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

Nick Malleson
School of Geography, University of Leeds, UK

December 14, 2021

ABSTRACT

The discipline of data assimilation (DA), sometimes known as data fusion, addresses the problem of how to make use of noisy, imcomplete experimental observations to provide information about unobserved properties of a dynamical system. DA has developed rapidly in some areas of research, notably weather forecasting, but relatively little progress has been made in DA techniques applicable to agent based modelling. Agent Based Models (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.

***K*eywords** Data assimilation, Bayesian inference, Agent based model, Integer linear programming, predator prey model

1 Introduction

Agent-based modelling (ABM) has become a popular tool for use in modelling the interactions of many discrete, heterogeneous entities, such as humans or animals. In such applications we often have noisy, aggregated and/or incomplete observations of the agents over some period of time and would like to use our ABM to model what happened to the agents during this time period, given our observations. However, this is not straightforward as we don't know the complete start-state of the model, or the exact behavioural decisions the agents would need to make in order to reproduce our observations, so simply running the model forward is not an option. The problem is better framed in terms of probabilistic inference: given the model, our prior beliefs about the start state and our observations, what is the probability distribution over all possible histories over the observed period? This is the problem of Data assimilation (DA) and the subject of this paper. Note that this problem is distinct from the problem of model estimation or parameter optimisation (Thiele, Kurth, & Grimm, 2014) which we do not address here.

This paper presents a new method that will allow for *data assimilation* (DA) to be applied to ABMs in order to make inference about the evolution of the agents over an observed time priod. DA has developed rapidly in applications such as weather forecasting (Kalnay, 2003) and the earth sciences more broadly (Reichle, 2008), but little progress has

*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)

been made in developing techniques that are applicable when the dynamical system is an agent based model. Many DA methods rely on the differentiability of the posterior distribution (Lewis, Lakshmivarahan, & Dhall, 2006), but since ABMs consist of 'agents' that often make discrete choices from a number of possible actions, the space of model trajectories is not continuous and techniques based on gradient ascent, such as 4D-VAR (Talagrand, 1997) cannot be used. In addition, the majority of model trajectories will be refuted by the observations, making it challenging to use algorithms that rely on perturbing the current solution such as non-gradient optimisation or most sampling algorithms. Some studies have tried to apply well-known DA methods such as Particle Filters and variants of the Kalman Filter to ABMs in applications such as 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), but these have not scaled well and the lack of DA techniques for ABM remains a serious methodological limitation. Other existing techniques for sampling from discrete probability distributions include discrete hit-and-run (Baumert et al., 2009) but because of the extreme sparsity of feasible points in the space of ABM trajectories this is unlikely to be efficient. Universal hashing (Meel et al., 2016) provides a promising alternative, but in our experience this doesn't scale to the number of dimensions needed for our application.

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 adapts techniques used in linear programming to provide a proposal function for Metropolis-Hastings sampling. We demonstrate the algorithm by performing data assimilation in an agent-based, spatial predator-prey model.

2 Formulation of the problem

2.1 Agents, States and Actions

Suppose we have a timestepping ABM that consists of agents with 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:

- An ordered list of agent actions $\mathcal{A} = \langle A_0 ... A_n \rangle$
- An ordered list of agent states $\mathcal{S} = \langle S_0 ... S_m \rangle$
- An *agent timestep*, $\pi : \mathbb{Z} \times \mathbb{Z}^{m+1} \times \mathbb{Z} \to \mathbb{R}$, 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*, $F : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^{m+1}$, 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, we define the "cat and mouse" ABM which, using the definitions above, we define as follows:

Agent actions, $\mathcal{A}$. In any given timestep, an agent can either move or stay still, so

$$\mathcal{A} = \langle \text{move, stay still} \rangle$$

.

Agent states, $\mathcal{S}$. An agent can be either a cat or a mouse and can be on one of two gridsquares, left or right, so

$$\mathcal{S} = \langle \text{left cat, right cat, left mouse, right mouse} \rangle$$

.

Agent timestep, π. The agent timestep can be expressed as

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

The top line says that a cat will move or stay still with probability 0.5, irrespective of other agents. The next line says that a mouse will stay still if there are no cats on the same gridsquare while the third line says that a mouse will move if there are any cats on the same gridsquare. Finally, the last line says that any other behaviours have zero probability.

