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 16, 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.

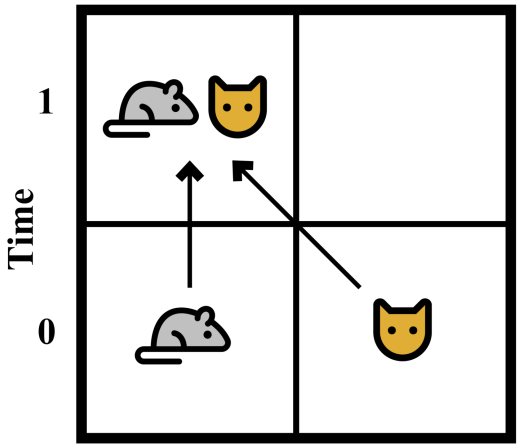

Figure 1: A simple cat and mouse model.

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

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

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[2]. We call this the *continuity constraint*

[2]This does not mean that agents cannot leave or enter the system, only that if they do then that change must be defined as part of an action.

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 (i.e. $C \in \mathbb{Z}^{SA}_{NJ}$, $L \in \mathbb{Z}_J$ and $U \in \mathbb{Z}_J$ for some J. 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.

So, if we let $\mathcal{P}^N_{SA}(f)$ be a $\mathbb{Z}$-polyhedron that over-approximates $\text{supp}(f)$ then from equation (5)

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

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

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

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

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

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

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

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 u, the maximum value that elements of the trajectory can take. In practice, this is not a problem as we can give u a value such that the probability of any trajectory of interest having any element larger than u is 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 can be expressed as a $\mathbb{Z}$-distribution.

As an illustration, consider a two-timestep trajectory of the cat and mouse model described in section **??**. Suppose we flip a fair coin to decide whether each agent state is occupied or empty at $t = 0$. Suppose also 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 is the constraint due to agent interactions. The impossible interactions 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

$$\forall t, \psi, a : 0 \leq T^t_{\psi a} \leq 1$$

then using the identity in (8) to take the union of each constraint in (9) with $\left\{T : T^t_{\psi a} = 0\right\}$ finally 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 $t = 1$ to describe the agent interactions.

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 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.

4.1 The set of Markov states

Given a $\mathbb{Z}$-polyhedron, we split the constraints into two sets: equalities (i.e. those whose lower and upper bound have the same value) and inequalities (i.e. those whose lower and upper bounds differ):

$$\mathcal{P} = \left\{ T \in \mathbb{Z}^N_{SA} : L \leq \sum_{t,\psi,a} C^{\psi a}_t T^t_{\psi a} \leq U \cap \sum_{t,\psi,a} D^{\psi a}_t T^t_{\psi a} = E \right\} \tag{10}$$

Suppose there are N_e equality constraints (i.e. E is an N_e dimensional vector) and we partition the elements of a trajectory T into 'basic' and 'non-basic' elements, so that there are exactly N_e basic elements and $N \times S \times A - N_e$ non-basic elements. Given such a partition we can define tensors $Q \in \mathbb{R}^{NN_e}_{SA}$ and $R \in \mathbb{R}^{NJ}_{SA}$, where $J = NSA - N_e$, that take 'flattened' vectors of basic, X, and non-basic, Y, variables and project them into a trajectory tensor such that

$$T^t_{\psi a} = \sum_i Q^{ti}_{\psi a} X_i + \sum_j R^{tj}_{\psi a} Y_j$$

If we now let

$$B^i_k = \sum_{t,\psi,a} D^{\psi a}_{tk} Q^{ti}_{\psi a}$$

and

$$N^j_k = \sum_{t,\psi,a} D^{\psi a}_{tk} R^{tj}_{\psi a}$$

then we can write the equality constraints as

$$\sum_i B_k^i X_i + \sum_j N_k^j Y_j = E_k$$

Now, if we choose the basic variables in such a way as to ensure that B has an inverse, then given an assignment of values to X_N we have

$$X_i = \sum_k (B^{-1})_i^k (E_k + \sum_j N_k^j Y_j)$$

So, we define a Markov state to be an assignment to the non-basic variables Y and map this to the trajectory using

$$T_{\psi a}^t = \sum_i Q_{\psi a}^{ti} \sum_k (B^{-1})_i^k (E_k + \sum_j N_k^j Y_j) + \sum_j R_{\psi a}^{tj} Y_j \tag{11}$$

subject to

$$0 \leq Y_j \leq u$$

where u is the upper limit on the occupancy of agent actions.

