Theoretical Population Biology 95 (2014) 13–23

Contents lists available at ScienceDirect

Theoretical Population Biology

journal homepage: www.elsevier.com/locate/tpb

Coalescent simulation in continuous space: Algorithms for large neighbourhood size

J. Kelleher[a,*], A.M. Etheridge[b], N.H. Barton[c]

[a] *Institute of Evolutionary Biology, University of Edinburgh, Kings Buildings, West Mains Road, Edinburgh EH9 3JT, UK*
[b] *Department of Statistics, University of Oxford, 1 South Parks Road, Oxford OX1 3TG, UK*
[c] *Institute of Science and Technology, Am Campus I, A-3400 Klosterneuburg, Austria*

ARTICLE INFO

Article history:
Received 28 October 2013
Available online 5 June 2014

Keywords:
Isolation by distance
Coalescent simulation

ABSTRACT

Many species have an essentially continuous distribution in space, in which there are no natural divisions between randomly mating subpopulations. Yet, the standard approach to modelling these populations is to impose an arbitrary grid of demes, adjusting deme sizes and migration rates in an attempt to capture the important features of the population. Such indirect methods are required because of the failure of the classical models of isolation by distance, which have been shown to have major technical flaws. A recently introduced model of extinction and recolonisation in two dimensions solves these technical problems, and provides a rigorous technical foundation for the study of populations evolving in a spatial continuum. The coalescent process for this model is simply stated, but direct simulation is very inefficient for large neighbourhood sizes. We present efficient and exact algorithms to simulate this coalescent process for arbitrary sample sizes and numbers of loci, and analyse these algorithms in detail.

1. Introduction

As Wright noted (1978, p. 54), many species, such as the dominant plants in grasslands, have an essentially continuous and uniform distribution in space. In these populations there are no divisions between discrete units, within which mating occurs randomly and between which migrants are exchanged. Despite this obvious observation, the majority of methods available to simulate the history of such populations require this arbitrary structure to be imposed. The typical approach is to use a stepping stone model in which a user must specify a deme size and migration rates between these demes. This approach models many evolutionary scenarios very well, but it can hardly be described as a natural representation of a continuously distributed population.

This arbitrary grid of demes is required because of the lack of a well-defined model to capture the dynamics of continuously distributed populations. The classical model of isolation by distance (Wright, 1943; Malécot, 1948) suffers from severe technical problems, most notably a lack of local density regulation leading to clumps of arbitrarily high density (Felsenstein, 1975). The Wright–Malécot model is also inconsistent with some important biological observations such as large-scale patterns, correlations across loci, and lower diversity than expected from census numbers (Barton et al., 2010b). Etheridge (2008) introduced a model that can describe extinction and recolonisation over a range of scales and provides a simple and effective means of modelling the evolution of a continuously distributed population. In this model, individuals occupy a fixed location in a continuous habitat during their lifetime. All movement, death and reproduction occur as a consequence of replacement events, which fall randomly throughout the habitat. Events may span a range of scales, from the regular process of reproduction within neighbourhoods to large scale demographic shifts, in which substantial fractions of the population are affected. At one extreme, the event may be the death of a small number of individuals, followed by their replacement by a random number of offspring, chosen so as to keep the statistical distribution of the population the same. At the other, most individuals in a wide area might die, again being replaced by new individuals from a smaller number of randomly chosen parents. The model solves the long-standing technical problems mentioned above, and opens the way for mathematically rigorous analytical and inference methods based on a true continuum.

This model of extinction and recolonisation can be viewed as a framework for describing population models, in which different realisations of the framework share a few essential characteristics.

[*] Corresponding author.
E-mail addresses: jerome.kelleher@ed.ac.uk (J. Kelleher), etheridg@stats.ox.ac.uk (A.M. Etheridge), n.barton@ist.ac.at (N.H. Barton).

http://dx.doi.org/10.1016/j.tpb.2014.05.001

A variety of different replacement mechanisms may be employed (Barton et al., 2010b, 2013b), but here we concentrate on the well-studied 'disc' model, in which events are confined to a circular area (Etheridge, 2008; Barton et al., 2010a, 2013a). Suppose that we have a population of individuals distributed uniformly at random on a square torus of diameter L with some density ρ. The evolution of the population is driven by extinction/recolonisation events that occur at rate λ. At an event, we choose a point $\mathbf{z}$ uniformly at random on the torus, and this event only affects the individuals within distance r of $\mathbf{z}$. Within this disc centred on $\mathbf{z}$, a small number of parents ν are chosen uniformly, and each individual has a probability u of dying. (The parameters r and u are often referred to as the radius and impact of an event, respectively.) A Poisson number of children with mean $\rho u \pi r^2$ are then thrown down uniformly within the disc, and the children choose parents uniformly from the pool of ν parents chosen earlier.

