

Discrete and continuous time simulations of spatial ecological processes predict different final population sizes and interspecific competition outcomes

Rebecca Mancy^{a,b,*}, Patrick Prosser^a, Simon Rogers^a

^a School of Computing Science, University of Glasgow, G12 8QQ, UK

^b Institute of Biodiversity, Animal Health and Comparative Medicine, University of Glasgow, G12 8QQ, UK

ARTICLE INFO

Article history:

Received 8 November 2012

Received in revised form 19 March 2013

Accepted 21 March 2013

Available online 22 April 2013

Keywords:

Cellular automaton

Discrete time

Continuous time

Spatial Gillespie simulator

Interspecific competition

ABSTRACT

Cellular automata (CAs) are commonly used to simulate spatial processes in ecology. Although appropriate for modelling events that occur at discrete time points, they are also routinely used to model biological processes that take place continuously. We report on a study comparing predictions of discrete time CA models to those of their continuous time counterpart. Specifically, we investigate how the decision to model time discretely or continuously affects predictions regarding long-run population sizes, the probability of extinction and interspecific competition. We show effects on predicted ecological outcomes, finding quantitative differences in all cases and in the case of interspecific competition, additional qualitative differences in predictions regarding species dominance. Our findings demonstrate that qualitative conclusions drawn from spatial simulations can be critically dependent on the decision to model time discretely or continuously. Contrary to our expectations, simulating in continuous time did not incur a heavy computational penalty. We also raise ecological questions on the relative benefits of reproductive strategies that take place in discrete and continuous time.

© 2013 Elsevier B.V. Open access under [CC BY](http://creativecommons.org/licenses/by/3.0/) license.

1. Introduction

Cellular automata (CAs) are commonly used to simulate dynamic spatial processes in ecology, contributing to developments in both applied and theoretical research. In a simple CA model of birth–death processes, individuals inhabit discrete sites, usually organised in a grid formation. Time progresses in discrete steps and an *update scheme* specifies how individuals die or give birth into neighbouring sites at each step. For example, in the applied literature CAs have been used to simulate the spatial distribution of insect colonies (Perfecto and Vandermeer, 2008; Vandermeer et al., 2008) and the effect of plant–soil feedbacks on relative tree abundance (Mangan et al., 2010), while in microbial ecology, Fox et al. (2008) used a CA to investigate the way in which plasmids invade bacterial populations. In contrast, Laird and Schamp (2008) used a CA to explore theoretical questions relating to differences between interspecific competition in spatial and non-spatial (homogeneous mixing) contexts while Roxburgh et al.

(2004) investigated mechanisms leading to long-term species coexistence in the context of the intermediate disturbance hypothesis. In addition, CAs have been used to validate simplifications required to solve models analytically, such as in the theoretical work on population self-structuring reviewed in Lion and van Baalen (2008), as well as in a range of publications on the evolution host–parasite interactions (see e.g. Kamo and Boots, 2004; Best et al., 2011, for parasite virulence and host resistance, respectively), the evolution of altruism (e.g. Lion and van Baalen, 2007) and the evolution of reproductive effort (Lion, 2010).

Among the assumptions embodied in CA models is that of discrete time. This in turn introduces the need to make additional assumptions in the form of modelling decisions regarding the *update scheme* used to govern the order in which sites are considered and events take place. When these modelling decisions are made carefully, CAs can form appropriate models for discrete time spatial processes. However, they are often employed to simulate continuous time ecological processes or models, frequently without acknowledgement that this introduces an additional layer of approximation. Fortunately, these continuous processes can be simulated directly using a discrete space version of the Gillespie algorithm (Gillespie, 1977). Following this algorithm, time is continuous in the sense that it progresses in arbitrarily small steps, the length of which varies according to event rates, and these are limited only by the precision of the computer on which it

* Corresponding author at: School of Computing Science, University of Glasgow, G12 8QQ, UK. Tel.: +44 330 8138.

E-mail addresses: Rebecca.Mancy@glasgow.ac.uk (R. Mancy), Patrick.Prosser@glasgow.ac.uk (P. Prosser), Simon.Rogers@glasgow.ac.uk (S. Rogers).

is implemented. Although a little more mathematically involved than discrete time approaches, the implementation of this algorithm reduces the number of modelling decisions and thus allows a stronger focus on the biology while enhancing comparability between studies. Once understood, the approach can also be applied to non-spatial and continuous space problems, as well as to evolutionary problems (see e.g. Meier et al., 2011).

Decisions about whether to simulate in continuous or discrete time, and in the latter case which update scheme to use, are not simply technical but should be made in direct relation to the dynamics of the biological system under study. In order to compare studies and make informed decisions about which approach to use, it is important to understand any disparities in predictions between continuous and discrete time simulations, especially in the case where CAs are used to model continuous time processes. Although it is known that CA update schemes (the order in which events are considered) can affect ecological dynamics (Ruxton and Saravia, 1998), differences in ecologically meaningful predictions between CAs and corresponding continuous time simulation approaches have never been tested.

In this paper, we assume stochastic real world processes that occur in continuous time with exponentially distributed waiting times between events, taking as a case study the asymmetric logistic model of population growth on a lattice (Matsuda et al., 1992). We regard this model as our benchmark and consider discrete time CA simulations as approximations to this model. Specifically, we simulate this model stochastically in continuous time and compare outcomes with those of simulations conducted using two probabilistic CA update schemes, a range of time step sizes and two methods for converting between the rates used in continuous time models and probabilities required for discrete time simulation. We conduct two experiments, focusing in the first on a single species and in the second on competition between two species. In both experiments, we report long-run population sizes, and in the context of interspecific competition, also predictions regarding coexistence and competitive exclusion, using these outcomes to highlight disparities between discrete and continuous time.

In the following sections we consider some of the modelling decisions that need to be made when using CAs, emphasising the conversion from rates to probabilities required to approximate continuous processes. We describe our experimental protocol, provide findings from our two experiments and conclude with modelling recommendations.

2. Modelling decisions of cellular automata

Provided that time steps are chosen carefully so that they match the periodicity of real ecological events, discrete time simulations can be appropriate when simulating ecological processes that occur synchronously (e.g. reproductive cycles in cicadas) or where there are strong cyclical patterns (e.g. due to seasonality). Their use becomes more difficult to justify when modelling continuous processes (e.g. disease transmission) or to validate analytic simplifications in continuous time models. Nonetheless, justifications for employing discrete time simulations, decisions regarding particular choices of update scheme and method of converting rates to probabilities are rarely reported (although see Best et al., 2011; Ovaskainen and Hanski, 2003, for articles including this information). This makes replication almost impossible, as well as hindering comparisons between studies and the interpretation of any conflicting findings. In this study, we investigate the extent of these problems by simulating the same system in continuous and discrete time, taking a continuous time model as our benchmark. We limit our discussion to one or

more species living on a finite grid, and assume ecological processes that take place continuously according to a well-understood model.

Simple CA models of ecological processes are usually constructed in the following way: organisms live on a grid of sites; time progresses in discrete time steps and at each iteration individuals persist, die, or give birth into *neighbouring* sites according to a set of *local transition rules*. In probabilistic models, event probabilities are often dependent on the configuration of occupied and empty sites in the neighbourhood. CAs are relatively straightforward to implement, requiring limited mathematical or modelling knowledge (Berec, 2002; Breckling et al., 2011) but their very ease of implementation belies a range of complexities. Specifically, important modelling decisions arise as a result of the discrete nature of time: these concern the order in which events are executed and the way in which event rates are converted to probabilities.

The issue of event ordering arises because in discrete time, events may occur simultaneously at the same site (e.g. two births into the same site) and decisions thus need to be made about the order in which events should take place and how to resolve competition. An *update scheme* is therefore used to determine event ordering. A large number of schemes have been proposed and comparisons between these in the computing science, theoretical physics and ecological literature demonstrate important differences in dynamics and steady state outcomes (e.g. Manzoni, 2012; Ingerson and Buvel, 1984; Lumer and Nicolis, 1994; Schönfisch and de Roos, 1999; Cornforth et al., 2002; Ruxton and Saravia, 1998).