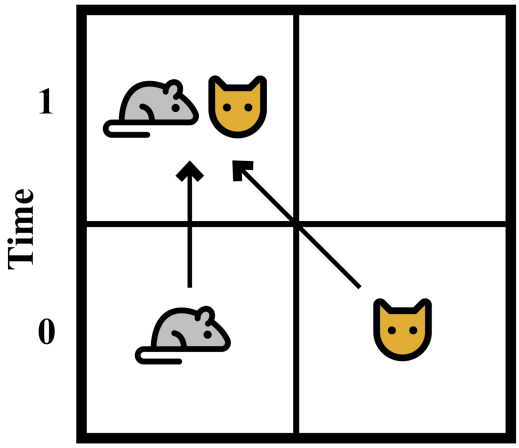

Figure 1: A simple cat and mouse model.

Action function, F. Expresses the movement of the agents. For example, $F(\psi = 1, a = 0)$ states that if there is an agent in state $\psi = 1$ (*right cat*) and it performs the action $a = 0$ (*move)* then the result is one cat in state $\psi = 0$ (*left cat*) which is expressed as the vector $\{1, 0, 0, 0\}$. Similarly:

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

2.2 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

$$
E = \begin{array}{c} \\ S_0 \\ S_1 \\ S_2 \\ S_3 \end{array}
\begin{array}{c} \begin{array}{cc} A_0 & A_1 \end{array} \\
\left[\begin{array}{cc} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{array}\right] \end{array}
$$

where all elements are zero except those representing agent S_1 (*right cat*) performing action A_0 (*move*) and agent S_2 (*left mouse*) performing action A_1 (*stay still*).

Finally, let a model trajectory, T, be an $(m \times n \times t)$ tensor consisting of t model timesteps. We use the notation T^t to denote the t^{th} timestep matrix, T^t_ψ to denote the ψ^{th} row of the t^{th} timestep matrix and $T^t_{\psi a}$ to denote the a^{th} element of the ψ^{th} row of the t^{th} timestep. By convention, 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 $\mathbb{R}^N_{SA}$.

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 of agents, they must be non-negative integers. We'll call this the *non-negative integer constraint* and define the set of all non-negative integer tensors

$$
\mathcal{I}^N_{SA} = \left\{ T \in \mathbb{R}^{\mathbb{N}}_{\mathbb{SA}} : \forall t, \psi, a : T^t_{\psi a} \geq 0, T^t_{\psi a} \in \mathbb{Z} \right\} \tag{1}
$$

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, with respect to an action function F:

$$\mathcal{C}_{SA}^{N}(F) = \left\{ T \in \mathbb{R}_{SA}^{N} : \forall t \in 1...N-1 : \forall \phi : \sum_{\psi,a} F(\psi,a)_{\phi} T_{\psi a}^{t-1} - \sum_{a} T_{\phi a}^{t} = 0 \right\} \tag{2}$$

So, the set of valid trajectories, $\mathcal{T}_{SA}^{N}(F)$, is given by the set of tensors that satisfy (1) and (2).

$$\mathcal{T}_{SA}^{N}(F) = \mathcal{I}_{SA}^{N} \cap \mathcal{C}_{SA}^{N}(F) \tag{3}$$

2.3 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

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

where $P(\Psi^0)$ is our prior belief about the model 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

$$P\left(T_{\psi}^{t} \mid \Psi^t\right) = \Psi_{\psi}^{t}! \prod_{a} \frac{\pi(\psi, \Psi^t, a)^{T_{\psi a}^{t}}}{T_{\psi a}^{t}!}.$$

So the prior probability of a trajectory is

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

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

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

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

$$P(T|\Omega) \propto \begin{cases} P(\Psi^0 = T^0\mathbf{1}) \prod_{(\omega,v)\in\Omega} P\left(\omega(T) = v\right) \prod_{t,\psi,a} \left(T_{\psi}^{t} \cdot \mathbf{1}\right)! \frac{\pi(\psi, T^t\mathbf{1}, a)^{T_{\psi a}^{t}}}{T_{\psi a}^{t}!} & \text{if } T \in \mathcal{T}_{SA}^{N}(F) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

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 the rest of this paper we'll show how to use the Metropolis-Hastings algorithm to generate samples from equation (4). The challenge will be to create a proposal function which randomly generates a proposed next sample given the current one. 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, $\text{supp}(P(T^t_{\psi a}|\Omega))$ (i.e. the set of trajectories that have non-zero probability).