Moving to the limit of high density, we then obtain the spatial Λ-Fleming–Viot process, for which many analytical results have been derived (Barton et al., 2013b), and the coalescent process is simple to describe. We begin with a sample of n lineages and proceed backwards in time event-by-event until one of these events overlaps at least one lineage. Then, this lineage has probability u of having been born in this event (since we are proceeding backwards in time); if it was, it jumps to the location of a parental lineage whose location is chosen uniformly at random from the same disc. Eventually, two lineages will be overlapped by the same event and with probability u^2/ν a coalescence event occurs and both lineages jump to the location of their parent. We may also have many lineages descending from a single parent in a event. This process can be formalised as an algorithm very simply (Kelleher et al., 2013); however, such a direct implementation is very inefficient when u is small, because large numbers of events must be simulated. This case of small u (as we see momentarily) corresponds to the biologically important case of large neighbourhood size.

Wright's neighbourhood size $\mathcal{N}$ (1943; 1946) is widely regarded as the most important parameter describing populations evolving in a two-dimensional continuum (see, e.g., Charlesworth and Charlesworth, 2010, pg. 332), and determines the relative rates of genetic drift and gene flow. Wright (1943) identified the approximate magnitudes of the neighbourhood sizes we might expect to observe in natural populations. Small values, with $\mathcal{N} < 100$, correspond to populations with short range dispersal and a high degree of differentiation. Large neighbourhood sizes, with $\mathcal{N} > 10^4$, correspond to populations that are approximately panmictic. Empirical estimates of $\mathcal{N}$ have supported Wright's analysis, with mobile species such as Drosophila having neighbourhood size greater than 10^3 and estimates for plant species being as little as 10 or less (Crawford, 1984). These estimates are also consistent with the range of observed F_{ST} values (Morjan and Rieseberg, 2004).

In the disc model, neighbourhood size is given very simply as $\mathcal{N} = \nu/u$ (Barton et al., 2013a), where ν is the number of potential parents and u is the probability an individual dies in an event. Thus, assuming a single parent and a modest neighbourhood size of 100, we arrive at an impact of $u = 1/100$. Unfortunately, these parameters present serious difficulties to a direct simulation of the coalescent process, in which we expect to generate $1/u$ events that intersect with a lineage before it jumps. For large neighbourhood size, this represents a heavy computational burden.

In this article we develop algorithms to simulate the coalescent process efficiently for large neighbourhood size. The key difference between these new algorithms and previously published methods (Kelleher et al., 2013), is that we no longer explicitly simulate all events and skip the majority of the events that do not affect the state of the simulation. The methods are exact, and we prove that the new algorithms simulate precisely the same stochastic process as before. In Section 2 we begin by defining an algorithm to simulate the important special case of the ancestry of a sample of size two. In this case it is simple to calculate the exact distribution of the time that elapses between events in which a lineage jumps, allowing us to simulate only these events. We then generalise these ideas in Section 3 to allow us to simulate the history of a sample of n lineages, and analyse the resulting algorithm to derive the optimal value for the spatial indexing grid size. Section 4 continues by generalising this algorithm so that we can simulate the history of a sample of individuals with a large number of loci. We describe a general algorithm to distribute genetic material among ancestors over an arbitrary number of loci, and show that this method is efficient when recombination is fast. In particular, we show that the approach is much more efficient than the method used by the classical `ms` program (Hudson, 2002). A Python interface to an efficient implementation of this multilocus algorithm is available at https://pypi.python.org/pypi/discsim under the terms of the GNU General Public License.

In this article we study algorithms in much more detail than is customary in the population genetics literature. We provide detailed and unambiguous algorithm listings in a well established format (Knuth, 1997a, Section 1.1), which is important for several reasons. Firstly, it is impossible to establish the correctness of an algorithm that is stated in an ambiguous natural language format. Given the centrality of coalescent algorithms in modern population genetics and the difficulty of detecting errors in computer programs with stochastic results, it is imperative that we are certain the underlying algorithm is correct. Secondly, since stochastic programs are prone to subtle errors, it is important that a diversity of implementations of a given algorithm exist. If we are dependent on a single (possibly opaque) implementation of a given algorithm, then it is very difficult to verify the results of this program. With a detailed algorithm listing, implementation is routine and allows for multiple independent implementations. Finally, a rigorous description of an algorithm allows us to analyse the properties of this algorithm using mathematical techniques, allowing us to improve performance without resorting to approximations.

