &= \mathcal{T}_{SA}^N \cap \\ &\quad \text{supp}(P(\Psi^0 = T^0 \mathbf{1})) \cap \\ &\quad \bigcap_{(\omega, v) \in \Omega} \text{supp}(P(\omega(T) = v)) \cap \\ &\quad \bigcap_{t, \psi, a} (\text{supp}(\pi(\psi, T^t \mathbf{1}, a)) \cup \{T : T_{\psi a}^t = 0\}) \end{aligned} \quad (7)$$

i.e. in order for T to have non-zero posterior probability, it must be a trajectory of the ABM, it must have a start state that has non-zero prior probability, all the observation likelihoods must be non-zero and each element of T must either denote an agent action with non-zero probability or must be equal to zero (to account for the fact that $0^0 = 1$ in the exponent terms in 6).

3.1 Fermionic Trajectories

In the following we will consider only *Fermionic trajectories*, which we define to be any trajectory whose elements are all either 0 or 1. i.e. a Fermionic trajectory must not only satisfy the constraints in 3 and 4 but must also satisfy the *Fermionic constraints*

$$\mathcal{B}_{SA}^N = \{T \in \mathbb{R}_{SA}^N : \forall t, \psi, a : T_{\psi a}^t \in \{0, 1\}\} \quad (8)$$

This can be interpreted as meaning that no two agents in the same state at the same time can perform the same action. Notice that the Fermionic constraints imply the non-negative integer constraints, so the set of Fermionic trajectories, \mathcal{F}_{SA}^N , is the set of tensors that satisfy 8 and 4.

$$\mathcal{F}_{SA}^N(F) = \mathcal{B}_{SA}^N \cap \mathcal{C}_{SA}^N(F) \quad (9)$$

If we now condition the posterior on $T \in \mathcal{B}_{SA}^N$ then the support of the posterior becomes

$$\text{supp}(P(T|\Omega, T \in \mathcal{B}_{SA}^N)) = \mathcal{B}_{SA}^N \cap \text{supp}(P(T|\Omega)) \quad (10)$$

In practice, the number of agents in an ABM is usually much smaller than the number of agent states, so it is overwhelmingly likely that a sample from the posterior will be Fermionic, and so intersection with \mathcal{B}_{SA}^N will often have very little effect on the posterior. However, section ** describes how to extend the algorithm to apply to non-Fermionic trajectories when the Fermionic constraints are not acceptable.

3.2 Convex \mathcal{B} -polyhedra and \mathcal{B} -distributions

We define a \mathcal{B} -polyhedron to be a set of binary vectors

$$\mathcal{B}_{SA}^N \cap \mathcal{P}$$

where $\mathcal{P} \subset \mathcal{R}_{SA}^N$ is a convex polyhedron in the sense that it can be expressed as a set of tensors that satisfy some number of linear constraints on their elements. More formally, there exists a tensor $C \in \mathbb{R}_{NJ}^{SA}$ and vectors $L \in \mathbb{R}_J$ and $U \in \mathbb{R}_J$ such that:

$$\mathcal{P} = \left\{ T \in \mathbb{R}_{SA}^N : L \leq \sum_{t,\psi,a} C_t^{\psi a} T_{\psi a}^t \leq U \right\}$$

Theorem 1. For any set \mathcal{X} there exists a convex polyhedron \mathcal{P} such that

$$\mathcal{B}_{SA}^N \cap \mathcal{X} = \mathcal{B}_{SA}^N \cap \mathcal{P}$$

Proof. Let $\mathcal{Y} = \mathcal{B}_{SA}^N \cap \mathcal{X}$ and let \mathcal{P} be the convex hull of \mathcal{Y} . Clearly if a tensor $T \in \mathcal{Y}$ then $T \in \mathcal{B}_{SA}^N \cap \mathcal{P}$. To show the converse, suppose there is a member $T' \in \mathcal{B}_{SA}^N \cap \mathcal{P}$. Since $T' \in \mathcal{P}$ then there must be a convex combination of points in \mathcal{Y} that equals T' . Call those points \mathcal{Z} . However, since $T' \in \mathcal{B}_{SA}^N$ and $\mathcal{Z} \subset \mathcal{B}_{SA}^N$ and no member of \mathcal{B}_{SA}^N is a convex combination of any other member then T' must be a member of \mathcal{Z} and so $T' \in \mathcal{Y}$. So $\mathcal{B}_{SA}^N \cap \mathcal{X} = \mathcal{B}_{SA}^N \cap \mathcal{P}$. \square

Under the assumption that all trajectories are Fermionic, we can rewrite equation 7 in the form

$$\begin{aligned} \mathcal{B}_{SA}^N \cap \text{supp}(P(T|\Omega)) &= \mathcal{B}_{SA}^N \cap \mathcal{C}_{SA}^N(F) \cap \\ &\quad \mathcal{B}_{SA}^N \cap \text{supp}(P(\Psi^0 = T^0 \mathbf{1})) \cap \\ &\quad \bigcap_{(\omega,v) \in \Omega} \mathcal{B}_{SA}^N \cap \text{supp}(P(\omega(T) = v)) \cap \\ &\quad \bigcap_{t,\psi,a} \mathcal{B}_{SA}^N \cap (\text{supp}(\pi(\psi, T^t \mathbf{1}, a)) \cup \{T : T_{\psi a}^t = 0\}) \end{aligned} \quad (11)$$

which is an intersection of terms in the form $\mathcal{B}_{SA}^N \cap \mathcal{X}$, so each term can equivalently be expressed as a \mathcal{B} -polyhedron.