4.2 The probability of a Markov state

Given a Markov state, Y, that satisfies $0 \leq Y_j \leq u$, the associated trajectory given by equation (11) is guaranteed to satisfy all the equality constraints. However, there is no guarantee that the basic variables will be within their bounds or that the inequality constraints will be satisfied. To deal with this we introduce the concept of *infeasibility* which is a measure of how far a Markov state is from being feasible.

Given a trajectory, T, and a set of inequality constraints

$$L_i \leq \sum_{t,\psi,a} C_{ti}^{\psi a} T_{\psi a}^t \leq U_i$$

let the infeasibility of the i^{th} constraint, ι_i^T, be defined to be equal to 0 if the variable is within its bounds or equal to the distance to the nearest bound otherwise

$$\iota(T)_i = \begin{cases} L_i - \sum_{t,\psi,a} C_{ti}^{\psi a} T_{\psi a}^t & \text{if } \sum_{t,\psi,a} C_{ti}^{\psi a} T_{\psi a}^t < L_i \\ \sum_{t,\psi,a} C_{ti}^{\psi a} T_{\psi a}^t - U_i & \text{if } \sum_{t,\psi,a} C_{ti}^{\psi a} T_{\psi a}^t > U_i \\ 0 & \text{otherwise} \end{cases}$$

and let the total infeasibility be the sum of infeasibilities of all constraints

$$\iota(T) = \sum_i \iota(T)_i$$

So, each Markov state is associated with a total infeasibility with respect to the inequalities of the $\mathbb{Z}$-distribution.

Given an upper bound on the trajectory's elements, u, we define the *clamped* trajectory

$$\overline{\underline{T}}_{\psi a}^t = \begin{cases} 0 & T_{\psi a}^t < 0 \\ u & T_{\psi a}^t > u \\ T_{\psi a}^t & \text{otherwise} \end{cases}$$

Furthermore, let $K_t^{\psi a}$ be a log-linear approximation to the target $\mathbb{Z}$-distribution so that

$$\sum_{t\psi a} K_t^{\psi a} T_{\psi a}^t \approx \log(P(T))$$

for feasible trajectories, and let the *log-linear probability*, P_K, be

$$\log(P_K(T)) = \sum_{t\psi a} K_t^{\psi a} \overline{\underline{T}}_{\psi a}^t + \kappa \iota(T) \tag{12}$$

where $\kappa < 0$ is a tunable parameter.

The probability of a Markov state, μ, is now given by

$$P(\mu) = \begin{cases} P(T) & \text{if } \iota(T) = 0 \\ P_K(T) & \text{if } \iota(T) > 0 \end{cases} \tag{13}$$

This means that infeasible states have non-zero probability, so the Markov chain will return some infeasible states. However, if we just throw these away then the remaining samples will be feasible samples of the target distribution. Allowing the Markov chain to pass through infeasible states has the advantage of ensuring that there exists a path between any two feasible states and improves mixing. The cost of this is the computational time spent moving through infeasible states that don't generate any useful samples. The κ term makes the infeasible Markov states into a Boltzmann distribution with $\frac{-1}{\kappa}$ being the "temperature". 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 $kappa$ 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 of the chain but low enough to ensure a reasonable proportion of samples are feasible. We have found that a good rule of thumb is to aim for 50% of samples infeasible.