Event frequency in continuous time is typically characterised by event rates and the assumption that waiting times between events are exponentially distributed. For use in CA models, these rates must be converted into event probabilities. We use two different approaches in our study, one that allows for multiple events and one that allows only a single event per time step Δt . In the first, we make use of the fact that for a process with exponentially distributed waiting times, the number of events within a specified time window follows a Poisson distribution. Thus, we sample the number of events from a Poisson distribution with parameter $r\Delta t$ where r is the instantaneous rate (note that this only makes sense for births). When discrete time is viewed as an approximation to a continuous process, this is similar to the τ -leaping idea proposed by Gillespie (2001). The second conversion is a cruder approximation that allows a maximum of one event per time step, bringing the simulation into line with most common CA approaches (see e.g. Best et al., 2011, for a study where this conversion is described explicitly). Probabilities in this approach are computed from rates as described in Section 3.3. System dynamics are expected to differ between conversion approaches and although it is known that reducing the time step should limit this effect (see e.g. Schönfisch and de Roos, 1999), it is unclear how small Δt needs to be before particular qualitative and quantitative properties of ecological models are indistinguishable.

3. Experimental protocol

Following Ruxton and Saravia's (1998) comparison of CA update schemes, we take as a case study one of the simplest spatial models, the asymmetric logistic model of population growth on a lattice. We simulate a stochastic version of this model for different birth and death rates using a model with exponentially distributed waiting times between events. This is compared to simulations using two CA update schemes, a range of time steps and two methods used to convert from rates to probabilities. In our analysis, we consider the continuous time simulation as our benchmark and the discrete time simulations as approximations to this. In Experiment

1, we simulate a single species and consider differences in long-run population sizes and probability of extinction. In Experiment 2, we consider two species and compare interspecific competition outcomes. Simulations are conducted for 1000 time steps and 100 repetitions unless otherwise stated.

3.1. Continuous time model

The lattice logistic model describes population growth that is regulated by the local availability of empty sites. In the standard version of the model, organisms live on an infinite network of sites, each of which is connected to n randomly selected neighbours. Organisms have two fundamental behaviours – birth and death – governed by rates, and can only give birth if there is an empty site in their neighbourhood (Matsuda et al., 1992).¹ We simulate the stochastic version of this continuous time model as a Poisson process. This is implemented in the form of a spatial version of the Gillespie algorithm (Gillespie, 1977) that resembles the algorithm proposed by Stundzia and Lumsden (1996) and that we refer to as *Gill*. The algorithms used are described formally in Appendix A and the code is available from <http://rebeccamancy.github.io/gillespie-cellular-automaton/>.

3.2. Cellular automaton update schemes

We compare the outcomes of our continuous time simulations to those of a range of CAs. We select two update schemes among those considered by Ruxton and Saravia (1998), deliberately choosing schemes in which site order is random and that differ in the level of structure introduced (Schönfisch and de Roos, 1999) to investigate the effects of update timing and event orderings.

The first scheme introduces structure by fixing the order of birth and death events and through delayed updating. At each generation, a new random site order is generated which is then used to execute death at all occupied sites and then birth at all occupied sites probabilistically, after which the population is updated for the next generation. The scheme is implemented as *RF_d2S* and *RF_d2M* following the naming conventions in Ruxton and Saravia (1998).² The second scheme introduces less structure since event ordering is random and updates are fully asynchronous. We generate a list of (site, event) pairs, executing these in a new random order at each generation and updating the state of system after each event. In Ruxton and Saravia (1998) this scheme is referred to as *RR1* and is implemented here for the two forms of rate conversion as *RR1S* and *RR1M*.

3.3. Rate conversion

We use two methods to convert rates: in the first, multiple births are permitted whereas in the second, an additional layer of approximation is introduced since at most one birth is performed. In the first approach, referred to as *multiple births*,³ we

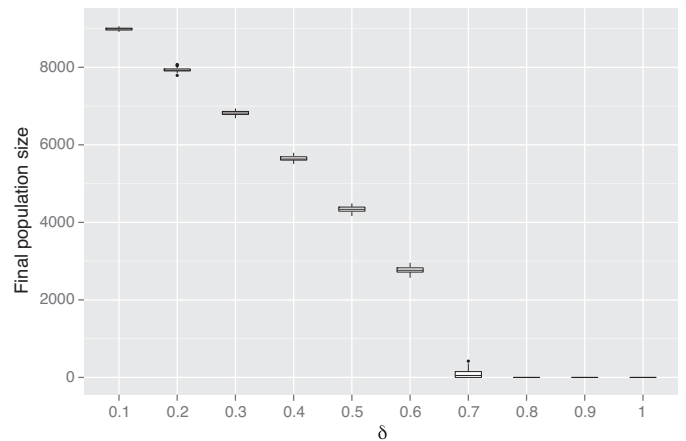


Fig. 1. Box plot showing final population sizes for death-to-birth ratios δ under *Gill*.

draw a number of events n_v where $n_v \sim \text{Poiss}(r\Delta_t)$ where, as previously, r represents the instantaneous rate and Δ_t is the time step. This conversion makes the assumption that births within a time step are independent.⁴ In the second, we assume that multiple births within a time step never occur so $\Pr(n_v > 1) \approx 0$ and thus that $\Pr(n_v = 1) \approx 1 - \Pr(n_v = 0) = 1 - e^{-r\Delta_t}$ (see e.g. Fleurence and Hollenbeak, 2007). Multiple births versions of the algorithms are suffixed *M* and single births *S*.

3.4. General model parameters

In our experiments, we consider *Gill* as our benchmark and compare with the four CA update schemes *RF_d2S*, *RR1S*, *RF_d2M* and *RR1M*, explained in Appendix A. We simulate all schemes for 1000 time units on a regular square lattice of 100×100 sites using a neighbourhood consisting of the 8 nearest neighbours (Moore neighbourhood) and wrapping boundaries in the form of a torus to mimic the infinite lattice of the theoretical model.⁵ We refer to the population after 1000 time units as the *final population*; in many cases this corresponds to the pseudo-steady state of the system although time to convergence depends on parameter values. We run 100 stochastic repetitions of all simulations and report summary statistics where appropriate. Except where otherwise indicated, all populations start with 1000 individuals randomly distributed across sites. In each experiment, we hold the intrinsic birth rate of organisms constant at $b=1$ and vary the death rate in steps of 0.1 from 0.1 to 1.0 inclusive,⁶ giving death-to-birth ratios, denoted δ , in the range 0.1–1.0. We test time step values in the set $\{2^0, 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}\}$. The simulation code is programmed in Java and the full code release is available from <http://rebeccamancy.github.io/gillespie-cellular-automaton/>, with post-processing conducted in R.

4. Experiment 1: single population

In Experiment 1 we explore final population sizes under *Gill* and the four CA update schemes. Final population sizes under *Gill* for

¹ In some versions of the lattice logistic model, death rate is also related to the density of neighbours. In the version we implement, death occurs at a constant rate independent of overcrowding while reproduction is limited by resource constraints.

² In the CA algorithm names, the first letter refers to site ordering, the second to event ordering, and the number to whether updating takes place immediately (1, because a single array is required) or with a delay (2, as two arrays are required). Specifically, *R* refers to the random order in which sites are visited, *F_d* indicates that the order of birth and death events is fixed with death occurring first, 2 indicates that two arrays are used to store the configurations to allow delayed updating, and the final letter (see Section 3.3) refers to the approach to rate conversion. The same scheme is referred to as *RF2* in Ruxton and Saravia (1998), where no birth-first schemes are considered.

³ This method is applied only to births as each site is only visited once per generation so death can occur at most once.