The union in the last line would be inconvenient to deal with computationally so we transform it away in the following way. Let $\mathcal{P}_{\psi a}^t$ be a convex polyhedron such that

$$\mathcal{B}_{SA}^N \cap \mathcal{P}_{\psi a}^t = \mathcal{B}_{SA}^N \cap \text{supp}(\pi(\psi, T^t \mathbf{1}, a))$$

so

$$\mathcal{B}_{SA}^N \cap (\text{supp}(\pi(\psi, T^t \mathbf{1}, a)) \cup \{T : T_{\psi a}^t = 0\}) = \mathcal{B}_{SA}^N \cap (\mathcal{P}_{\psi a}^t \cup \{T : T_{\psi a}^t = 0\})$$

If we let

$$\mathcal{P}_{\psi a}^t = \left\{ T \in \mathbb{R}_{SA}^N : L \leq \sum_{s,\phi,b} C_s^{\phi b} T_{\phi b}^s \leq U \right\}$$

for some $C \in \mathbb{R}_{NJ}^{SA}$ and $L \in \mathbb{R}_J$ and $U \in \mathbb{R}_J$ then we can remove the union using the identity

$$\begin{aligned} \mathcal{B}_{SA}^N \cap \left(\left\{ T \in \mathbb{R}_{SA}^N : L \leq \sum_{s,\phi,b} C_s^{\phi b} T_{\phi b}^s \leq U \right\} \cup \{T : T_{\psi a}^t = 0\} \right) &= \\ \mathcal{B}_{SA}^N \cap \left\{ T \in \mathbb{R}_{SA}^N : -\infty \leq \sum_{s,\phi,b} C_s^{\phi b} T_{\phi b}^s + (\overline{B} - U) T_{\psi a}^t \leq \overline{B} \right\} \cap \\ \mathcal{B}_{SA}^N \cap \left\{ T \in \mathbb{R}_{SA}^N : \underline{B} \leq \sum_{s,\phi,b} C_s^{\phi b} T_{\phi b}^s + (\underline{B} - L) T_{\psi a}^t \leq \infty \right\} \end{aligned} \quad (12)$$

where the elements of $\bar{B} \in \mathbb{R}_I$ are defined as

$$\bar{B}_i = \frac{\sum_{s,\phi,b} (|C_{si}^{\phi b}| + C_{si}^{\phi b})}{2}$$

and the elements of $B \in \mathbb{R}_I$ are defined as

$$B_i = \frac{\sum_{s,\phi,b} (|C_{si}^{\phi b}| - C_{si}^{\phi b})}{2}$$

To see why this identity holds, consider the right hand polyhedrons in 12. When $T_{\psi a}^t = 1$ the first is equal to $\sum_{s,\phi,b} C_s^{\phi b} T_{\phi b}^s \leq U$ and the second is equal to $L \leq \sum_{s,\phi,b} C_s^{\phi b} T_{\phi b}^s$ so their intersection is $\mathcal{P}_{\psi a}^t$, whereas when $T_{\psi a}^t = 0$ the right hand side terms give $B \leq \sum_{s,\phi,b} C_s^{\phi b} T_{\phi b}^s \leq \bar{B}$. But the elements B_i and \bar{B}_i are an lower and upper bounds on the value of $\sum_{s,\phi,b} C_{si}^{\phi b} T_{\phi b}^s$ given that $T \in \mathcal{B}_{SA}^N$, so this is satisfied for all \mathcal{B}_{SA}^N such that $T_{\psi a}^t = 0$.

Using this transformation the Fermionic support of the posterior can be reduced to the intersection of \mathcal{B} -polyhedra. Given the identity

$$(\mathcal{B}_{SA}^N \cap P_1) \cap (\mathcal{B}_{SA}^N \cap P_2) = \mathcal{B}_{SA}^N \cap (P_1 \cap P_2)$$

we can calculate 11, and thereby represent the support of the posterior as a set of linear constraints, by calculating each term and simply concatenating their linear constraints.

The idea of a \mathcal{B} -polyhedron as the support for a probability distribution naturally leads to the idea of a \mathcal{B} -distribution which is a probability distribution which is a discrete distribution defined on the points of a \mathcal{B} -polyhedron. From the above, it can be seen that the posterior distribution of an ABM trajectory is a \mathcal{B} -distribution.

As an illustration, consider two timesteps of the cat and mouse model. Suppose we flip a fair coin to decide whether each agent state is occupied or empty at $t = 0$ and that we observe a cat in the left grid-square at time $t = 1$. Our aim is to construct a set of linear constraints, Q , such that

$$\mathcal{B}_{42}^2 \cap \text{supp}(P(T|\Omega)) = \mathcal{B}_{42}^2 \cap Q$$

Working through 11 term by term, the $\mathcal{B}_{SA}^N \cap \mathcal{C}_{SA}^N(F)$ term is equivalent to a \mathcal{B} -polyhedron with the continuity constraints in 4, which are already in linear form so we're done. The second term is the support of the prior. This constrains each agent state at $t = 0$ to be at most 1, which can be expressed as

$$\bigcap_{\psi} \{T : T_{\psi 0}^0 + T_{\psi 1}^0 \leq 1\}$$

The third term is the support of the observation. Since we observe a cat in the left grid-square at time $t = 1$ we need to add the constraint

$$T_{00}^1 + T_{01}^1 = 1$$

The final term guards against impossible actions. The only impossible actions are a mouse staying put when there is a cat on the same gridsquare or moving when there are no cats, which translates to the four cases

$$\begin{aligned} \text{supp}(\pi(2, T^t \mathbf{1}, 0)) &= \{T : -T_{00}^t - T_{01}^t \leq -1\} \\ \text{supp}(\pi(3, T^t \mathbf{1}, 0)) &= \{T : -T_{10}^t - T_{11}^t \leq -1\} \\ \text{supp}(\pi(2, T^t \mathbf{1}, 1)) &= \{T : T_{00}^t + T_{01}^t \leq 0\} \\ \text{supp}(\pi(3, T^t \mathbf{1}, 1)) &= \{T : T_{10}^t + T_{11}^t \leq 0\} \end{aligned}$$