2. Pairwise coalescent

Simulating the ancestry of a sample of size two is an important special case, as we are often interested in pairwise statistics. In the extinction/recolonisation continuum model this involves tracing the history of two lineages as they move around the range before meeting and, eventually, coalescing. Events fall uniformly at random across the range, but it is only events that fall within a disc of radius r around the location of at least one of the two lineages that can result in a lineage jumping to a new location. Similarly, the lineages can only coalesce in a given event if the centre of the event falls in the intersection of the discs around these lineages: the centre must be within distance r of both lineages or they cannot both be born in the event.

Remark 1. A lineage is defined by a point $\mathbf{x}$ on the torus, as this is the location of the ancestral individual in question. To simplify the following discussions, however, we refer to a lineage as the disc of radius r around this point. For example, when we refer to the intersection of two lineages this is a shorthand for the intersection of the discs of radius r centred on the locations of the lineages.

The idea behind the pairwise coalescent algorithm is straightforward—we only simulate events that result in a lineage jumping, and calculate the distribution of the time that elapses between these events. Since events fall in a Poisson process with rate λ, we can 'thin' this process to generate only the events that result in jumps. Specifically, if events of a particular type occur with probability p, we know that these events constitute an independent process with rate $p\lambda$. In the pairwise algorithm we must thin the Poisson process twice: once to remove the events whose centres fall outside of the union of the two lineages, and again to remove the events that fall within this area but do not result in a lineage jumping.

To calculate the rate at which at least one lineage jumps, we must calculate the area covered by exactly one lineage, α_1, and the area covered by exactly two lineages, α_2. These are given by the symmetric difference and the intersection of the two lineages, respectively. (The symmetric difference of two sets A and B is $(A \cup B) \setminus (A \cap B)$.) Let $A_r(x)$ denote the area of the intersection of two discs of radius r with centres distance x apart, i.e.,

$$A_r(x) = 2r^2 \arccos\left(\frac{x}{2r}\right) - \frac{x}{2}\sqrt{4r^2 - x^2} \tag{1}$$

for $x < 2r$ and $A_r(x) = 0$ otherwise. We also define

$$u_k^* = 1 - (1-u)^k \tag{2}$$

in the interest of brevity.

Lemma 1. *Given two lineages separated by distance x, the rate at which at least one lineage jumps is*

$$\Omega_2 = \frac{\lambda}{L^2}\left(\alpha_1 u + \alpha_2 u_2^*\right)$$

where $\alpha_2 = A_r(x)$ *and* $\alpha_1 = 2\pi r^2 - 2\alpha_2$.

Proof. Events fall uniformly at rate λ and hit the area covered by exactly one lineage at rate $\lambda\alpha_1/L^2$. A single lineage subsequently jumps with probability u, and so the rate at which jump events fall in the symmetric difference of the two lineages is $\lambda u \alpha_1/L^2$. Similarly, events fall in the intersection of the two lineages at rate $\lambda\alpha_2/L^2$. Then, since there are two lineages that may jump in this region, the probability that at least one jumps is u_2^*. □

Lemma 2. *Conditional on at least one lineage jumping, the probability that the centre of the event is in the intersection of the two lineages is* $1 - \alpha_1/(\alpha_1 - \alpha_2(u-2))$.

Proof. The rate at which events fall in the intersection of the lineages and at least one lineage jumps is $\omega_2 = \lambda\alpha_2 u_2^*/L^2$; therefore, the probability that a given jump event is of this class is ω_2/Ω_2. □

Lemma 3. *Conditional on the centre of a jump event falling in the intersection of the two lineages, the probability that both lineages jump and coalesce is* $u/(2-u)$.

Proof. The probability that both lineages are hit is u^2, and the probability of at least one lineage being hit is u_2^*. Therefore, the probability of both being hit conditional on at least one being hit is $u^2/u_2^* = u/(2-u)$. □

To describe the algorithm concisely, we require some notation. Let $D_r(\mathbf{z})$ define a disc of radius r centred at $\mathbf{z}$ on a two-dimensional torus of diameter L, and let $\|\mathbf{x}\|$ be the Euclidean norm of the vector $\mathbf{x}$ on such a torus (we suppress the dependence on L for simplicity). We also require notation to describe sampling values from random variables drawn from a variety of distributions. Let $\mathcal{R}_\Delta(\xi_1, \ldots, \xi_k)$ define a single independent sample from a random variable with distribution Δ and parameters $\xi_1, \ldots, \xi_k$. (Note that each instance of $\mathcal{R}_\Delta(\xi_1, \ldots, \xi_k)$ within an algorithm listing represents an *independent* random sample from the specified distribution.) Using this notation, we define $\mathcal{R}_U(A)$ to be an element of the set A chosen uniformly at random, and $\mathcal{R}_E(\lambda)$ as a sample from an exponentially distributed random variable with rate λ.