From equation (4)

$$\begin{aligned}\text{supp}(P(T|\Omega)) =& \mathcal{T}^N_{SA} \cap \\ & \text{supp}(P(\Psi^0 = T^0\mathbf{1})) \cap \\ & \bigcap_{(\omega,v)\in\Omega} \text{supp}\left(P\left(\omega(T) = v\right)\right) \cap \\ & \bigcap_{t,\psi,a} \left(\text{supp}\left(\pi(\psi, T^t\mathbf{1}, a)\right) \cup \left\{T : T^t_{\psi a} = 0\right\}\right)\end{aligned} \tag{5}$$

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 denote an agent action with non-zero probability.

3.1 Convex $\mathbb{Z}$-polyhedra and $\mathbb{Z}$-distributions

Let a $\mathbb{Z}$-polyhedron be a set of tensors with integer elements that can be described as a set of linear constraints on the elements:

$$\mathcal{P}^{\mathcal{N}}_{\mathcal{SA}} = \left\{ T \in \mathbb{Z}^N_{SA} : L \leq \sum_{t,\psi,a} C^{\psi a}_t T^t_{\psi a} \leq U \right\}$$

where L, U and $C^{\psi a}_t$ are vectors (this is similar to the $\mathbb{Z}$-polyhedron described in (Quinton, Rajopadhye, & Risset, 1996)).

From equation 3 we can see immediately that $\mathcal{T}^N_{SA}$ is a $\mathbb{Z}$-polyhedron. The supports of the prior, $P(\Psi^0)$, the observations, $P(\omega(T) = v)$, and the agent actions, $\pi(\psi, T^t\mathbf{1}, a)$, can often be easily expressed as $\mathbb{Z}$-polyhedra. If this is not the case, each of the probability distributions can be expressed as computer programs. Once in this form, abstract interpretation tools(Cousot & Cousot, 1977) using the domain of convex polyhedra (Cousot & Halbwachs, 1978)(Becchi & Zaffanella, 2018)(Fukuda, 2020) can be used to construct a convex polyhedron that contains the support (note that it's fine to overestimate the support, i.e. include points that aren't in the support, but not to exclude points that are in the support. Abstract interpretation tools(Henry, Monniaux, & Moy, 2012)(Gurfinkel & Navas, 2021) are perfectly suited to this purpose). In addition, if the number of agents is very much smaller than the number of agent states (which is often the case with agent based models) then we may be willing to make the assumption that at any timestep there is at most one agent performing a given action from a given start state (i.e. $\forall \psi, a, t : T^{\psi a}_t \in \{0, 1\}$). Under this assumption, which we'll call the *Fermionic assumption* , the set of trajectories is a subset of the corners of the unit hypercube. Any such subset is a $\mathbb{Z}$-polyhedron.

If we write the $\mathbb{Z}$-polyhedral over-approximation of the support of a function, f as $\mathcal{P}^N_{SA}(f) \supseteq \text{supp}(f)$ then we can rewrite equation (5) in the form

$$\begin{aligned}\text{supp}(P(T|\Omega)) \subseteq& \mathcal{T}^N_{SA} \cap \\ & \mathcal{P}^N_{SA}(P(\Psi^0 = T^0\mathbf{1})) \cap \\ & \bigcap_{(\omega,v)\in\Omega} \mathcal{P}^N_{SA}\left(P\left(\omega(T) = v\right)\right) \cap \\ & \bigcap_{t,\psi,a} \left(\mathcal{P}^N_{SA}\left(\pi(\psi, T^t\mathbf{1}, a)\right) \cup \left\{T : T^t_{\psi a} = 0\right\}\right)\end{aligned} \tag{6}$$