Using the identity in 12 to take the union of each of these with $\{T : T_{\psi a}^t = 0\}$ gives the four constraints

$$\begin{aligned} -T_{00}^t - T_{01}^t + T_{20}^t &\leq 0 \\ -T_{10}^t - T_{11}^t + T_{30}^t &\leq 0 \\ T_{00}^t + T_{01}^t + 2T_{21}^t &\leq 2 \\ T_{10}^t + T_{11}^t + 2T_{31}^t &\leq 2 \end{aligned}$$

for each timestep $t = 0$ and $t = 1$.

Taken together, these constraints define a \mathcal{B} -polyhedron that is the set of Fermionic trajectories for the cat and mouse ABM, and when combined with equation 6 defines $P(T|\Omega)$ as a \mathcal{B} -distribution.

4 Transforming between representations of a \mathcal{B} -polyhedron

Now that we can express the support of the posterior as a \mathcal{B} -polyhedron in the form

$$\text{supp}(P(T|\Omega)) = \mathcal{B}_{SA}^N \cap \left\{ T : L \leq \sum_{t,\psi,a} C_t^{\psi a} T_{\psi a}^t \leq U \right\} \quad (13)$$

we now describe some different ways of representing the same \mathcal{B} -polyhedron and methods of transforming from one representation to another. This will be useful in the development that follows.

4.1 The standard form of a \mathcal{B} -polyhedron

Since there are many ways of representing a polyhedron, it will be useful to define a standard form when representing \mathcal{B} -polyhedra. We'll use the form

$$\mathcal{B}(Q, N, b, L, U) = \left\{ X : X = Q \begin{pmatrix} X_B \\ X_N \end{pmatrix}, X_B = N X_N + b, X_B \in \mathbb{Z}^I, L \leq X_B \leq U, X \in \{0, 1\}^K, X_N \in \{0, 1\}^J \right\} \quad (14)$$

where $X \in \mathbb{R}^K$, $X_N \in \mathbb{R}^J$ and $X_B \in \mathbb{R}^I$ are now vectors and Q is a $K \times (I + J)$ matrix that selects $J \leq K \leq I + J$ elements from $(X_B | X_N)^T$ (i.e. each column of Q has at most one element equal to 1, each row has exactly one element equal to 1 and all other elements are 0). We'll call the elements of X_B "basic-variables" and the elements of X_N "non-basic variables"².

Equation 13 can easily be expressed in the form 14 by "flattening" the trajectory, T , into a J -dimensional vector (without changing the values of the elements) using a tensor $R \in \mathbb{R}_{NJ}^{SA}$ where $J = N S A$ (i.e. for every (t, ψ, a) there is exactly one j such that $R_{tj}^{\psi a} = 1$ and for every j there is exactly one (t, ψ, a) such that $R_{tj}^{\psi a} = 1$ and all other elements are zero) so that

$$X_N = \sum_{t,\psi,a} R_t^{\psi a} T_{\psi a}^t$$

and letting N be the $I \times J$ matrix

$$N = \sum_{t,\psi,a} C_t^{\psi a} R_t^{\psi a}$$

and finally letting Q be the matrix such that $X_N = Q(X_B | X_N)$ and $b = \mathbf{0}$. The constraint $X_B \in \mathbb{I}^I$ is satisfied since all the elements of N are integers in our case.

4.2 The pivot transformation

The introduction of the matrix Q in 14 allows us to transform the representation without changing the \mathcal{B} -polyhedron that is represented. We now describe the "pivot transformation" which will be a central operation in the sampling algorithm, and should look familiar to anyone acquainted with the simplex algorithm (see e.g. (Vanderbei, 2020)).

We begin by transforming 14 to the equivalent form

$$\mathcal{B} = \left\{ X : X = Q \begin{pmatrix} X_B \\ X_N \end{pmatrix}, (I - N) \begin{pmatrix} X_B \\ X_N \end{pmatrix} = b, \begin{pmatrix} L \\ \mathbf{0} \end{pmatrix} \leq \begin{pmatrix} X_B \\ X_N \end{pmatrix} \leq \begin{pmatrix} U \\ \mathbf{1} \end{pmatrix}, X \in \{0, 1\}^K, \begin{pmatrix} X_B \\ X_N \end{pmatrix} \in \mathbb{I}^{I+J} \right\} \quad (15)$$

If we let S_{ij} be the permutation matrix such that

$$\begin{pmatrix} X'_B \\ X'_N \end{pmatrix} = S_{ij} \begin{pmatrix} X_B \\ X_N \end{pmatrix}$$

has the effect of swapping the i^{th} element of X_B with the j^{th} element of X_N then we can change variables in the equality constraint so that

$$(I - N) S_{ij}^{-1} \begin{pmatrix} X'_B \\ X'_N \end{pmatrix} = b$$

(note that $S_{ij}^{-1} = S_{ij}$).