⁴ A similar approach, known as *tau-leaping*, is used in approximations of the standard Gillespie algorithm (Gillespie, 2001).

⁵ Our choice of network topology is motivated by the prevalence of square lattices in the literature; we acknowledge the arguments for the use of hexagonal lattices (Birch, 2006; Birch et al., 2007; Holland et al., 2007; White and Kiester, 2008). The implemented model diverges from the theoretical model in that latter assumes an infinite random regular network rather than an orthogonal lattice.

⁶ A similar approach is used by Ruxton and Saravia (1998) in choosing a death rate for their simulations; however, we test the full range of death rates throughout.

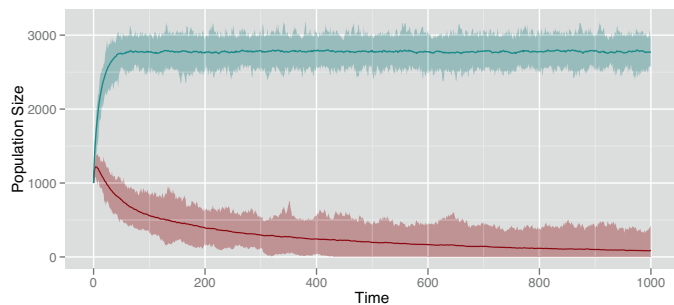


Fig. 2. Time series plot showing mean and range for $\delta = 0.7$ (lower line) and $\delta = 0.6$ (upper line) under *Gill*.

the different death-to-birth ratios (δ) are shown in Fig. 1, demonstrating small variance between runs. In general, populations with lower values of δ are larger, and for values of $\delta = 0.6$ and below, the population grows rapidly and reaches quasi-stationarity; extinction occurs slowly for $\delta = 0.7$ and rapidly for higher values (see Fig. 2). We refer to the transition between persistent populations and population extinction that occurs between $\delta = 0.6$ and $\delta = 0.7$ as the *persistence threshold* (Adler and Nuernberger, 1994).

Comparing final population sizes and proportion of runs extinct under *Gill* and the CA algorithms, qualitative patterns differed little between the two single birth algorithms and we focus on RF_d2S . For all parameter values, RF_d2S underestimates final population size and the largest differences in mean final population size occur for the longest time steps and intermediate values of δ . The finding that the largest deviations are found for the longest time steps is expected. The deviation for intermediate values of δ can be explained by the occupancy level of grid: at these values, occupancy is around 50% and population size is more sensitive to differences between the algorithms than at lower δ values where lack of available sites dominates algorithm effects. Because final populations under *Gill* are small for $\delta = 0.7$, differences between *Gill* and RF_d2S are also small, even though all RF_d2S runs actually went extinct for the four longest time steps (see Fig. 3(b) for proportions of runs extinct by $t = 1000$ under RF_d2S).

Fig. 3(b) shows *algorithm bias*, the proportion under-estimate of final population size compared with *Gill* for RF_d2S . Strongest bias was found for δ close to the persistence threshold, and was worst for longer time steps. The highest levels of algorithm bias are explained by the higher extinction rates of RF_d2S than *Gill*; nonetheless, even for parameters where populations did not go extinct under either algorithm, bias increases as we increase step size and towards the persistence threshold. Comparing the proportion of runs that went extinct under *Gill* (Fig. 3(b)) with those of RF_d2S (Fig. 1) shows that populations simulated under *Gill* are more resilient and this difference is most obvious at $\delta = 0.7$ (31% runs extinct under *Gill* compared with 71% under RF_d2S and 67% under *RR1S* for the shortest time step).

Similar patterns were seen in the relationship between accuracy and parameter values for the multiple births scheme RF_d2M , except that this algorithm overestimated population sizes for all parameter values apart from $\delta = 0.7$ (Fig. 4). In contrast, the *RR1M* scheme gave large overestimates of final population size for all parameter values, and no run went extinct. The difference between the two multiple birth versions of the algorithms is very marked: although simulating multiple births improves estimates when updating is delayed until the end of a generation (RF_d2M) it produces large overestimates when updating takes place immediately (*RR1M*). Under the RF_d2M scheme, the increase in birth rate due to multiple births is partially compensated by the increase in death rate since all organisms are considered for death at the next time step. Under *RR1M*, births can take place into sites that have already been

evaluated for death during the current generation and are therefore not evaluated for death until the following generation. In other words, the effective birth rate is increased much more than the corresponding death rate for *RR1M*; this algorithm gives a poor approximation of *Gill* and we consider it no further.

Finally, we consider the minimum time step required to generate final population sizes that are statistically indistinguishable from those of *Gill* on the basis of *t*-tests. Table 1 shows that RF_d2S performed slightly better than *RR1S*, but this effect is largely due to the fact that sizes under RF_d2S represent the highest point in each birth-death cycle as population size is measured after births. To simulate final population sizes that are statistically indistinguishable from *Gill*, an unidentified step sizes below 2^{-6} is required for *RR1S* for all $\delta > 0.1$, whereas this is the case for RF_d2S only for δ close to the persistence threshold. Among the multiple birth schemes, RF_d2M performed better than both single birth schemes for values of δ close to the persistence threshold, but less well than RF_d2S for low δ . The impact of simulating multiple births in RF_d2M also had a greater positive effect on algorithm bias for larger step sizes where the largest errors were found under the single births version. This is because with larger step sizes *Gill* tends to generate more cases of multiple births within the equivalent of a time step, so the single birth schemes are more inaccurate for larger step sizes.

In conclusion, Experiment 1 demonstrated relatively large discrepancies in final population sizes between the continuous and discrete time simulations, and these differed with time step and death-to-birth ratio. Time steps need to be reduced to values of 2^{-3} or smaller for final population sizes to be statistically indistinguishable from those of the Gillespie simulator, although required step size depended on both the update scheme and δ . It is therefore important for researchers who simulate in discrete time to be explicit about the time step and update scheme employed. Simulating multiple births to better emulate the possibility of multiple births under Gillespie was a helpful strategy under RF_d2M , at least for values of δ near the persistence threshold, but not under *RR1M* where it introduced heavy bias. Overall, the step sizes required to accurately approximate continuous time are small, they have a complex relationship with other model parameters and the computational cost of simulation under sufficiently small step sizes is high.

5. Experiment 2: interspecific competition

We now investigate the outcomes of interspecific competition between two species. The question of interspecific competition is of importance in a range of practical contexts such as when predicting the spread of invasive species and has also been studied extensively in the mathematical biology literature (see Vandermeer and Yitbarek, 2012, for a recent example in a spatial context). For a range of simple deterministic models where competition is for a single resource, it can be shown that there are four possible biological outcomes at equilibrium: *extinction* of both species, *competitive exclusion* of species one by species two, *competitive exclusion* of species two by species one and *coexistence* (Levin, 1974). The competition model under logistic growth in the non-spatial case for species subscripted 1 and 2 can be written as

$$\dot{p}_1 = (b_1 p_0 - d_1) p_1$$

$$\dot{p}_2 = (b_2 p_0 - d_2) p_2$$

where p represents the population density (subscript zero indicating density of empty sites), b the birth rate and d the death rate. The equilibrium condition can be found by solving simultaneously for $\dot{p}_1 = \dot{p}_2 = 0$, also showing that coexistence is possible only when the two species have exactly the same death-to-birth ratio $\delta_1 = \delta_2$