The intersection of two $\mathcal{Z}$-polyhedra is easy to construct by just concatenating the constraints

$$\begin{aligned}&\left\{ T \in \mathbb{Z}^N_{SA} : L \leq \sum_{t,\psi,a} C^{\psi a}_t T^t_{\psi a} \leq U \right\} \cap \left\{ T \in \mathbb{Z}^N_{SA} : L' \leq \sum_{t,\psi,a} D^{\psi a}_t T^t_{\psi a} \leq U' \right\} \\ &\qquad = \left\{ T \in \mathbb{Z}^N_{SA} : \begin{pmatrix} L \\ L' \end{pmatrix} \leq \sum_{t,\psi,a} \begin{pmatrix} C^{\psi a}_t \\ D^{\psi a}_t \end{pmatrix} T^t_{\psi a} \leq \begin{pmatrix} U \\ U' \end{pmatrix} \right\}\end{aligned} \tag{7}$$

so the only difficulty is the union in the final term of (6). To transform this into an intersection we introduce an auxiliary variable $I^t_{\psi a}$ and use the identity

$$\left\{T \in \mathbb{Z}^N_{SA} : L \leq \sum_{s,\phi,b} C^{\phi b}_s T^s_{\phi b} \leq U\right\} \cup \left\{T : T^t_{\psi a} = 0\right\} =$$
$$\left\{T \in \mathbb{Z}^N_{SA}, I^t_{\psi a} \in \{0,1\} :\right.$$
$$-\infty \leq \sum_{s,\phi,b} C^{\phi b}_s T^s_{\phi b} + (\overline{B} - U) I^t_{\psi a} \leq \overline{B},$$
$$\underline{B} \leq \sum_{s,\phi,b} C^{\phi b}_s T^s_{\phi b} + (\underline{B} - L) I^t_{\psi a} \leq \infty,$$
$$0 \leq m I^t_{\psi a} - T^t_{\psi a},$$
$$\left. I^t_{\psi a} - T^t_{\psi a} \leq 0\right\} \quad (8)$$

where m is the maximum value that any element of T can take, the elements of $\overline{B} \in \mathbb{R}_I$ are defined as

$$\overline{B}_i = \frac{m \sum_{s,\phi,b} \left(C^{\phi b}_{si} + \left|C^{\phi b}_{si}\right|\right)}{2}$$

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

$$\underline{B}_i = \frac{m \sum_{s,\phi,b} \left(C^{\phi b}_{si} - \left|C^{\phi b}_{si}\right|\right)}{2}$$

To see why this identity holds, note first that the constraints on $I^t_{\psi a}$ make it into an indicator variable that is 0 if $T^t_{\psi a} = 0$ or 1 otherwise. When $I^t_{\psi a} = 1$ the first set of constraints is equal to $\sum_{s,\phi,b} C^{\phi b}_s T^s_{\phi b} \leq U$ and the second is equal to $L \leq \sum_{s,\phi,b} C^{\phi b}_s T^s_{\phi b}$ so their intersection is $L \leq \sum_{s,\phi,b} C^{\phi b}_s T^s_{\phi b} \leq U$ as required, whereas when $I^t_{\psi a} = 0$ we have the constraints $\underline{B} \leq \sum_{s,\phi,b} C^{\phi b}_s T^s_{\phi b} \leq \overline{B}$. But $\underline{B}$ and $\overline{B}$ are lower and upper bounds on the value of $\sum_{s,\phi,b} C^{\phi b}_{si} T^s_{\phi b}$ so this is satisfied for all trajectories, as required.

There are two things worth noting here. Firstly if we make the Fermionic assumption then $I^t_{\psi a} = T^t_{\psi a}$ and the auxiliary indicator variable becomes unnecessary. Secondly, we must impose a finite value for m, the maximum value that elements of the trajectory can take. In practice, this is not a problem as we can give m a value such that the probability of any trajectory of interest having any element larger than m is vanishingly small.

Using this transformation the support of the posterior can be reduced to a $\mathbb{Z}$-polyhedron

The idea of a $\mathbb{Z}$-polyhedron as the support for a probability distribution naturally leads to the idea of a $\mathbb{Z}$-distribution which is a discrete probability distribution defined over the members of a $\mathbb{Z}$-polyhedron. From the above, it can be seen that the posterior distribution of an ABM trajectory is a $\mathbb{Z}$-distribution.

As an illustration, consider a two-timestep trajectory 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 $\mathbb{Z}$-polyhedron, $\mathcal{P}^2_{4\,2}(P(T|\Omega))$, that describes the support of the posterior.