²The introduction of the requirement that $X_B \in \mathbb{Z}$ does not reduce expressivity as long as the elements of N and b can be expressed as rational numbers. In this case they can be converted to integers by multiplying each row by the product of the denominators on that row and dividing by the greatest common divisor. Once N and b are integer then X_B is guaranteed to be integer for any $X_N \in \{0, 1\}^J$.

However, the matrix $(I| - N)S_{ij}^{-1}$ is no longer in the standard form since S_{ij}^{-1} has the effect of replacing a column in the identity with a column in the $-N$ matrix, leaving the first I columns of $(I| - N)S_{ij}^{-1}$ equal to

$$B_{ij} = (I| - N)S_{ij}^{-1} \begin{pmatrix} I \\ \mathbf{0} \end{pmatrix}$$

which is not, in general, the identity. However, as long as $N_{ij} \neq 0$ then B is invertible and takes the simple form of the identity matrix whose i^{th} column is replaced by a function of the j^{th} column of N

$$B_{ij}^{-1} = \begin{pmatrix} 1 & & -\frac{N_{0j}}{N_{ij}} & & \\ & \ddots & \vdots & & \\ & & \frac{1}{-N_{ij}} & & \\ & & \vdots & \ddots & \\ & & -\frac{N_{nj}}{N_{ij}} & & 1 \end{pmatrix}$$

We can now recover the standard form by pre-multiplying by B^{-1}

$$B_{ij}^{-1}(I| - N)S_{ij}^{-1} = (B_{ij}^{-1}| - B_{ij}^{-1}N)S_{ij}^{-1} = (I| - N') = B_{ij}^{-1}b = b' \quad (16)$$

Since B^{-1} is invertible the transformed constraint is equivalent to the original, so if we let

$$Q' = QS_{ij}^{-1}$$

we can express \mathcal{B} in the form

$$\mathcal{B} = \left\{ X : X = Q' \begin{pmatrix} X'_B \\ X'_N \end{pmatrix}, (I| - N') \begin{pmatrix} X'_B \\ X'_N \end{pmatrix} = b', S_{ij} \begin{pmatrix} L \\ \mathbf{0} \end{pmatrix} \leq \begin{pmatrix} X'_B \\ X'_N \end{pmatrix} \leq S_{ij} \begin{pmatrix} U \\ \mathbf{1} \end{pmatrix}, X \in \{0, 1\}^K, \begin{pmatrix} X'_B \\ X'_N \end{pmatrix} \in \mathbb{I}^{I+J} \right\} \quad (17)$$

This is almost in the standard form, apart from the lower and upper bounds on X'_N which may have an element with bounds other than 0 and 1. However, if we add the constraint that a pivot transformation can only be performed for $i : L_i = 0, U_i = 1$ then S_{ij} makes no change and the transformed representation is standard.

It is easy to see that this transformation can be extended to any permutation matrix S such that $B = (I| - N)S^{-1}(I|\mathbf{0})^T$ is invertible. So, given a reference representation of a \mathcal{B} -polyhedron, $\mathcal{B}(Q, N, b, L, U)$, we can define the set of all equivalent ‘‘pivot states’’ with respect to (N, L, U) to be

$$\mathcal{S}(N, L, U) = \left\{ S : S \text{ is a permutation matrix, } (I| - N)S^{-1} \begin{pmatrix} I \\ \mathbf{0} \end{pmatrix} \text{ is invertible, } S \begin{pmatrix} L|U \\ \mathbf{0}|\mathbf{1} \end{pmatrix} = \begin{pmatrix} L|U \\ \mathbf{0}|\mathbf{1} \end{pmatrix} \right\}$$

so that each pivot state, when applied to the reference state, gives an equivalent standard form representation.

4.3 Removal of fixed variables

If we use the method described in section 3.2 to create a \mathcal{B} -polyhedron of the support then each continuity constraint (and possibly some of the observation constraints) will result in elements of X_B that have fixed values (i.e. $L_i = U_i$). Removing these will reduce the dimensionality of the problem and so make valid solutions easier to find.

Suppose the i^{th} element of X_B is fixed. We can perform a pivot on (i, j) , for some $j : N_{ij} \neq 0$ to swap the fixed variable with the j^{th} element of X_N so that the fixed variable ends up in X_N . This would leave the upper and lower bounds in a non-standard state since X_N must have bounds $\mathbf{0} \leq X_N \leq \mathbf{1}$. However, the fixed variable can be removed from X_N by adding the j^{th} column of N times the fixed variable to left and right sides of the equality constraint.

$$(I| - N) \begin{pmatrix} X_B \\ X_N \end{pmatrix} + N^j X_{Nj} = b + N^j X_{Nj} = (I| - N') \begin{pmatrix} X_B \\ X'_N \end{pmatrix} = b'$$

where N^j denotes the j^{th} column of N , X'_N is X_N with the j^{th} element removed and N' is N with the j^{th} column removed. Since X_{Nj} is fixed, then b' is also fixed and we have recovered the standard form, but now with X'_N having one less dimension than X_N .

With the variable change, we also need to update Q and the upper and lower bounds by pre-multiplying by S_{ij} then removing the j^{th} column (since the fixed variable is not part of X , the j^{th} column in Q will be $\mathbf{0}$ so will not change X , and the j^{th} column in the bounds will be the offending fixed variable bounds, which, when removed will return them to the standard form).

In the following, we'll assume that all fixed variables have been removed from any representation of a \mathcal{B} -polyhedron.

5 Sampling from a \mathcal{B} -distribution

Armed with the ability to express $P(T|\Omega)$ as a \mathcal{B} -distribution and some tools to manipulate representations of \mathcal{B} -polyhedra, we can now go about defining a Markov process which will allow us to sample from a \mathcal{B} -distribution.