Algorithm P (*Pairwise Coalescent*)**.** Simulate the coalescence time t of two lineages sampled at distance x under a model in which events with radius r and impact u occur at rate λ on a two-dimensional torus of diameter L.

```
P1. [Initialisation.] Set $\mathbf{y}_1 \leftarrow (0,0)$, $\mathbf{y}_2 \leftarrow (0,x)$ and $t \leftarrow 0$.
P2. [Event.] Set $\alpha_2 \leftarrow A_r(\|\mathbf{y}_1 - \mathbf{y}_2\|)$ and $\alpha_1 \leftarrow 2\pi r^2 - 2\alpha_2$. Then, set $\Omega \leftarrow \lambda(\alpha_1 u + \alpha_2 u_2^*)/L^2$ and $t \leftarrow t + \mathcal{R}_E(\Omega)$. If $\mathcal{R}_U([0,1)) < 1/2$, set $j \leftarrow 1$; otherwise set $j \leftarrow 2$. Finally, if $\mathcal{R}_U([0,1)) < 1 - \alpha_1/(\alpha_1 - \alpha_2(u-2))$, go to P4.
P3. [Symmetric difference.] Set $k \leftarrow (j \bmod 2) + 1$. Then set $\mathbf{z} \leftarrow \mathcal{R}_U(D_r(\mathbf{y}_j) \setminus D_r(\mathbf{y}_k))$ and go to P5.
P4. [Intersection.] Set $\mathbf{z} \leftarrow \mathcal{R}_U(D_r(\mathbf{y}_1) \cap D_r(\mathbf{y}_2))$. Then if $\mathcal{R}_U([0,1)) < u/(2-u)$, terminate the algorithm.
P5. [Jump.] Set $\mathbf{y}_j \leftarrow \mathcal{R}_U(D_r(\mathbf{z}))$ and go to P2. ■
```

The algorithm sets up two lineages on the torus separated by distance x and simulates their history jump-by-jump until the lineages coalesce, returning their coalescence time t. In step P2 we first determine the distance between the two lineages and compute α_1 and α_2. We then calculate the rate at which jumps occur using Lemma 1 and increment t accordingly.

After this, we choose a lineage j to jump, and decide if the centre of the event $\mathbf{z}$ is in the symmetric difference of the two lineages or their intersection according to Lemma 2. If the centre falls in the symmetric difference of the two lineages, we generate $\mathbf{z}$ uniformly in $D_r(\mathbf{y}_j) \setminus D_r(\mathbf{y}_k)$ in P3 and then proceed immediately to step P5.

If, on the other hand, $\mathbf{z}$ falls in the intersection of the two lineages we proceed to step P4, where we throw $\mathbf{z}$ down uniformly within this area. Then, according to Lemma 3, both lineages jump with probability $u/(2-u)$ and so a coalescence occurs and we terminate the algorithm. Otherwise, lineage j jumps to a new location in the disc centred on $\mathbf{z}$, and we return to P2.

Given Lemmas 1–3 it is straightforward to show that Algorithm P simulates the coalescent process correctly. In Fig. 2 we compare the results of calculating the probability of identity in state using Algorithm P with numerical estimates (outlined in the Appendix). Under the infinitely many alleles model, the probability of identity in state for two genes, F, is the probability that no mutations have occurred since the lineages diverged. We therefore have $F = \exp(-2\mu t)$ for mutation rate μ and coalescence time t. By taking many replicates we can estimate F from simulations, and we see an excellent agreement between the results of Algorithm P and numerical methods in Fig. 2.

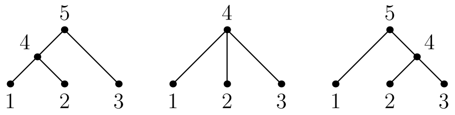
The algorithm is clearly very efficient, since each lineage jump requires a constant number of algorithm steps, and each of these steps is straightforward. Indeed, other than generating points uniformly within the symmetric difference and intersection of the lineages, which can be achieved via standard methods (Knuth, 1997b, Section 3.4), the time required for each jump is constant.