Working through (6) term by term, the $\mathcal{T}^2_{4\,2}$ term is just the continuity constraints in (2), 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

$$\bigcap_{\psi} \left\{T : T^0_{\psi 0} + T^0_{\psi 1} \leq 1\right\}$$

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

$$T^1_{00} + T^1_{01} = 1$$

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

$$\begin{aligned}
\mathrm{supp}(\pi(2, T^t\mathbf{1}, 0)) &= \left\{T : -T^t_{00} - T^t_{01} \leq -1\right\} \\
\mathrm{supp}(\pi(3, T^t\mathbf{1}, 0)) &= \left\{T : -T^t_{10} - T^t_{11} \leq -1\right\} \\
\mathrm{supp}(\pi(2, T^t\mathbf{1}, 1)) &= \left\{T : T^t_{00} + T^t_{01} \leq 0\right\} \\
\mathrm{supp}(\pi(3, T^t\mathbf{1}, 1)) &= \left\{T : T^t_{10} + T^t_{11} \leq 0\right\}
\end{aligned} \quad (9)$$

If, for simplicity, we make the Fermionic assumption by adding the constraints

$$\forall t, \psi, a : T^t_{\psi a} \leq 1$$

then using the identity in (8) to take the union of each constraint in (9) with $\left\{T : T^t_{\psi a} = 0\right\}$ gives the four constraints

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

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

Taken together, these constraints define a $\mathbb{Z}$-polyhedron that is the set of (Fermionic) trajectories for the cat and mouse ABM, and when combined with equation (4) defines $P(T|\Omega)$ as a $\mathbb{Z}$-distribution.

4 Transforming between representations of a $\mathbb{Z}$-polyhedron

TODO: Make this specific for Markov state: selection of J variables (permutation matrix Q) with their values at feasible levels (X_N). Holding J-1 feasible variables fixed defines a perturbation direction. Points where at least one additional variable is at feasible, integer level defines perturbation distance.

Now that we can express the support of the posterior as a $\mathbb{Z}$-polyhedron in the form

$$\text{supp}(P(T|\Omega) \subseteq \left\{ T \in \mathbb{Z}^N_{SA} : L \leq \sum_{t,\psi,a} C^{\psi a}_t T^t_{\psi a} \leq U \right\} \tag{10}$$

we now describe some different ways of representing the same $\mathbb{Z}$-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 $\mathbb{Z}$-polyhedron

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

$$\mathcal{P}(Q, N, b, L, U) = \left\{ X \in \mathbb{Z}^K : X = Q\begin{pmatrix} X_B \\ X_N \end{pmatrix}, X_B = b - NX_N, L \leq \begin{pmatrix} X_B \\ X_N \end{pmatrix} \leq U, X_B \in \mathbb{Z}^I, X_N \in \mathbb{Z}^J \right\} \tag{11}$$

where 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 (10) can easily be expressed in the form (11) by "flattening" the trajectory, T, into a J-dimensional vector (without changing the values of the elements) using a tensor $R \in \mathbb{R}^{SA}_{NJ}$ where $J = NSA$ (i.e. for every (t, ψ, a) there is exactly one j such that $R^{\psi a}_{tj} = 1$ and for every j there is exactly one (t, ψ, a) such that $R^{\psi a}_{tj} = 1$ and all other elements are zero) so that

$$X_N = \sum_{t,\psi,a} R^{\psi a}_t T^t_{\psi a}$$

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

$$N = \sum_{t,\psi,a} C^{\psi a}_t R^{\psi a}_t$$

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.

[2]The 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 $X_N \in \{0, 1\}^J$.

4.2 The pivot transformation

The introduction of the matrix Q in (11) allows us to transform the representation without changing the $\mathbb{Z}$-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 expressing the linear constraint in the form

$$(I|N)\begin{pmatrix} X_B \\ X_N \end{pmatrix} = b$$

where I is the identity matrix.

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 have

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

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

If we now pre-multiply this with the matrix

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

it can be shown (Maros, 2002) that

$$B_{ij}^{-1}(I|N)S_{ij}^{-1} = (B_{ij}^{-1}|B_{ij}^{-1}N)S_{ij}^{-1} = (I|N') \tag{12}$$

and so

$$B_{ij}^{-1}(I|N)S_{ij}^{-1}\begin{pmatrix} X'_B \\ X'_N \end{pmatrix} = B_{ij}^{-1}b = (I|N')\begin{pmatrix} X'_B \\ X'_N \end{pmatrix} = b' \tag{13}$$

Now, if we choose an i and j such that $N_{ij} = \pm 1$, then all elements of B_{ij}^{-1} are integer and so all elements of N' are also integer, as are the elements of b'. This ensures that if $X_N \in \mathbb{Z}^J$ then $X_B \in \mathbb{Z}^I$ and $X \in \mathbb{Z}^K$ as required.