To do this we need to define

- a set of Markov states, \mathcal{M}
- a probability measure $P : \mathcal{M} \rightarrow \mathbb{R}$ which gives the probability of each Markov state (this need not be normalised, though, as we'll only ever be interested in probability ratios)
- a stochastic proposal function $f : \mathcal{M} \rightarrow \mathcal{M}$ from which we can generate transitions to a new Markov state given the current Markov state
- a mapping $E : \mathcal{M} \rightarrow \mathbb{R}_{SA}^T$ which maps Markov states to trajectories so we can recover the sample.

In order to be useable in the Metropolis-Hastings algorithm, the proposal function, f , must have the following properties:

- For any two Markov states there should exist a set of transitions with non-zero probability which forms a path between those states.
- For any transition from state $S_a \rightarrow S_b$ with non-zero probability, the probability of the reverse transition from $S_b \rightarrow S_a$ should also be non-zero. This allows us to attain detailed balance in the Metropolis Hastings algorithm. The average ratio of forward and backward probabilities times the ratio of start and end state probabilities should be close to 1 to ensure that a reasonable proportion of proposals are accepted.
- Given a current Markov state, there should be computationally efficient ways of generating a proposal and calculating the ratio the probability of that transition and the reverse transition.

5.1 The Markov states

Given a \mathcal{B} -polyhedron and a reference representation in standard form, $\mathcal{B}(Q, N, b, U, L)$, we define a Markov state to be a pair (X, S) where $X \in \mathcal{B}(Q, N, b, U, L)$ is a member of the \mathcal{B} -polyhedron and S is a pivot state which defines a variable permutation

$$\begin{pmatrix} X'_B \\ X'_N \end{pmatrix} = S \begin{pmatrix} X_B \\ X_N \end{pmatrix}$$

which, when applied to the reference representation gives a transformed representation (Q', N', b', U, L) .

Given a solution X , there is clearly a unique X_N for the representation where all elements of X are in X_N (such a representation must exist as it's the representation we begin with when we first construct the polyhedron from the ABM). So for any transformed representation there is also a unique X'_N which can be found by transforming to a representation with all X in X_N , calculating $(X_B|X_N)$ and transforming back to get $(X'_B|X'_N)$. Clearly the converse also holds, given an X_N in any representation there is a unique X given by

$$X = Q \begin{pmatrix} NX_N + b \\ X_N \end{pmatrix}$$

so we can talk interchangeably of the X or X'_N of the Markov state.

5.2 The probability of a Markov state

Notice that for a given feasible solution, X , there are many Markov states, one for each possible pivot state. However, since the number of pivot states is the same for all feasible states we can assign to each feasible Markov state a probability $\frac{P(X)}{N}$, where $P(X)$ is the probability of solution X and N is the number of valid pivot states. So, the probability of being in a Markov state associated with solution X (summed over all pivot states) is $P(X)$. In the Metropolis-Hastings algorithm we only ever need to calculate probability ratios so, since N is independent of X , we never need to calculate the value of N^3 .

5.3 The transitions between Markov states

5.3.1 Bounds swaps

Given the X_N of a Markov state, a simple transition between Markov states is to swap the i^{th} element of X_N to it's other value while keeping the representation fixed. We'll call this a bound swap. If we're lucky the bound swap will not push X_B outside its bounds and we'll have found another feasible Markov state. However, not all bound swaps are feasible in this way, and worse, there is no guarantee that given two feasible Markov states with the same representation there exists a sequence of feasible bound swaps that forms a path from one to the other.

³which is handy because it would be very difficult to calculate

5.3.2 Pivot transitions

Another simple transition between Markov states is to apply the pivot transition S_{ij} to the representation for some valid i and j . Since the variable that moves from X_B to X_N must end up having a value of either 0 or 1, we also specify which state this variable should end up in after the pivot. So, a pivot transition is fully defined by a triplet (i, j, b) .

5.3.3 Infeasibility, infeasibility objective and potential energy

Unfortunately, if we only allow bound swaps and pivot transitions between feasible states there is no guarantee that there is a path between any two feasible states (note that we can't pivot on the non-binary basic variables or on elements of N that don't have an absolute value of 1, so we can't guarantee that all vertices of the polyhedron are reachable and so can't rely on the proof that the polyhedron is connected).

In order to deal with this, we relax the bounds on the basic variables, $L \leq X_B \leq U$, and include Markov states where some elements of X_B go outside their bounds. This ensures a path between any two feasible states (for example, by just performing bound swaps on the elements of X_N) without affecting the definition of bound swaps and pivot transitions.

With this modification the sampling algorithm will sometimes return infeasible samples which don't represent valid posterior ABM trajectories. However, if we just throw these away the remaining feasible samples will have the correct distribution. So, our aim will be to ensure that the infeasible/feasible sample ratio is small enough not to excessively slow down the sampling process while being large enough to ensure proper mixing in the Markov process.

5.4 Choosing a transition

Now that we've defined the Markov states and the transitions between them, we next provide an algorithm to choose a proposal transition given the current Markov state. The algorithm should be able to make the choice, and work out the ratio of the probability of choosing that transition and the probability of choosing the reverse transition from the destination state.

The challenge here is to ensure that, during the sampling process, feasible states sometimes transition to infeasible states (in order to ensure good mixing) while ensuring that, once in an infeasible state, the sampling process quickly moves back to a (probably different) feasible state to create the next feasible sample.