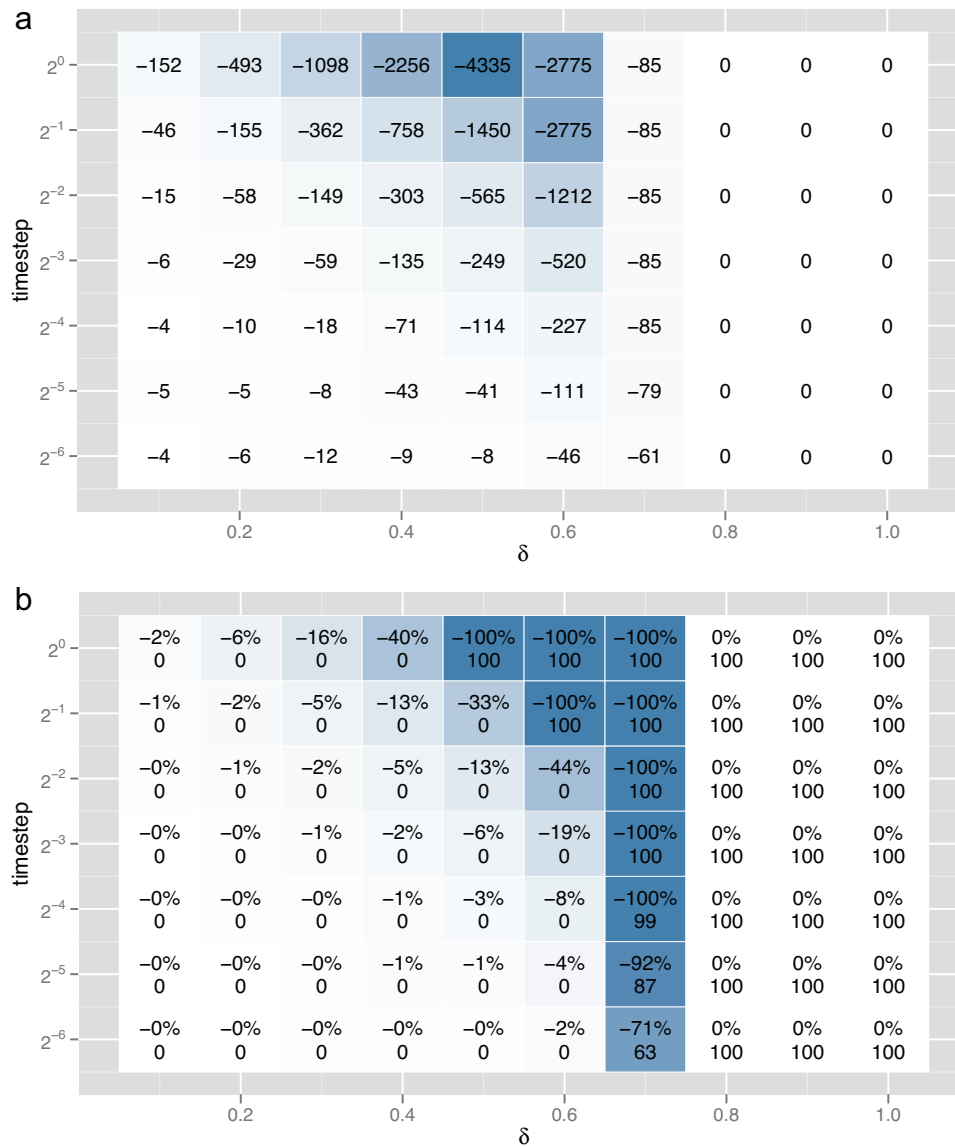


Fig. 3. Plots of (a) difference in average population size (including extinct runs) for *Gill* – *RF_d2S* (shading highlights difference in final population size) and (b) percentage algorithm bias (upper value in each cell; shading highlights size of algorithm bias) and number of extinct runs out of 100 under *RF_d2S* (lower value).

and that a slight disadvantage for either species eventually leads to its competitive exclusion. We therefore also expected our spatial model to be sensitive to small deviations around this point. Since the conversion from continuous to discrete time introduces fine adjustments to birth and death rates, any differences between update schemes and methods of modelling time are likely to be apparent around this point. The decision to simulate at this point implies that differences uncovered in the experiments described below constitute a worst case scenario; however, we believe that they constitute a relevant and realistic one. For example, it is

not unreasonable to assume that death-to-birth ratios of invasive species will be similar to those of native species. Furthermore, many of the studies in the theoretical literature that use CAs to validate analytic simplifications (e.g. Lion and van Baalen, 2007) are concerned with the evolution of altruism, where fine adjustments of birth and death rates for otherwise similar species are of particular interest.

We simulate interspecific competition for a range of birth-to-death ratios, holding constant the relationship between species such that $\delta_1 = \delta_2$ and with starting populations of 1000 for each

Table 1

Maximum time step size giving indistinguishable final population sizes as *Gill* (2-tailed Student *t*-test, unequal variances at the 5% level).

δ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
<i>RF_d2S</i>	2^{-3}	2^{-4}	2^{-5}	2^{-6}	–	–	–	N/A	N/A	N/A
<i>RR1S</i>	2^{-6}	–	–	–	–	–	–	N/A	N/A	N/A
<i>RF_d2M</i>	2^{-6}	2^{-6}	2^{-6}	–	2^{-6}	2^{-5}	2^{-3}	N/A	N/A	N/A
<i>RR1M</i>	–	–	–	–	–	–	–	*	*	*

– indicates that final population sizes differed for all of the time steps considered; N/A indicates populations that went extinct for both simulation approaches; * indicates that this algorithm did not correctly predict extinction for these values.

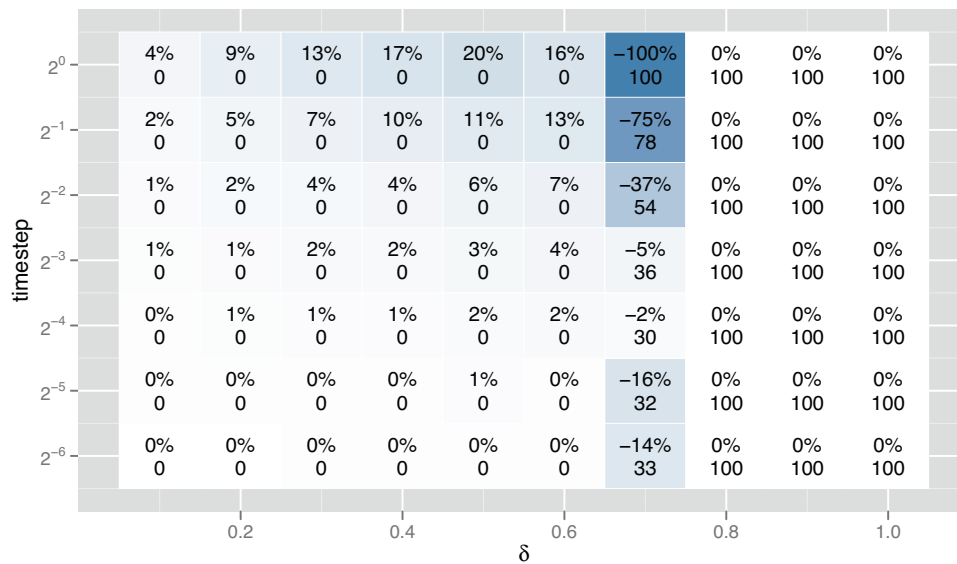


Fig. 4. Percentage algorithm bias for RF_d2M (upper value in each cell) and number of runs extinct out of 100 (lower value). Negatives represent underestimates compared with *Gill*; shading highlights absolute algorithm bias.

species. We use the same range of birth and death rates as in Experiment 1 for species one and allocate exactly half these rates to species two, referring to species one as having *higher population turnover*. We investigate the range of parameter values for which the CA models predict the same interspecific competition outcomes as *Gill*. In line with the stochastic nature of our simulations in which all populations would ultimately go extinct, we consider that qualitative predictions are the same when in both algorithms the same species dominates over 50% of runs at time 1000 in the sense of having larger population size.