So if we let

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

we can express the original $\mathbb{Z}$-polyhedron in the equivalent form

$$\mathcal{P} = \left\{ X \in \mathbb{Z}^K : X = Q'\begin{pmatrix} X'_B \\ X'_N \end{pmatrix}, X'_B = b' - N'X'_N, S_{ij}L \leq \begin{pmatrix} X'_B \\ X'_N \end{pmatrix} \leq S_{ij}U, X'_B \in \mathbb{Z}^I, X'_N \in \mathbb{Z}^J \right\} \tag{14}$$

It is easy to see that this transformation defines an equivalence set so that, given a reference representation of a $\mathbb{Z}$-polyhedron, $\mathbb{Z}(Q, N, b, L, U)$, we can define the set of all equivalent "pivot states" $\mathcal{S}(Q, N, b, L, U)$.

4.3 Removal of fixed variables

If we use the method described in section (3.1) to create a $\mathbb{Z}$-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. If there exists some $j : N_{ij} = \pm 1$ we can perform a pivot on (i, j), to swap the fixed variable with the j^{th} element of X_N so that the fixed variable ends up in X_N. The fixed variable can be removed from X_N by adding the j^{th} column of N times the variable's fixed value to b.

5 Sampling from a $\mathbb{Z}$-distribution

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

To do this we need to define

- a set of Markov states, $\mathcal{M}$
- a probability measure $P : \mathcal{M} \to \mathbb{R}$ 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 $f : \mathcal{M} \to \mathcal{M}$ from which we can generate transitions to a new Markov state given the current Markov state
- a mapping $E : \mathcal{M} \to \mathbb{R}^T_{SA}$ 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 $S_a \to S_b$ with non-zero probability, the probability of the reverse transition from $S_b \to S_a$ 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 $\mathbb{Z}$-polyhedron, we define a Markov state to be a pair (X, S) where X is a member of the $\mathbb{Z}$-polyhedron and S is a pivot state that defines a standard form representation of that polyhedron.

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.

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).

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

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 draw a proposal from the transition-distribution and work out the ratio of probabilities of choosing a transition and its reverse transition.

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 into 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 I" 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 present 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 towards generating a proposal 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 and all feasible points lie at the minimum. So, the problem is one of convex optimisation, complicated by the fact that we're only interested in integer feasible solutions. Given this, a sensible strategy would be to somehow move down the gradient of ι towards optimal values.

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

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

Since the total infeasibility is convex, if D_j^μ is non zero we can tell which direction the j^{th} non-basic variable, X_{Nj}, needs to be perturbed in order to potentially reduce infeasibility (or more precisely, which direction will definitely increase infeasibility). So, variables for which a bound-swap would move down the gradient D^μ 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 1 if a perturbation to X_{Nj} is (potentially) able to move down the gradient, and 0 otherwise.

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

$$P(j|\mu) = \frac{e^{k_c E_j^\mu}}{\sum_l e^{k_c E_l^\mu}} \tag{15}$$

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, $\mu \to \mu'$, is chosen with a probability proportional to the exponential of $-k_r \iota^{\mu'}$ for some constant k_r. i.e.

$$P(\mu'|j,\mu) = \frac{e^{-k_r \iota^{\mu'}} e^{k_c E_j^{\mu'}}}{\sum_{\rho \in \alpha_j^\mu} e^{-k_r \iota^\rho}} \tag{16}$$

5.4.3 Proposal function summary

Given a transition from state μ to state $\mu' \neq \mu$ there is a unique non-basic variable X_{Nj} that was involved in the transition [4] we'll use the notation $j^{\mu \to \mu'}$ to denote the index of this variable.