In order to do this we take as our inspiration the algorithm described in (Maros, 1986). This algorithm is intended to be used as a "phase 1" of the simplex algorithm (see e.g. (Vanderbei, 2020)) where the aim is just to find a feasible state as efficiently as possible rather than to create a Markov process. Here we describe a Markov process that has a high probability of making the same transition as Maros' algorithm. Given this, we should expect the Markov process to quickly move into a feasible state.

5.4.1 Choosing a non-basic variable: Infeasibility and potential energy

The first step in the algorithm to propose a transition is to choose a non-basic variable to modify. To understand how this is done we'll first introduce a few concepts.

Given a Markov state, μ , let the infeasibility of the i^{th} basic variable, ι_i^μ , be defined to be equal to 0 if the variable is within its bounds or equal to the distance to the nearest bound otherwise

$$\iota_i^\mu = \begin{cases} L_i - X_{Bi} & \text{if } X_{Bi} < L_i \\ X_{Bi} - U_i & \text{if } X_{Bi} > U_i \\ 0 & \text{otherwise} \end{cases}$$

and let the total infeasibility be the sum of infeasibilities of all basic variables

$$\iota^\mu = \sum_i \iota_i^\mu$$

Note that the total infeasibility is piecewise linear, convex, has a minimum of 0 and all feasible points lie at the minimum. So, a good strategy would be to move down the gradient of ι towards zero.

The gradient of ι^μ at any point is given by

$$\frac{d\iota^\mu}{dX_{Bi}} = \begin{cases} -1 & \text{if } X_{Bi} < L_i \\ 1 & \text{if } X_{Bi} > U_i \\ 0 & \text{otherwise} \end{cases}$$

Since $X_B = NX_N + b$, the gradient of ι^μ with respect to X_N , which we'll call it D^μ for convenience, is given by

$$D^\mu = \frac{d\iota^\mu}{dX_N} = \frac{d\iota^\mu}{dX_B} \frac{dX_B}{dX_N} = \frac{d\iota^\mu}{dX_B} N$$

So, given D^μ , we can tell which direction each non-basic variable, X_{Nj} , needs to be perturbed in order to potentially reduce infeasibility. More precisely, D_j^μ gives the infeasibility gradient with respect to X_{Nj} at the current value of X_N , but any perturbation may cross a piecewise linear boundary so the gradient may change as X_N changes. However, because of the convexity of ι we know that heading up the gradient will definitely increase infeasibility, for any size of perturbation, whereas heading down the gradient will reduce infeasibility unless we're already at a minimum with respect to X_{Nj} (note that in a piecewise linear function the minimum may lie at the junction between linear segments that have non-zero gradient). So, variables for which a bound-swap would move down the gradient are good potential candidates for perturbation compared to those that don't.

Given this, we define the potential energy of the j^{th} non-basic variable to be equal to $|D_j^\mu|$ if a bound-swap would move down the gradient, and 0 otherwise. This can be expressed mathematically as

$$E_j^\mu = D_j^\mu (X_{Nj} - H(-D_j^\mu))$$

where H is the Heaviside step function.

In order to encourage the selection of columns with positive potential, the j^{th} non-basic variable is chosen with probability proportional to the exponential of its potential energy times some constant k_c

$$P(j|\mu) = \frac{e^{k_c E_j^\mu}}{\sum_l e^{k_c E_l^\mu}}$$

5.4.2 Choosing a column perturbation and pivot row

Once a non-basic variable, X_{Nj} , is chosen for the transition, we can choose to perform a bound-swap on X_{Nj} or perform a pivot transition on the j^{th} column and any row that satisfies $|N_{ij}| = 1$. If a pivot is chosen, the value of the new non-basic variable needs to be chosen to be either 0 or 1. An efficient algorithm to calculate the infeasibility resulting from each of these transitions is described in (Maros, 2002, Chapter 9). For reasons that will become clear later, we also include the "null transition" which is a loop back to the same Markov state.

Let α_j^μ be the set of destination states reachable through transitions on X_{Nj} . In order to encourage transitions to lower infeasibility, while maintaining the possibility of reverse transitions to allow detailed balance, a transition, $\mu \rightarrow \mu'$, is chosen with a probability proportional to the exponential of $-k_r \iota^{\mu'}$ for some constant k_r . i.e.

$$P(\mu' | j, \mu) = \frac{e^{-k_r \iota^{\mu'}}}{\sum_{\rho \in \alpha_j^\mu} e^{-k_r \iota^\rho}} \quad (18)$$

5.4.3 Proposal function summary

If we transition from state μ to state $\mu' \neq \mu$ it is easy to identify which non-basic variable X_{Nj} was involved in the transition ⁴ we'll use the notation $j_{\mu \rightarrow \mu'}$ to denote the index of this variable.

Starting from state μ , the probability of transitioning to μ' is the probability that we choose the non-basic variable, $X_{Nj_{\mu \rightarrow \mu'}}$ then choose the state μ' from $\alpha_{j_{\mu \rightarrow \mu'}}^\mu$

$$P(\mu \rightarrow \mu') = P(\mu' | j_{\mu \rightarrow \mu'}, \mu) P(j_{\mu \rightarrow \mu'} | \mu) = \frac{e^{-k_r \iota^{\mu'}}}{\sum_{\rho \in \alpha_{j_{\mu \rightarrow \mu'}}^\mu} e^{-k_r \iota^\rho}} \frac{e^{k_c E_{j_{\mu \rightarrow \mu'}}^\mu}}{\sum_l e^{k_c E_l^\mu}}$$