It is not difficult to generalise Algorithm P to simulate several classes of event occurring at different rates. Suppose we have k classes of event, occurring at rate λ_j with radius r_j and impact u_j. To do this, we must modify step P2. In this new step, we first calculate the rate at which successful events are happening for all of these classes by setting $\omega_j \leftarrow \lambda_j(\alpha_{j,1}u_j + \alpha_{j,2}u_{j,2}^*)/L^2$ for $1 \le j \le k$, where $\alpha_{j,2} = A_{r_j}(\|\mathbf{y}_1 - \mathbf{y}_2\|)$ and $\alpha_{j,1} = 2\pi r_j^2 - 2\alpha_{j,2}$, as before. Having calculated the rates at which each of these classes of event are occurring, we increment time by setting $\Omega \leftarrow \omega_1 + \cdots + \omega_k$ and $t \leftarrow t + \mathcal{R}_E(\Omega)$. We then choose an event class j with probability ω_j/Ω, and set $r \leftarrow r_j$ and $u \leftarrow u_j$. After this, we can proceed with P2 as before, choosing the location of the event in either the intersection or symmetric difference of the lineages. The remaining steps of the algorithm are unchanged.

3. Single locus coalescent

Algorithm P provides a satisfactory method to simulate the history of two genes. The basic idea of the algorithm is to partition the torus into regions in which zero, one or two lineages intersect, which can be done quite simply. Although we can generalise the Lemmas 1–3 to n lineages easily, it is not so simple to actually calculate the areas involved. Specifically, calculating the area in which k out of a given n discs intersect is not a trivial problem. We approach the problem instead by calculating these areas approximately using a tessellation of the torus into square pixels and then correct for the errors introduced precisely using rejection sampling.

The output of the algorithm is the simulated history of the sample (π, τ), where π is an oriented tree and τ the corresponding node times. An oriented tree (Knuth, 2011, p. 461) is a sequence $\pi_1\pi_2\dots$, such that π_j is the parent of node j and j is a root if $\pi_j = 0$. For example, the oriented trees



correspond to the sequences 44 550, 4440 and 54 450, respectively. Oriented trees provide an elegant means of describing genealogies, since only parent–child relationships are encoded and the order of children at a node is not important. Oriented trees also have several advantages over more traditional approaches such as nested parentheses. The principal advantage for our purposes here is that we can describe coalescent algorithms in terms of oriented trees concisely and without ambiguity.

3.1. Algorithm N

The pairwise coalescent, Algorithm P, is almost entirely defined by Lemmas 1–3. Although we cannot generalise the algorithm in a direct way because of geometric difficulty, we still require the corresponding results for a sample of n lineages. The following results generalise the pairwise results, and form the basis of the subsequent algorithm.

Lemma 4. *Given a sample of n lineages, the rate at which at least one lineage jumps is*

$$\Omega_n = \frac{\lambda}{L^2}\sum_{k=1}^{n}\alpha_k u_k^*$$

where α_k is the area covered by the intersection of exactly k lineages.

Proof. The centre of an event $\mathbf{z}$ causing a lineage to jump must fall in the intersection of one or more lineages, and an event falls in the intersection of k lineages with probability α_k/L^2. If $\mathbf{z}$ falls in the intersection of k lineages, then at least one lineage jumps with probability u_k^*. Summing over all the possible intersections of lineages, we obtain an overall jump rate of $\lambda\sum_{k=1}^{n}\alpha_k u_k^*/L^2$ as required. □

Lemma 5. *Conditional on at least one lineage jumping, the probability that the centre of the event is in the intersection of exactly k lineages is $\alpha_k u_k^*/\sum_{j=1}^{n}\alpha_j u_j^*$.*

Proof. The rate at which an event falls in the intersection of k lineages and at least one lineage jumps is $\omega_k = \lambda\alpha_k u_k^*/L^2$; therefore, the probability that a given jump event is of this class is ω_k/Ω_n. □

Lemma 6. *Given that $\mathbf{z}$ falls in the intersection of k lineages and that at least one of them jumps, the probability that exactly j jump is*

$$\beta(k, j) = \binom{k}{j}\frac{u^j(1-u)^{k-j}}{u_k^*}.$$

Proof. Exactly j lineages jump with probability $u^j(1-u)^{k-j}$, and there are $\binom{k}{j}$ ways this can happen. Then, since we have conditioned on at least one lineage jumping, we divide by $1-(1-u)^k$, to give us $\beta(k, j)$ as required. □

Calculating the area in which k lineages intersect is nontrivial, but we can use Lemmas 4–6 to derive an efficient simulation algorithm. In this algorithm we proceed by dividing our torus into pixels of edge s, and then use this tessellation to quickly calculate an over-estimate of the area in which k lineages intersect. The errors introduced by this over-estimate can then be precisely cancelled out by discarding potential jump events with a certain probability using rejection sampling.