Starting from state μ, the probability of proposing a transition to μ' is the probability that we choose the non-basic variable, $X_{Nj^{\mu\to\mu'}}$ then choose the state μ' from $\alpha^\mu_{j^{\mu\to\mu'}}$, so from (15) and (16)

$$P(\mu \to \mu') = P(\mu'|j^{\mu\to\mu'},\mu)P(j^{\mu\to\mu'}|\mu) = \frac{e^{-k_r \iota^{\mu'}} e^{k_c E^{\mu'}_{j^{\mu'\to\mu}}}}{\sum_{\rho \in \alpha^\mu_{j^{\mu\to\mu'}}} e^{-k_r \iota^\rho}} \frac{e^{k_c E^\mu_{j^{\mu\to\mu'}}}}{\sum_l e^{k_c E_l^\mu}}$$

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

$$\frac{P(\mu')P(\mu' \to \mu)}{P(\mu)P(\mu \to \mu')} = \frac{P(\mu')}{P(\mu)} \frac{e^{-k_r \iota^\mu} e^{k_c E^\mu_{j^{\mu\to\mu'}}}}{\sum_{\rho \in \alpha^{\mu'}_{j^{\mu'\to\mu}}} e^{-k_r \iota^\rho}} \frac{\sum_{\rho \in \alpha^\mu_{j^{\mu\to\mu'}}} e^{-k_r \iota^\rho}}{e^{-k_r \iota^{\mu'}} e^{k_c E^{\mu'}_{j^{\mu'\to\mu}}}} \frac{e^{k_c E^{\mu'}_{j^{\mu'\to\mu}}}}{\sum_l e^{k_c E_l^{\mu'}}} \frac{\sum_l e^{k_c E_l^\mu}}{e^{k_c E^\mu_{j^{\mu\to\mu'}}}}$$

However thanks to the fact that we included the null transition, $\alpha^{\mu'}_{j^{\mu\to\mu'}} = \alpha^\mu_{j^{\mu'\to\mu}}$ [5] the two sums over α cancel, leaving

$$\frac{P(\mu')P(\mu' \to \mu)}{P(\mu)P(\mu \to \mu')} = \frac{P(\mu')}{P(\mu)} \frac{e^{-k_r \iota^\mu}}{e^{-k_r \iota^{\mu'}}} \frac{\sum_l e^{k_c E_l^\mu}}{\sum_l e^{k_c E_l^{\mu'}}} \tag{17}$$

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. If we let

$$P(\mu) = P_e(T_\mu|\Omega) e^{-k_r \iota^\mu} \frac{1}{n} \sum_l e^{k_c E_l^\mu} \tag{18}$$

where 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 (4), while for infeasible states is calculated in the same way but the requirement that the trajectory belongs to $\mathcal{T}^N_{SA}(F)$ 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.

Inserting (18) into (**??**) gives the final form for the acceptance

$$\frac{P(\mu')P(\mu' \to \mu)}{P(\mu)P(\mu \to \mu')} = \frac{P_e(T_{\mu'}|\Omega)}{P_e(T_\mu|\Omega)} \tag{19}$$

Notice that states with lower infeasibility and lower potential energy have higher probability, so transitions that immediately reduce infeasibility are encouraged, but so are pivots that replace high potential energy variables with low potential variables. This has the consequence that if a high potential, non-basic variable is chosen but it can't immediately reduce infeasibility, it is more likely to be made basic where it is free to change its value in subsequent transitions. This helps the process find a basis where infeasibility-reducing transitions exist.

[4] if 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 (12)

[5] since $\alpha^\mu_{j^{\mu\to\mu'}}$ describes the set of states that lie on the straight line that passes through μ and μ'

5.4.4 Choosing temperature

The exponential term $e^{\frac{-E^{\mu}}{T}}$ in equation (18) is a Boltzmann distribution so T can be thought of as a "temperature" and the sampling process will act as a thermodynamic system in equilibrium (with the posterior $P_e(T_\mu|\Omega)$ acting as an additional potential energy). It has been shown that thermodynamic systems of this type are able to solve large integer optimisation problems (Kirkpatrick, Gelatt, & Vecchi, 1983). In our case, we don't need to change the temperature during the sampling process, but we do need to choose a value of T initially. Higher temperatures will increase mixing in the Markov chain, but will also increase the proportion of time spent in infeasible states so we need to find a temperature that is high enough to ensure good mixing but low enough to ensure a reasonable proportion of samples are feasible.

[TODO: Effective procedure for choosing T and rationale. Choose start and target states at random and make linear probability with target state as maximum, then tune T to minimum samples to first hitting target. Repeat. Proportion of infeasible samples against proportion of feasible->infeasible samples. Average infeasible chain length proportion of infeasible samples/proportion of feasible->infeasible transitions.]