Fig. 5 shows final population sizes for *Gill*. Both species went extinct in all runs for values of δ of 0.8 and above. For values of 0.5 and below, the population with the higher population turnover (species one) demonstrated larger final population sizes, although the difference failed to reach statistical significance for $\delta=0.5$. In contrast, for $\delta=0.6$, the species with slower population turnover showed higher average population sizes. For $\delta=0.7$, species one went extinct in most runs, and species two also tended

towards extinction, but more slowly. Overall, higher population turnover is the more effective strategy for low values of δ while lower population turnover is the preferred strategy near to the persistence threshold. Time series plots showed that for persistent runs ($\delta < 0.7$), the mean population sizes were stable, although variability between runs was large.

We now examine the conditions under which the CA schemes give the same qualitative predictions as *Gill*. Fig. 6 shows the results of interspecific competitions for RF_d2S and RF_d2M (findings for $RR1S$ were very similar to RF_d2S and are not shown). Overall, RF_d2M performed better than RF_d2S , making the same qualitative predictions as *Gill* for a larger range of parameter values (for all values of δ for step sizes 2^{-2} and smaller). In contrast, RF_d2S predicted an advantage for the population with slower turnover for larger step sizes and all values of δ , and for lower values of δ this advantage was strong. Under RF_d2S , step size needed to be reduced to 2^{-4} before qualitative predictions concurred with *Gill* for any value of δ , and for $\delta=0.4$ and $\delta=0.5$ none of the time steps tested were sufficiently

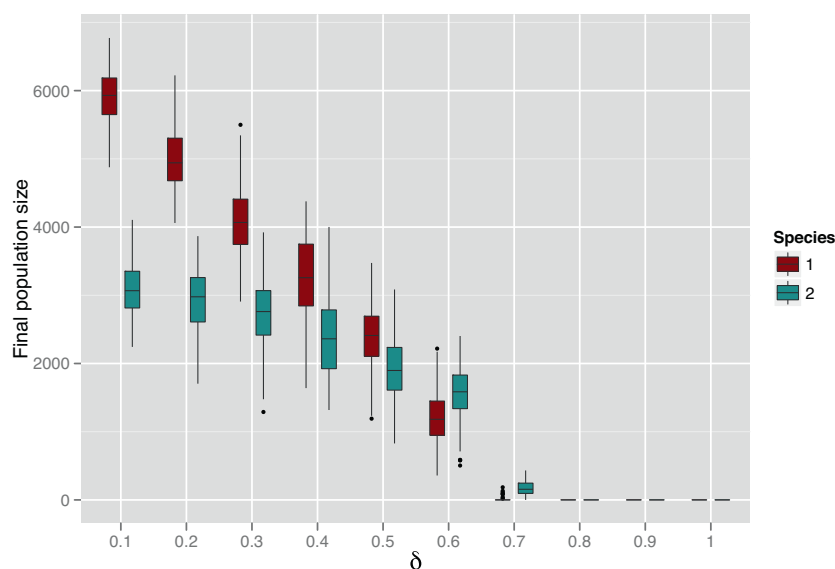


Fig. 5. Box plot showing final population sizes for the two species under different death-to-birth ratios δ under *Gill*.



Fig. 6. Interspecific competition for RF_d2S (top panel) and RF_d2M (bottom panel). The upper two values in each cell represent the final population size for species one and two respectively with bold type used to highlight the larger of the two values. The bottom value in each cell gives percentage of runs in which species one dominated at time 1000. Shading indicates parameter sets giving the same qualitative predictions as *Gill*.

small for the two models to make the same predictions. Fig. 7 highlights the differences in competitive outcomes between the update schemes and shows the time series plots for *Gill*, RF_d2S and RF_d2M for $\delta = 0.1$ and for the CA models, a time step of size 1. The time series differs very considerably between *Gill* (top) and RF_d2S (middle), but much less between *Gill* and RF_d2M (bottom).

We conducted robustness testing with different initial population sizes, starting one species with a population size of 10 or 100 and the other with size 1000. Under *Gill*, the simulations showed that for large differences in initial population size, population turnover no longer dominated competitive outcomes, and populations with larger population sizes had the advantage. This effect seemed to be due to the species with larger initial size dominating the grid in early stages. In the standard *Gill* simulation with equal starting population sizes, higher population turnover led to

a larger number of births at the perimeter of populated areas giving the species more rapid access to unoccupied territories where the birth rate was reduced less by competition. This effect was compounded by the resulting faster increase in population size leading to a higher species-level birth rate for the larger population. With the same starting populations, both the discrete time models tended to under-predict the final size of the population with faster turnover (species one) compared with *Gill*; these discrete time models thus performed better when species two had larger initial size since this initial imbalance in population sizes also led to dominance by species two under *Gill*.

We also conducted robustness tests to check values that deviated slightly from $\delta_1 = \delta_2$ by reducing death-to-birth ratios of species one by r_δ of 5%, 10%, 15% and 20% thus giving species one less of an advantage under *Gill*. Although the simulation paradigms

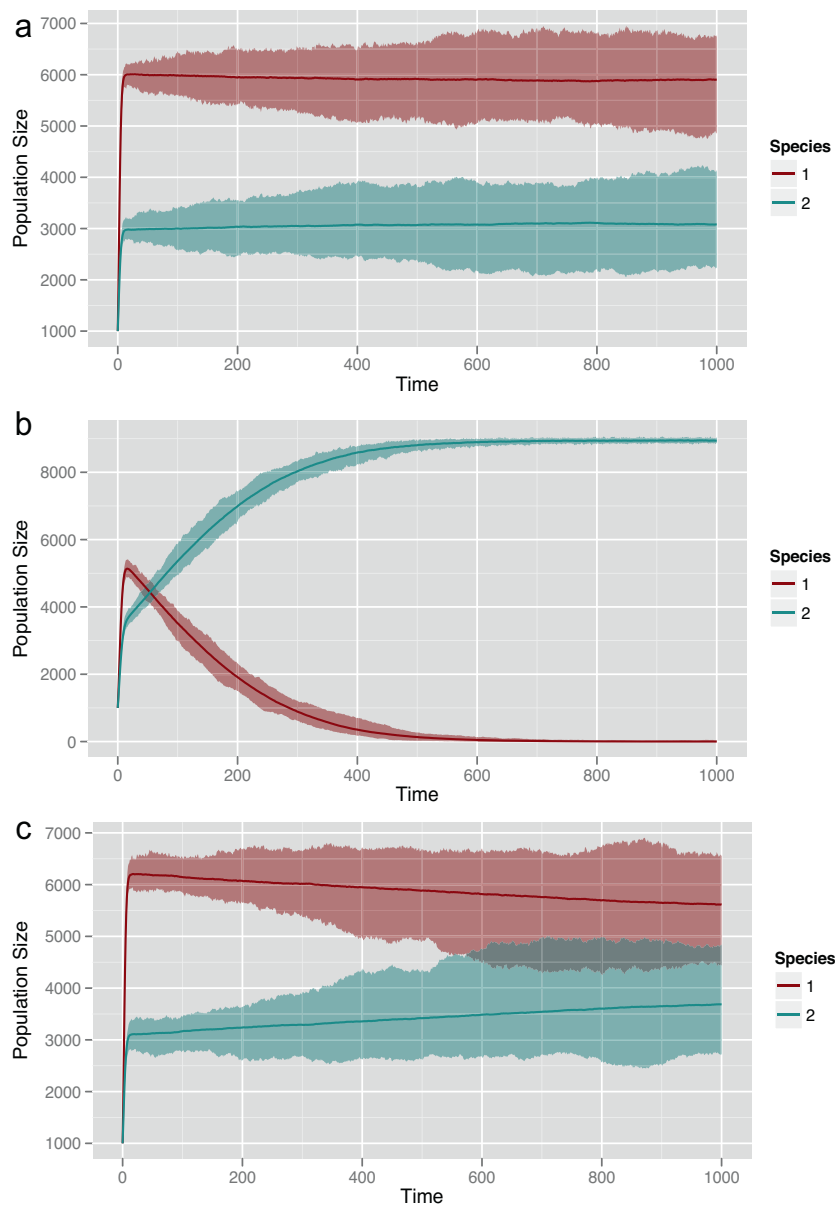


Fig. 7. Time series plots (mean and range) for *Gill*, RF_d2S and RF_d2M , $\delta = 0.1$.

differed less as we moved away from $\delta_1 = \delta_2$, differences were still apparent for RF_d2S for all r_δ for some time steps and values of δ . For RF_d2M , differences were seen for the 5% deviation from equality only. These tests showed that predictions remained sensitive to the update scheme used for death-to-birth ratios deviating from $\delta_1 = \delta_2$.