The key idea of Algorithm N is to divide the range into pixels of edge s; we assume that L/s is an integer so that pixels do not overlap. Let a pixel $\mathbf{v} \in \{1, \dots, L/s\}^2$ define an $s \times s$ square on the torus such that a point $\mathbf{x}$ is in the pixel $\mathbf{v}$ if $0 \le s\mathbf{v}_1 - \mathbf{x}_1 < s$ and $0 \le s\mathbf{v}_2 - \mathbf{x}_2 < s$ (i.e., the top-right corner of pixel $\mathbf{v}$ is $s\mathbf{v}$ in torus coordinates). Let $\mathbb{D}_r^s(\mathbf{z})$ be the set of pixels that intersect with $D_r(\mathbf{z})$, i.e.:

$$\mathbb{D}_r^s(\mathbf{z}) = \{(\lceil \mathbf{x}_1/s\rceil, \lceil \mathbf{x}_2/s\rceil) \mid x \in D_r(\mathbf{z})\}.$$

We also require some further notation to describe sampling from random variables. Firstly, we extend the notation $\mathcal{R}_U(A)$ for sampling an element of a set A uniformly, by letting $\mathcal{R}_U(A, k)$ be a k-subset of A chosen uniformly at random (that is, $\mathcal{R}_U(A, k)$ is a random sample of k elements chosen from A without replacement). Secondly, we let $\mathcal{R}_D(\mathbf{p})$ be a random sample from a discrete distribution with probability mass function defined by the vector $\mathbf{p}$, such that if we set $j \leftarrow \mathcal{R}_D(\mathbf{p})$, then $j = k$ with probability $\mathbf{p}_k$. Also, let $\varnothing$ be the null point.

Algorithm N (*Single Locus Coalescent*)**.** Simulate the ancestry (π, τ) of individuals sampled at locations $\mathbf{x}_1, \dots, \mathbf{x}_n$ under a model in which events with radius r and impact u occur at rate λ on a two-dimensional torus of diameter L using pixel size s, such that $L/s \in \mathbb{N}$.

N1. [Initialisation.] Set $\pi_j \leftarrow 0$, $\tau_j \leftarrow 0$ and $\mathbf{y}_j \leftarrow \varnothing$ for $1 \le j < 2n$ and then set $\kappa \leftarrow n$, $\eta \leftarrow n+1$, $t \leftarrow 0$ and $h^* \leftarrow 0$. Set $P_\mathbf{v} \leftarrow \emptyset$ for $\mathbf{v} \in \{1, \dots, L/s\}^2$ and set $Q_k \leftarrow \emptyset$ for $1 \le k \le n$. Then, for $1 \le j \le n$ set $\mathbf{y}_j \leftarrow \mathbf{x}_j$ and for each $\mathbf{v} \in \mathbb{D}_r^s(\mathbf{x}_j)$, set $P_\mathbf{v} \leftarrow P_\mathbf{v} \cup \{j\}$, $h \leftarrow |P_\mathbf{v}|$, $Q_h \leftarrow Q_h \cup \{\mathbf{v}\}$, $h^* \leftarrow \max(h^*, h)$, and, if $h > 1$, set $Q_{h-1} \leftarrow Q_{h-1} \setminus \{\mathbf{v}\}$.

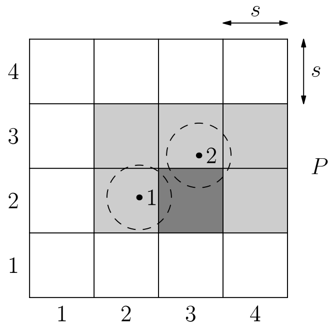
N2. [Rate.] While $|Q_{h^*}| = 0$, set $h^* \leftarrow h^* - 1$. Then, set $w \leftarrow 0$ and for $1 \le j \le h^*$, set $\mathbf{p}_j \leftarrow \left|Q_j\right| u_j^*$ and $w \leftarrow w + \mathbf{p}_j$. Afterwards, set $\mathbf{p}_j \leftarrow \mathbf{p}_j/w$ for $1 \le j \le h^*$.

N3. [Location.] Set $t \leftarrow t + \mathcal{R}_E(w\lambda s^2/L^2)$, $h \leftarrow \mathcal{R}_D(\mathbf{p})$, $\mathbf{v} \leftarrow \mathcal{R}_U(Q_h)$ and $\mathbf{z} \leftarrow s(\mathbf{v} - \mathcal{R}_U([0, 1)^2))$. Then set $S \leftarrow \emptyset$ and, for each $j \in P_\mathbf{v}$, if $\mathbf{z} \in D_r(\mathbf{y}_j)$ set $S \leftarrow S \cup \{j\}$. Finally, if $\mathcal{R}_U([0, 1)) < 1 - u_{|S|}^*/u_h^*$, return to N3.