If we first sample uniformly from the set of all states, we can get an idea of the number of states at each energy level...or, during burn-in we reduce temperature until we hit the desired proportion of (increase temp while feasible, reduce while infeasible, gradually reducing the size of the step).

The ratio of κ_r and κ_c: If κ_c is too low, we will tend to have long infeasible runs as high potential columns will not be pivoted-in, if it is too high, then many potential transitions into infeasibility will be rejected due to the increase in potential energy. If κ_r is to high (absolute value) then transitions into infeasibility will not be proposed often enough, if too low then potential transitions back into feasibility may be ignored even when available. The ratio is the ratio of the energy in an infeasible variable and a high-potential column....how important is reducing infeasibility against reducing potential energy?...

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 (5) 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-complete[6] 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

$$\mathrm{supp}(P(\rho(.) = y)) \subset \mathcal{P}(\rho, v)$$

Tools to perform abstract interpretation already exist (e.g. PAGAI(Henry et al., 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 $\mathrm{supp}(P(\rho(.) = y))$). 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 $\mathrm{supp}(P(\rho(.) = y))$ then it will just be rejected. This is fine as long as we generate acceptable proposals at a reasonable rate.

[6]Consider, 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

9 Notes

Why does convergence scale so badly with number of timesteps? Probably because as the number of non-zero elements per col/row increases, so does the average energy change in a feasible to infeasible transition, so we must increase the temperature to get the same amount of mixing. This results in a lot more time spent in infeasible space.

9.1 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 (1) and (2) but must also satisfy the *Fermionic constraints*

$$\mathcal{B}_{SA}^{N} = \left\{T \in \mathbb{R}_{SA}^{N} : \forall t, \psi, a : T_{\psi a}^{t} \in \{0, 1\}\right\} \tag{20}$$

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 (20) and (2).

$$\mathcal{F}_{SA}^{N}(F) = \mathcal{B}_{SA}^{N} \cap \mathcal{C}_{SA}^{N}(F) \tag{21}$$

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

$$supp(P(T|\Omega, T \in \mathcal{B}_{SA}^{N})) = \mathcal{B}_{SA}^{N} \cap supp(P(T|\Omega)) \tag{22}$$

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.

Theorem 1. *For any set $\mathcal{X}$ there exists a convex polyhedron 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 $\mathcal{B}_{SA}^{N} \cap \mathcal{X} = \mathcal{B}_{SA}^{N} \cap \mathcal{P}$. □

References

Baumert, S., Ghate, A., Kiatsupaibul, S., Shen, Y., Smith, R. L., & Zabinsky, Z. B. (2009). Discrete hit-and-run for sampling points from arbitrary distributions over subsets of integer hyperrectangles. *Operations Research*, *57*(3), 727–739.

Becchi, A., & Zaffanella, E. (2018). An efficient abstract domain for not necessarily closed polyhedra. In *International static analysis symposium* (pp. 146–165).

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).

Fukuda, K. (2020). Polyhedral computation.
doi: https://doi.org/10.3929/ethz-b-000426218

Gurfinkel, A., & Navas, J. A. (2021). Abstract interpretation of LLVM with a region-based memory model. In *Software verification - 13th international conference, VSTTE 2021, and 14th international workshop, NSV 2021, october 18-19, 2021, revised selected papers.*

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

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671–680.

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.

Meel, K. S., Vardi, M. Y., Chakraborty, S., Fremont, D. J., Seshia, S. A., Fried, D., ... Malik, S. (2016). Constrained sampling and counting: Universal hashing meets sat solving. In *Workshops at the thirtieth aaai conference on artificial intelligence.*

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.

Quinton, P., Rajopadhye, S., & Risset, T. (1996). On manipulating z-polyhedra. *Technical Report 1016, IRISA, Campus Universitaire de Bealieu, Rennes, France*.

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

Talagrand, O. (1997). Assimilation of Observations, an Introduction. *Journal of the Meteorological Society of Japan. Ser. II*, *75*(1B), 191–209.

Thiele, J. C., Kurth, W., & Grimm, V. (2014). Facilitating Parameter Estimation and Sensitivity Analysis of Agent-Based Models: A Cookbook Using NetLogo and 'R'. *Journal of Artificial Societies and Social Simulation*, *17*(3). doi: 10.18564/jasss.2503

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