6. Discussion and conclusions

Our findings illustrate the importance of the decision to model time continuously or discretely, as well as the role of update scheme and step size when using CAs to approximate continuous time models or processes. Experiment 1 showed that in comparison with *Gill*, RF_d2S and $RR1S$ underestimated final population sizes for a single species, while RF_d2M and $RR1M$ gave overestimates. Although simulating multiple births in RF_d2M led to an improvement over the single births version, the same was not true of $RR1M$ which made large overestimates of population sizes. Algorithm bias varied as a function of death-to-birth ratio, and differences were most

apparent at values close to the persistence threshold. Experiment 2 demonstrated differences in interspecific competition as a function of update scheme, step size and death-to-birth ratio. In order to provide the same qualitative prediction as *Gill* about species dominance at time 1000, both RF_d2S and $RR1S$ required small step sizes, although none of the values tested were sufficiently small for $\delta = 0.4$ or $\delta = 0.5$. In contrast, RF_d2M predicted interspecific competition outcomes fairly accurately, with a step size of 2^{-2} being sufficiently small for all δ values.

Our findings also demonstrate the importance of timescale: it is obvious from Fig. 7 that selecting an earlier or later time at which to sample final population sizes would lead to different comparisons between the algorithms from those shown in Fig. 6. The dynamics differ considerably between the algorithms and while more extended runs show that *Gill* has reached a quasi-stationary equilibrium, RF_d2S has reached a steady state with extinction of species one in all runs, and RF_d2M has not yet converged. In *Gill*, species two is more sensitive to extinction due to the smaller population sizes, and in 100 longer runs (not shown), species two first went

extinct around time 33,000 while no populations of species one, the species with higher population turnover, had gone extinct by time 50,000.

Although the importance in ecological modelling of selecting an appropriate CA update scheme has been highlighted previously for single species models (Ruxton and Saravia, 1998), the particular question of differences between continuous and discrete time simulations, and the role of update schemes in the approximation of continuous time processes have not been explored. This is despite the pervasive use of CA models in the literature. Our simulations demonstrate that the decision to model in continuous or discrete time matters, and that choice of CA update scheme affects both final population sizes and the outcomes of interspecific competition. As an approximation of the continuous time approach, the *RR1M* scheme failed to provide accurate predictions for any of the step sizes examined and this scheme should be avoided. More interestingly, step size had important effects on final population sizes and competitive outcomes, and to achieve accurate predictions across the range of values of δ , very small step sizes were required. Doubling step size also doubles execution time, so selecting a step size that is sufficiently small to guarantee accurate approximation to *Gill* requires considerable computational power. We tested our code on two machines,⁷ and although we did not specifically aim to optimise our code, runtime for *Gill* was shorter than all CA update schemes for time steps of 0.5 and smaller, and on the second of the two machines, was shorter than *RF_d2M* for all step sizes. We suggest that whenever continuous time is assumed the Gillespie simulator approach should be selected. As noted in the introduction, this approach has the advantage of reducing the number of modelling decisions, enhancing comparability between studies and allowing researchers to direct their focus towards biological processes rather than technical implementation. Where CAs are nonetheless employed, we recommend that researchers be explicit about both their choice of update scheme and step size, and that they conduct robustness tests to check sensitivity to these parameters.

The differences between discrete and continuous time and the effect of update schemes have not been explored for stochastic simulations of interspecies competition. Our simulations demonstrated that *RF_d2S* and *RR1S* required very small step sizes in order to accurately predict competitive outcomes, and for large step sizes both made large qualitative errors about species dominance for two species with the same death-to-birth ratio. In contrast, *RF_d2M* made relatively accurate predictions regarding species dominance with realistic step sizes. This result is perhaps surprising given the smaller step sizes required in the single species experiment and in light of Caron-Lormier et al.'s (2008) findings that in a continuous space paradigm, differences in predictions became worse for more complex systems. However, the disparity between our findings and those of Caron-Lormier et al. (2008) can probably be explained by the limitations that the space available put on the degrees of freedom in our system.

The finding that the results of interspecific competitions can be so sensitive to modelling assumptions is of concern. Although different modelling paradigms often lead to quantitatively different predictions, we also find that modelling decisions affect qualitative conclusions about the direction of competitive advantage in interspecific competition. When modelling interspecific competition, it is therefore important to choose update schemes and model parameters with care. We recommend that the decision to simulate interspecific competition in discrete or continuous time

should be informed by, and ideally correspond to, the biological processes under consideration. Nonetheless, *RF_d2M* appears to represent a possible approximation for the continuous time lattice logistic model of population growth. The recommendations that researchers should explicitly state the update scheme and step size apply as for single species simulations.

Throughout this paper, we have assumed that CAs are used as an approximation to continuous time models or processes, using *Gill* as a benchmark. However, if we consider both discrete and continuous time models as accurate representations of different real world systems, their predictions can also be interpreted from an ecological perspective. Our findings raise questions about the value to populations of different reproductive strategies. For example, comparing the proportion of runs that went extinct under *Gill* and the two update schemes *RF_d2S* and *RF_d2M* we find that populations of organisms with synchronised reproductive cycles are more sensitive to extinction events than those with continuous reproduction, and for long cycles (large time steps) this is true even when multiple births are possible. In the context of interspecific competition, our simulations showed advantages of slower population turnover in populations of organisms with highly synchronised reproductive cycles whereas faster turnover had the competitive advantage for continuously reproducing species, except near the persistence threshold.

In conclusion, we recommend that researchers exercise caution if using CAs to simulate continuous time processes by checking that their conclusions are not sensitive to the time step chosen, that they justify and explicate their modelling decisions in full with reference to the biological system considered, and ideally that they use continuous time models to simulate continuous processes.

Acknowledgements

We gratefully acknowledge the contribution of Haroon Ahmed in the form of an independent version of the code for algorithm verification, as well as helpful feedback on earlier drafts from Dan Haydon, Hanna Kokko and two anonymous referees. RM is supported by an EPSRC studentship.

Table 2
Symbols used in algorithms.

Symbol	Description	Algorithms
i	Species identifier	All
t, t_{max}	Time, maximum time	All
A, B	Habitat occupancy; index gives site and entry contains species identifier i	All
b_i^r, d_i^r	Per capita birth and death rate for species i	<i>Gill</i>
B_i^r, D_i^r	Total population birth and death rate summed over individuals of species i	<i>Gill</i>
N_i, N	Number of individuals of species i , total number of organisms	<i>Gill</i>
τ	Time until the next event under <i>Gill</i>	<i>Gill</i>
λ	Total event rate	<i>Gill</i>
S	Set of species	<i>Gill</i>
β, ∂	Probability that the next event is a birth, death (summed over all species)	<i>Gill</i>
b_i^p, d_i^p	Per capita probability of birth and death within a given timestep	<i>RF_d2S, RF_d2M, RR1S</i>
Δt	Size of a timestep (fixed value)	<i>RF_d2S, RF_d2M, RR1S</i>
$\mathcal{E}$	List of (site, eventType) pairs	<i>RR1S</i>
$\mathcal{C}$	List of (species, site) pairs for occupied sites	<i>RF_d2S, RF_d2M</i>

⁷ Tests were conducted by running simulations in serial with other CPU load minimised to essential processes on (1) an iMac7,1 Intel Core 2 Duo @ 2.8 GHz with cache size 4 MB running java version 1.6.0_35 and (2) an Intel(R) Xeon(R) CPU E5606 @ 2.13GHz with cache size 8192 KB running java version 1.6.0_24.