N4. [Choose children.] Set $\mathbf{b}_j \leftarrow \beta(|S|, j)$ for $1 \le j \le |S|$, then set $j \leftarrow \mathcal{R}_D(\mathbf{b})$ and $C \leftarrow \mathcal{R}_U(S, j)$. Then, if $|C| = 1$, set $k \leftarrow C_1$; otherwise, set $k \leftarrow \eta$.

N5. [Remove children.] For each $j \in C$, and for each $\mathbf{v} \in \mathbb{D}_r^s(\mathbf{y}_j)$ set $h \leftarrow |P_\mathbf{v}|$, $P_\mathbf{v} \leftarrow P_\mathbf{v} \setminus \{j\}$, $Q_h \leftarrow Q_h \setminus \{\mathbf{v}\}$ and, if $h > 1$, set $Q_{h-1} \leftarrow Q_{h-1} \cup \{\mathbf{v}\}$.

N6. [Insert parent.] Set $\mathbf{y}_k \leftarrow \mathcal{R}_U(D_r(\mathbf{z}))$ and then for each $\mathbf{v} \in \mathbb{D}_r^s(\mathbf{y}_k)$ set $P_\mathbf{v} \leftarrow P_\mathbf{v} \cup \{k\}$, $h \leftarrow |P_\mathbf{v}|$, $Q_h \leftarrow Q_h \cup \{\mathbf{v}\}$, $h^* \leftarrow \max(h^*, h)$ and, if $h > 1$, set $Q_{h-1} \leftarrow Q_{h-1} \setminus \{\mathbf{v}\}$. Finally, if $|C| = 1$ go back to N2.



$$P = \begin{pmatrix} \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \{1\} & \{2\} & \{2\} \\ \emptyset & \{1\} & \{1,2\} & \{2\} \\ \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}$$

$$Q_1 = \{(2,2),(2,3),(3,3),(4,2),(4,3)\}$$
$$Q_2 = \{(3,2)\}$$

Fig. 1. Illustration of the indexing structures used in Algorithm N. The torus is divided into pixels of edge s. P is a square array of sets, where each set contains the set of lineages that intersect with the pixel in question. Q_j is the set of pixels with occupancy j.

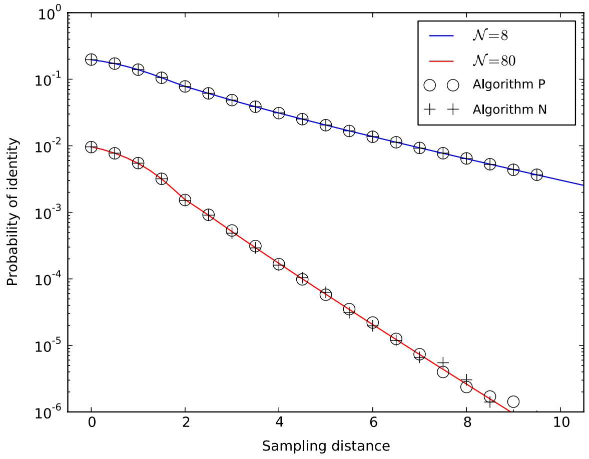


Fig. 2. Comparison of identity in state as calculated using simulations and numerical methods for $L = 100, s = 2, \lambda = 1, r = 1$ and two neighbourhood sizes corresponding to $u = 1/8$ and $u = 1/80$. The mutation rate to new alleles $\mu = 10^{-6}$. In Algorithm P, we take 10^6 replicates for every sampling distance x, and in Algorithm N we take 10^6 replicates of a simulation in which we sample individuals at 100 regularly spaced locations (of which a subset is shown).

N7. [Coalesce.] For each $j \in C$, set $\pi_j \leftarrow \eta$. Then, set $\tau_\eta \leftarrow t$, $\eta \leftarrow \eta + 1$ and $\kappa \leftarrow \kappa - |C| + 1$. Finally, if $\kappa > 1$ go back to N2.