4.3 Transitions between Markov states

Given a Markov state, we can transition to another state by choosing an element of Y and perturbing it by ± 1 (or, if the element is on one of its bounds then to perturb it away from the bound in the feasible direction. In the case of Fermionic trajectories, the elements of Y are always on their bounds so there is always exactly one perturbation per element: a swap to the other bound). This defines the set of transitions between Markov states. Let the set of all Markov states reachable from state T be denoted by $\mathcal{D}(T)$.

4.3.1 The probability of transitions

The probability of a transitions from T to T' is defined as

$$P(T \to T') = \frac{\min\left(1, \frac{P_K(T')}{P_K(T)}\right)}{S(T)}$$

where P_K is the log-linear probability defined in (12) and

$$S(T) = \sum_{T' \in \mathcal{D}(T)} \min\left(1, \frac{P_K(T')}{P_K(T)}\right)$$

so that

$$\frac{P(T' \to T)}{P(T \to T')} = \frac{P_K(T)}{P_K(T')} \frac{S(T)}{S(T')}$$

and the Metropolis-Hastings acceptance probability is

$$\alpha = \frac{P(T')P(T' \to T)}{P(T)P(T \to T')} = \frac{P_K(T)}{P(T)} \frac{P(T')}{P_K(T')} \frac{S(T)}{S(T')}$$

When T is infeasible $\frac{P_K(T)}{P(T)} = 1$. When T is feasible $P_K(T)$ approximates $P(T)$ so their ratio should be close to 1. Finally, $S(T)$ will not change very much between transitions so the ratio $\frac{S(T)}{S(T')}$ should also be close to 1.

4.4 Choosing a basis

The definition of the Markov chain depends on a partition of the trajectory into basic and non-basic elements. We now describe how such a partition is chosen.

TODO: Finish this

4.5 An efficient proposal algorithm

We next provide an efficient algorithm to choose a proposal with the correct probability from the set of transitions given the current Markov state.

TODO: Finish this

5 Results

6 Further work

6.1 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[3] 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

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

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 $\text{supp}(P(\rho(.) = y))$). 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 $\text{supp}(P(\rho(.) = y))$ then it will just be rejected. This is fine as long as we generate acceptable proposals at a reasonable rate.

7 Conclusion

8 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.

8.1 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*

$$\mathcal{B}^N_{SA} = \left\{T \in \mathbb{R}^N_{SA} : \forall t, \psi, a : T^t_{\psi a} \in \{0, 1\}\right\} \tag{14}$$

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, $\mathcal{F}^N_{SA}$, is the set of tensors that satisfy (14) and (2).

$$\mathcal{F}^N_{SA}(F) = \mathcal{B}^N_{SA} \cap \mathcal{C}^N_{SA}(F) \tag{15}$$

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

$$supp(P(T|\Omega, T \in \mathcal{B}^N_{SA})) = \mathcal{B}^N_{SA} \cap supp(P(T|\Omega)) \tag{16}$$

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 $\mathcal{B}^N_{SA}$ 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*

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

Proof. Let $\mathcal{Y} = \mathcal{B}^N_{SA} \cap \mathcal{X}$ and let $\mathcal{P}$ be the convex hull of $\mathcal{Y}$. Clearly if a tensor $T \in \mathcal{Y}$ then $T \in \mathcal{B}^N_{SA} \cap \mathcal{P}$. To show the converse, suppose there is a member $T' \in \mathcal{B}^N_{SA} \cap \mathcal{P}$. 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 $T' \in \mathcal{B}^N_{SA}$ and $\mathcal{Z} \subset \mathcal{B}^N_{SA}$ and no member of $\mathcal{B}^N_{SA}$ is a convex combination of any other member then T' must be a member of $\mathcal{Z}$ and so $T' \in \mathcal{Y}$. So $\mathcal{B}^N_{SA} \cap \mathcal{X} = \mathcal{B}^N_{SA} \cap \mathcal{P}$. □

[3]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)

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