So, in the Metropolis-Hastings algorithm, the probability that a proposed transition, $\mu \rightarrow \mu'$, is accepted is given by

$$\frac{P(\mu') P(\mu' \rightarrow \mu)}{P(\mu) P(\mu \rightarrow \mu')} = \frac{P(\mu')}{P(\mu)} \frac{e^{-k_r \iota^\mu}}{\sum_{\rho \in \alpha_{j_{\mu' \rightarrow \mu}}^{\mu'}} e^{-k_r \iota^\rho}} \frac{\sum_{\rho \in \alpha_{j_{\mu \rightarrow \mu'}}^\mu} e^{-k_r \iota^\rho}}{e^{-k_r \iota^{\mu'}}} \frac{e^{k_c E_{j_{\mu' \rightarrow \mu}}^{\mu'}}}{\sum_l e^{k_c E_l^{\mu'}}} \frac{\sum_l e^{k_c E_l^\mu}}{e^{k_c E_{j_{\mu \rightarrow \mu'}}^\mu}}$$

However, thanks to the fact that we included the null transition, for any transition $\mu \rightarrow \mu'$, $\alpha_{j_{\mu \rightarrow \mu'}}^{\mu'} = \alpha_{j_{\mu' \rightarrow \mu}}^\mu$ ⁵ so the two sums over α cancel, leaving

$$\frac{P(\mu') P(\mu' \rightarrow \mu)}{P(\mu) P(\mu \rightarrow \mu')} = \frac{P(\mu')}{P(\mu)} e^{k_r (\iota^{\mu'} - \iota^\mu)} \frac{e^{k_c E_{j_{\mu' \rightarrow \mu}}^{\mu'}}}{\sum_l e^{k_c E_l^{\mu'}}} \frac{\sum_l e^{k_c E_l^\mu}}{e^{k_c E_{j_{\mu \rightarrow \mu'}}^\mu}} \quad (19)$$

⁴if N is unchanged then the transition was a bound swap on the only non-basic that has changed, if N changes to N' then it was a pivot transition on the only j that satisfies equation 16

⁵since $\alpha_{j_{\mu \rightarrow \mu'}}^\mu$ describes the set of states that lie on the straight line that passes through μ and μ'

However, since infeasible states are thrown away in the final algorithm, we are free to assign them any probability we wish. This gives us some freedom to ensure that the algorithm spends an appropriate time in infeasible states and allows us to engineer an acceptance probability that is convenient to calculate. We choose

$$P(\mu) = \frac{P_e(T_\mu|\Omega)}{n} e^{-k_r \iota^\mu} e^{-k_c \sum_j E_j^\mu} \sum_j e^{k_c E_j^\mu} \quad (20)$$

where n is the number of non-basic variables, T_μ is the trajectory associated with Markov state μ and $P_e(T_\mu|\Omega)$ is an “extended posterior”, which for feasible states is just the posterior given by equation 6, while for infeasible states is calculated in the same way but the requirement that the trajectory belongs to $\mathcal{T}_{SA}^N(F)$ is dropped and if any of the individual probabilities is zero it is replaced by its expectation over all trajectories (with some convenient prior over trajectories, we used a uniform distribution). Notice that for feasible states $P(\mu) = P(T_\mu|\Omega)$ so if we apply this definition to all Markov states the feasible samples will be samples from the desired posterior.

Notice that this gives higher probability to states with lower infeasibility and lower potential energy. So transitions that immediately reduce infeasibility are encouraged, but so are pivots that move non-basic variables with high potential energy into the basic variables. This means that even if a high potential variable can’t immediately reduce infeasibility, it is still likely to be made basic, where it has the ability to change its value in a subsequent transition.

Inserting 20 into 19 gives

$$\frac{P(\mu')P(\mu' \rightarrow \mu)}{P(\mu)P(\mu \rightarrow \mu')} = \frac{P_e(T_{\mu'}|\Omega)}{P_e(T_\mu|\Omega)} e^{-k_c \Delta E_{\mu \rightarrow \mu'}}$$

where

$$\Delta E_{\mu \rightarrow \mu'} = \sum_{j \neq j_{\mu \rightarrow \mu'}} E_j^{\mu'} - E_j^\mu$$

is the change in total potential energy in the transition $\mu \rightarrow \mu'$, ignoring the $j_{\mu \rightarrow \mu'}^{th}$ column.

6 Results

7 Further work

7.1 Abstract interpretation using convex polyhedra

In many cases a \mathcal{B} -polyhedron of the set of Fermionic trajectories in the support of the prior, observations and timestep of an agent can easily be constructed by hand in the form of a set of linear inequalities. In some cases, however, it may be less obvious how to construct this. In this case the linear inequalities can be constructed automatically from a computer program that calculates the function whose support we’re trying to find.

The first two terms in equation 7 consists of the supports of computer programs whose inputs are ABM trajectories and whose outputs are given, i.e. the set of trajectories that, when passed to a computer program, would produce a given output.

Calculating the support of a computer program for a given output is, in full generality, NP-complete⁶ but it is possible to use a technique known as *abstract interpretation* (Cousot & Cousot, 1977) to efficiently calculate a superset of the support. So, given a computer program ρ , we can calculate a set $\mathcal{P}(\rho, v)$ such that

$$\text{supp}(P(\rho(\cdot) = y)) \subset \mathcal{P}(\rho, v)$$

Tools to perform abstract interpretation already exist (e.g. PAGAI (Henry, Monniaux, & Moy, 2012)) and are used widely in applications such as the verification of safety critical systems (Blanchet et al., 2003) and in practice $\mathcal{P}(\rho, v)$ is often reasonably tight (i.e. most members of $\mathcal{P}(\rho, v)$ are in $\text{supp}(P(\rho(\cdot) = y))$). For our application we choose to express $\mathcal{P}(\rho, v)$ in terms of a set of linear inequalities on ρ ’s inputs, this corresponds to the abstract domain of convex polyhedra (Cousot & Halbwachs, 1978) (Becchi & Zaffanella, 2018). Calls to the random number generator can be dealt with in the abstract domain by generating a new variable, r , that satisfies $0 \leq r < 1$ for each call to `Random()`. These can either be left in as “auxiliary” variables in the same way as slack variables, or removed as soon as the variable goes out of scope by finding the convex hull of the projection into a lower dimensional space (this can be done using the double description method (Motzkin, Raiffa, Thompson, & Thrall, 1953)).