Appendix A. Algorithms

The algorithms *Gill*, *RF_d2S*, *RF_d2M* and *RR1S* are explained below using the notation in Table 2. The algorithms are described for multiple species, where the species is denoted by an identifier in the form of the subscript *i*. All algorithms take the following arguments (additional arguments are described in the text of each algorithm): the maximum time t_{max} and habitat occupancy *A*, a data structure in which the index represents site and the entry is either empty or holds the value of *i* for the species living at the site. Superscripts *r* and *p* represent rates and probabilities respectively.

Algorithm A.1. *Gill* algorithm for multiple species.

```

1 Gill( $t_{max}$ , A,  $\mathcal{S}$ ,  $b^r$ ,  $d^r$ , N)
2 begin
3    $t \leftarrow 0.0$ 
4   while  $t \leq t_{max}$  do
5      $\lambda \leftarrow 0.0$ ,  $lwb \leftarrow 0.0$ ,  $p \leftarrow rand()$ 
6     for  $i \in \mathcal{S}$  do
7        $B_i^r \leftarrow b_i^r N_i$ 
8        $D_i^r \leftarrow d_i^r N_i$ 
9        $\lambda \leftarrow \lambda + B_i^r + D_i^r$ 
10    for  $i \in \mathcal{S}$  do
11       $\partial \leftarrow D_i^r / \lambda$ 
12       $\beta \leftarrow B_i^r / \lambda$ 
13      if  $p \leq \partial + lwb$  then doDeath(i, A, N), break
14       $lwb \leftarrow lwb + \partial$ 
15      if  $p \leq \beta + lwb$  then doBirth(i, A, N), break
16       $lwb \leftarrow lwb + \beta$ 
17     $\tau \leftarrow -\ln(rand()) / \lambda$ 
18     $t \leftarrow t + \tau$ 

```

A.1. *Gill*

The *Gill* algorithm (space constrained Gillespie with multiple species) is presented in Algorithm A.1. In addition to t_{max} and *A*, it takes as arguments a set $\mathcal{S}$ of species, two vectors containing the *per capita* birth and death rate for each species b^r and d^r , and a vector *N* containing the total number of organisms of each species currently inhabiting the grid.

The algorithm iterates until the maximum time is reached (outer **while** loop) and has three main sections: lines 6–9 compute the relevant population level rates, lines 10–16 compute the next event and lines 17–18 compute the elapsed time. In lines 6–9 the population level birth and death rates (B^r and D^r) are computed for each species by multiplying per capita rates by the number of organisms, and the total event rate λ is computed as the sum of all event rates. In lines 10–16, species are considered in turn and the probability $\partial \in [0, 1)$ that the next event is a death of the current species is computed by normalising the species rate; the probability $\beta \in [0, 1)$ that the next event is a birth of the current species is calculated analogously. The algorithm then compares ∂ and β to the uniformly random number $p \in [0, 1)$ (generated in line 5), executing a single event if this value plus the current lower bound *lwb* is greater than or equal to *p* (a death at line 13 or a birth at line 15). When an event takes place the **for** loop is exited via the call to **break**. The time to the next event τ is then computed by drawing from an exponential distribution with mean λ (line 17) and time *t* increased by this amount (line 18).

The call to *doDeath*(*i*, *A*, *N*) (line 13) randomly selects a site in *A* occupied by an organism of species *i*, sets this site to be empty and decrements the population count N_i . Similarly, the call to *doBirth*(*i*, *A*, *N*) (line 15) randomly selects a site in *A* occupied by an organism of species *i* and chooses with uniform probability between neighbouring sites; if the chosen site is unoccupied a birth takes place into this site and the population count N_i is incremented.⁸

Algorithm A.2. *RF_d2S* algorithm for multiple species.

```

1 RFd2S( $t_{max}$ , A,  $\Delta_t$ ,  $b^p$ ,  $d^p$ )
2 begin
3    $t \leftarrow 0.0$ 
4   while  $t \leq t_{max}$  do
5      $B \leftarrow A$ 
6      $\mathcal{C} \leftarrow getSites(A)$ 
7     shuffle( $\mathcal{C}$ )
8     for (i, site)  $\in \mathcal{C}$  do
9       if  $d_i^p \geq rand()$  then doDeath(B, site)
10    for (i, site)  $\in \mathcal{C}$  do
11      if  $b_i^p \geq rand()$  then doBirth(i, A, B, site)
12     $t \leftarrow t + \Delta_t$ 
13     $A \leftarrow B$ 

```

A.2. *RF_d2S* algorithm

Algorithm A.2 shows *RF_d2S*. As with *Gill*, this algorithm takes arguments t_{max} and *A*, as well as the time step Δ_t and two vectors containing the *per capita* birth and death probability per time step for each species b^p and d^p . These vectors are calculated from b^r and d^r using the standard conversion from rates to probabilities: $b_i^p = 1 - e^{-\Delta_t b_i^r}$ (and analogously for d^p).

Time progresses in regular steps Δ_t (each constituting a *generation*) until t_{max} (lines 4 and 12). At each generation, *A* is copied into *B* (line 5), then the set of occupied sites $\mathcal{C}$ is computed as a list of (*species*, *site*) pairs and the order of sites is randomised using a Knuth shuffle⁹ (lines 6–7). Lines 8–9 execute death events: for each site in $\mathcal{C}$, a random number is drawn and if this is less than or equal to the probability of death of the species at that site, a death is executed on *B*. Lines 10–11 execute birth events in a similar way.

The overloaded procedures *doDeath* and *doBirth* work as follows. Procedure call *doDeath*(*B*, *site*) (line 9) eliminates the organism at the given site in *B*. In procedure call *doBirth*(*i*, *A*, *B*, *site*) (line 11), if the given site is not occupied by species *i* in *A* then nothing happens; otherwise a neighbouring site σ is selected with uniform probability and if σ is unoccupied in *A* a birth of species *i* takes place into site σ in *B*. On completing a generation *B* is copied into *A* (line 13).

⁸ Note that if the site into which a birth is due to take place is already occupied, the birth event does not take place. The same is true of all the algorithms presented here. Although this approach gives rise to 'wasted' calculations, the alternative would be to represent population structure explicitly and adjust rates according to occupancy; however, this would require updating birth rates at all neighbouring sites every time an event took place and recalculating total birth rate, adding to computational load. In our implementation, because every organism of a species has the same basic birth rate, we can use sets rather than ordered lists to store organisms, reducing the computational load.

⁹ Shuffling the site ordering for death events is actually unnecessary since death events take place independently of one another.

Algorithm A.3. RR1S algorithm for multiple species.

```

1  RR1S( $t_{max}$ ,  $A$ ,  $\Delta_t$ ,  $b^p$ ,  $d^p$ )
2  begin
3     $t \leftarrow 0.0$ 
4    while  $t \leq t_{max}$  do
5       $\mathcal{E} \leftarrow getEvents(A)$ 
6      shuffle( $\mathcal{E}$ )
7      for  $(site, eventType) \in \mathcal{E}$  do
8        if isOccupied( $site$ ,  $A$ ) then
9           $i \leftarrow getSpecies(site, A)$ 
10         if  $eventType = death \wedge d_i^p \geq rand()$  then
11           doDeath( $A$ ,  $site$ )
12         if  $eventType = birth \wedge b_i^p \geq rand()$  then
13           doBirth( $i$ ,  $A$ ,  $A$ ,  $site$ )
14        $t \leftarrow t + \Delta_t$ 

```

A.3. RR1S algorithm

RR1S is presented in Algorithm A.3, and takes the same arguments as RF_d2S . At each Δ_t increment (generation) the set of ($site$, $eventType$) pairs is calculated where $eventType$ is either a birth or a death (line 5). The order of these pairs is randomised (line 6). In lines 7–11, ($site$, $eventType$) pairs are considered in turn. It is possible that a pair ($site$, $birth$) follows a pair ($site$, $death$) and the focal $site$ is then unoccupied; this is tested for in line 8. For an occupied $site$ (lines 9 to 11), if the event is a death then $doDeath$ is called with argument A and affects this data structure directly. If it is a birth event $doBirth$ is called with A twice, so that births take place on A directly.