Constraining our proposal function to members of the superset in equation ?? instead of the true support won’t affect the stationary distribution of the Markov Chain. If the proposal function happens to return a trajectory that isn’t in $\text{supp}(P(\rho(\cdot) = y))$ then it will just be rejected. This is fine as long as we generate acceptable proposals at a reasonable rate.

⁶Consider, for example, a program that accepts an assignment of variables to truth values, and returns true if that assignment satisfies a Boolean formula. Deciding whether the support of this program, given that it returns true, is empty or not is equivalent to solving the Boolean satisfiability problem, which is known to be NP-complete (Cook, 1971)

8 Conclusion

References

- Becchi, A., & Zaffanella, E. (2018). An efficient abstract domain for not necessarily closed polyhedra. In *International static analysis symposium* (pp. 146–165).
- Bertino, L., Evensen, G., & Wackernagel, H. (2003, August). Sequential Data Assimilation Techniques in Oceanography. *International Statistical Review*, 71(2), 223–241. doi: 10.1111/j.1751-5823.2003.tb00194.x
- Blanchet, B., Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., . . . Rival, X. (2003). A static analyzer for large safety-critical software. In *Proceedings of the acm sigplan 2003 conference on programming language design and implementation* (pp. 196–207).
- Clay, R., Kieu, L.-M., Ward, J. A., Heppenstall, A., & Malleson, N. (2020). Towards Real-Time Crowd Simulation Under Uncertainty Using an Agent-Based Model and an Unscented Kalman Filter. In Y. Demazeau, T. Holvoet, J. M. Corchado, & S. Costantini (Eds.), *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection* (Vol. 12092, pp. 68–79). Cham: Springer International Publishing. doi: 10.1007/978-3-030-49778-1_6
- Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual acm symposium on theory of computing* (pp. 151–158).
- Cousot, P., & Cousot, R. (1977). Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th acm sigact-sigplan symposium on principles of programming languages* (pp. 238–252).
- Cousot, P., & Halbwachs, N. (1978). Automatic discovery of linear restraints among variables of a program. In *Proceedings of the 5th acm sigact-sigplan symposium on principles of programming languages* (pp. 84–96).
- Dantzig, G. B., Orden, A., Wolfe, P., et al. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2), 183–195.
- Henry, J., Monniaux, D., & Moy, M. (2012). Pagai: A path sensitive static analyser. *Electronic Notes in Theoretical Computer Science*, 289, 15–25.
- Kalnay, E. (2003). *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press.
- Kieu, L.-M., Malleson, N., & Heppenstall, A. (2020). Dealing with uncertainty in agent-based models for short-term predictions. *Royal Society Open Science*, 7(1), 191074. doi: 10.1098/rsos.191074
- Lewis, J. M., Lakshmivarahan, S., & Dhall, S. (2006). *Dynamic Data Assimilation: A Least Squares Approach*. Cambridge: Cambridge University Press.
- Lloyd, D. J. B., Santitissadeekorn, N., & Short, M. B. (2016). Exploring data assimilation and forecasting issues for an urban crime model. *European Journal of Applied Mathematics*, 27(Special Issue 03), 451–478. doi: 10.1017/S0956792515000625
- Lueck, J., Rife, J. H., Swarup, S., & Uddin, N. (2019). Who Goes There? Using an Agent-based Simulation for Tracking Population Movement. In *Winter Simulation Conference, Dec 8 - 11, 2019*. National Harbor, MD, USA.
- Malleson, N., Minors, K., Kieu, L.-M., Ward, J. A., West, A., & Heppenstall, A. (2020). Simulating crowds in real time with agent-based modelling and a particle filter. *Journal of Artificial Societies and Social Simulation*, 23(3), 3. doi: 10.18564/jasss.4266
- Maros, I. (1986). A general phase-i method in linear programming. *European Journal of Operational Research*, 23(1), 64–77.
- Maros, I. (2002). *Computational techniques of the simplex method* (Vol. 61). Springer Science & Business Media.
- Motzkin, T. S., Raiffa, H., Thompson, G. L., & Thrall, R. M. (1953). The double description method. *Contributions to the Theory of Games*, 2(28), 51–73.
- Reichle, R. H. (2008, November). Data assimilation methods in the Earth sciences. *Advances in Water Resources*, 31(11), 1411–1418. doi: 10.1016/j.advwatres.2008.01.001
- Vanderbei, R. J. (2020). *Linear programming* (Vol. 5). Springer.
- Wang, M., & Hu, X. (2015). Data assimilation in agent based simulation of smart environments using particle filters. *Simulation Modelling Practice and Theory*, 56, 36–54. doi: 10.1016/j.simpat.2015.05.001
- Ward, J. A., Evans, A. J., & Malleson, N. S. (2016). Dynamic calibration of agent-based models using data assimilation. *Royal Society Open Science*, 3(4). doi: 10.1098/rsos.150703