A.4. Multiple births RF_d2M and RR1M**Algorithm A.4.** RF_d2M algorithm for multiple species.

```

1   $RF_d2M(t_{max}, A, \Delta_t, b^r, d^p)$ 
2  begin
3     $t \leftarrow 0.0$ 
4    while  $t \leq t_{max}$  do
5       $B \leftarrow A$ 
6       $\mathcal{C} \leftarrow getSites(A)$ 
7      shuffle( $\mathcal{C}$ )
8      for  $(i, site) \in \mathcal{C}$  do
9        if  $d_i^p \geq rand()$  then doDeath( $B$ ,  $site$ )
10     for  $(i, site) \in \mathcal{C}$  do
11        $m \leftarrow PoissRand(\Delta_t b_i^r)$ 
12       for  $j \in [1..m]$  do doBirth( $i$ ,  $A$ ,  $B$ ,  $site$ )
13      $t \leftarrow t + \Delta_t$ 
14      $A \leftarrow B$ 

```

The multiple births algorithms RF_d2M and $RR1M$ differ only slightly from the single births version, and therefore only RF_d2M is shown in Algorithm A.4. Firstly, instead of calculating the probability of birth within a timestep, a number of births m is drawn from a Poisson distribution with mean $\Delta_t b_i^r$ (line 11) and correspondingly the algorithms take b_i^r as an argument (in place of b_i^p , line 1). Secondly, $doBirth(i, A, B, site)$ is called m times (where m represents the number of births to attempt and may be zero) at line 12.

Note that Algorithm A.4 differs from Algorithm A.2 only in the way births are performed, i.e. line 11 in Algorithm A.2 performs zero or one birth and lines 11 and 12 in Algorithm A.4 perform zero or many births.

References

- Adler, F., Nuernberger, B., 1994. Persistence in patchy irregular landscapes. *Theoretical Population Biology* 45, 41–75.
- Berec, L., 2002. Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis. *Ecological Modelling* 150, 55–81.
- Best, A., Webb, S., White, A., Boots, M., 2011. Host resistance and coevolution in spatially structured populations. *Proceedings of the Royal Society B: Biological Sciences* 278, 2216–2222.
- Birch, C.P., 2006. Diagonal and orthogonal neighbours in grid-based simulations: Buffon's stick after 200 years. *Ecological Modelling* 192, 637–644.
- Birch, C.P., Oom, S.P., Beecham, J.A., 2007. Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. *Ecological Modelling* 206, 347–359.
- Breckling, B., Pe'er, G., Matsinos, Y.G., 2011. Cellular automata in ecological modelling. In: Jopp, F., Reuter, H., Breckling, B. (Eds.), *Modelling Complex Ecological Dynamics*. Springer, Berlin, Heidelberg, pp. 105–117.
- Caron-Lormier, G., Humphry, R.W., Bohan, D.A., Hawes, C., Thorbeck, P., 2008. Asynchronous and synchronous updating in individual-based models. *Ecological Modelling* 212, 522–527.
- Cornforth, D., Green, D.G., Newth, D., Kirley, M., 2002. Do artificial ants march in step? Ordered asynchronous processes and modularity in biological systems. In: Standish, R., Abbass, H., Bedau, M. (Eds.), *Artificial Life VIII*. MIT Press, pp. 28–32.
- Fleurence, R.L., Hollenbeak, C.S., 2007. Rates and probabilities in economic modelling: transformation, translation and appropriate application. *Pharmacoeconomics* 25, 3–6.
- Fox, R.E., Zhong, X., Krone, S.M., Top, E.M., 2008. Spatial structure and nutrients promote invasion of IncP-1 plasmids in bacterial populations. *The ISME Journal* 2, 1024–1039.
- Gillespie, D.T., 1977. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry* 81, 2340–2361.
- Gillespie, D.T., 2001. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics* 115, 1716.
- Holland, E.P., Aegerter, J.N., Dytham, C., Smith, G.C., 2007. Landscape as a model: the importance of geometry. *PLoS Computational Biology* 3, e200.
- Ingerson, T.E., Buvel, R.L., 1984. Structure in asynchronous cellular automata. *Physica D: Nonlinear Phenomena* 10, 59–68.
- Kamo, M., Boots, M., 2004. The curse of the pharaoh in space: free-living infectious stages and the evolution of virulence in spatially explicit populations. *Journal of Theoretical Biology* 231, 435–441.
- Laird, R.A., Schamp, B.S., 2008. Does local competition increase the coexistence of species in intransitive networks? *Ecology* 89, 237–247.
- Levin, S.A., 1974. Dispersion and population interactions. *The American Naturalist* 108, 207–228.
- Lion, S., 2010. Evolution of reproductive effort in viscous populations: the importance of population dynamics. *Journal of Evolutionary Biology* 23, 866–874.
- Lion, S., van Baalen, M., 2007. From infanticide to parental care: why spatial structure can help adults be good parents. *The American Naturalist* 170, E26–46.
- Lion, S., van Baalen, M., 2008. Self-structuring in spatial evolutionary ecology. *Ecology Letters* 11, 277–295.
- Lumer, E.D., Nicolis, G., 1994. Synchronous versus asynchronous dynamics in spatially distributed systems. *Physica D: Nonlinear Phenomena* 71, 440–452.
- Mangan, S.A., Schnitzer, S.A., Herre, E.A., Mack, K.M.L., Valencia, M.C., Sanchez, E.I., Bever, J.D., 2010. Negative plant-soil feedback predicts tree-species relative abundance in a tropical forest. *Nature* 466, 752–755.
- Manzoni, L., 2012. Asynchronous cellular automata and dynamical properties. *Natural Computing* 11, 269–276.
- Matsuda, H., Ogita, N., Sasaki, A., Sato, K., 1992. Statistical mechanics of population. *Progress of Theoretical Physics* 88, 1035–1049.
- Meier, C.M., Starrfelt, J., Kokko, H., 2011. Mate limitation causes sexes to coevolve towards more similar dispersal kernels. *Oikos* 120, 1459–1468.
- Ovaskainen, O., Hanski, I., 2003. Extinction threshold in metapopulation models. *Annales Zoologici Fennici* 40, 81–97.
- Perfecto, I., Vandermeer, J., 2008. Spatial pattern and ecological process in the coffee agroforestry system. *Ecology* 89, 915–920.
- Roxburgh, S.H., Shea, K., Wilson, J.B., 2004. The intermediate disturbance hypothesis: patch dynamics and mechanisms of species coexistence. *Ecology* 85, 359–371.
- Ruxton, G.D., Saravia, L.A., 1998. The need for biological realism in the updating of cellular automata models. *Ecological Modelling* 107, 105–112.
- Schönfisch, B., de Roos, A., 1999. Synchronous and asynchronous updating in cellular automata. *Biosystems* 51, 123–143.

- Stundzia, A.B., Lumsden, C.J., 1996. Stochastic simulation of coupled reaction–diffusion processes. *Journal of Computational Physics* 127, 196–207.
- Vandermeer, J., Perfecto, I., Philpott, S.M., 2008. Clusters of ant colonies and robust criticality in a tropical agroecosystem. *Nature* 451, 457–459.
- Vandermeer, J., Yitbarek, S., 2012. Self-organized spatial pattern determines biodiversity in spatial competition. *Journal of Theoretical Biology* 300, 48–56.
- White, D., Kiester, A.R., 2008. Topology matters: network topology affects outcomes from community ecology neutral models. *Computers, Environment and Urban Systems* 32, 165–171.