As illustrated in Fig. 1, Algorithm N maintains a spatial index via the matrix P. Each entry in P corresponds to a pixel, and is the set of lineages that intersect with that pixel. The list Q is used to keep track of the pixels with a given occupancy (defined as the number of lineages that intersect with it), such that Q_h is the set of pixels with occupancy h. These data structures serve several purposes. Firstly, Q allows us to quickly determine the number of pixels with a given occupancy when we calculate the rate at which events fall in N2, and then gain access to these pixels when we choose one uniformly. We then choose the centre of the event $\mathbf{z}$ uniformly within the pixel $\mathbf{v}$, and we know that all lineages that can possibly be affected by this event are in the set $P_\mathbf{v}$.

Step N1 initialises the simulation, by first setting up the required data structures and then adding each location $\mathbf{x}_j$ into the spatial indexes P and Q, updating the maximum occupancy h^* as necessary. Once this initialisation is complete, we calculate the rate at which events fall based on the current state of the sample in N2, after first adjusting h^* downwards, if necessary. Once we know the current rate at which jump events may occur, we then increment time according to this rate and choose an occupancy value h. We then select a pixel uniformly from those with occupancy h and choose a point $\mathbf{z}$ uniformly within this pixel. After this we find S, the set of lineages that $\mathbf{z}$ intersects with, and calculate the probability of jumping based on the size of this set. With probability $u^*_{|S|}/u^*_h$ we move on to step N4; otherwise, we return to the start of N3 and generate a new event according to the same rates and probabilities. It is this process of rejection sampling that ensures we simulate precisely the correct process, even though the areas we calculate are approximate.

In step N4, we know that at least one lineage of the set S was born in this event and so we choose the exact number born according to Lemma 6. The variable k is used represent the the parental lineage. If exactly one child is born in the event, then there is no coalescence, and the lineage simply jumps; thus, we set k to be the child lineage (through a slight abuse of notation, where we refer to the first element in the set C as C_1). Otherwise, when more than one child is born in the event, a coalescence occurs, and k is set to the new lineage η.

Step N5 then removes the child lineages from P and Q. Afterwards, in step N6 we choose the location of the parent lineage, and then update P and Q to reflect the insertion of this new lineage, revising the maximum occupancy h^* upwards, as required.

If the set of children C contains more than one lineage, we then proceed on to step N7, where these lineages coalesce and we update the oriented tree and node time structures to reflect this. We set $\pi_j \leftarrow \eta$ for each $j \in C$ to signify that the parent of lineage j is the new lineage η, and also set $\tau_\eta \leftarrow t$ to record the fact that lineage η entered the sample at time t. Finally, we calculate the number of remaining lineages, and if this is greater than 1, return to step N2.

In the previous section we discussed the process by which the pairwise coalescent can be extended to multiple event classes. Unfortunately, it is not so simple to incorporate such events into Algorithm N. To fully generalise the algorithm we must maintain a spatial index for each class of event, and use this to calculate the rates at which events of the various classes are currently happening. While this is fairly straightforward to implement, the memory requirements and the costs of updating the indexes soon become a burden. In the most common case that we wish to model, however, there is a simpler and more efficient approach. If our model consists of small frequent events interspersed with rare large events (Barton et al., 2010a), then we can simply calculate the probability of a large event occurring without conditioning on at least one lineage jumping. Given that such large events are rare, this approach is very efficient since the cost of the large events that we simulate in which no lineages are affected is negligible when amortised over a large number of reproduction events. This is the approach taken in the `discsim` software mentioned in the introduction.

3.2. Correctness

Unlike Algorithm P, it is not immediately obvious that Algorithm N correctly simulates the coalescent process given the rates that lineages jump. The algorithm works by over-estimating the area covered by the intersection of k lineages and then discarding potential jumps with a certain probability, chosen to precisely cancel the error in jump rate that this introduces. The following lemmas prove that this process of rejection sampling results in the correct jump and coalescence rates.

Lemma 7. *Lineages jump at the correct rate in Algorithm N.*

Proof. The rate at which lineages jump as the result of events falling in the intersection of k lineages is $\lambda \alpha_k u^*_k / L^2$. Therefore, we must show that this rate holds for Algorithm N. Suppose we choose a pixel $\mathbf{v}$ with occupancy h. Then, let $\alpha^{\mathbf{v}}_k$ be the area of this pixel covered by the intersection of k lineages and let us calculate the rate at which jumps occur as a result of events falling in the intersection of k lineages. We begin by calculating the value of w based on the current occupancy of the pixels in step N2. We then move on to step N3 and increment time with rate $w\lambda s^2/L^2$. We then choose an occupancy value h with probability $|Q_h|\, u_h^*/w$, and a pixel $\mathbf{v}$ with probability $1/\,|Q_h|$. After throwing $\mathbf{z}$ down uniformly within the pixel $\mathbf{v}$ there is a probability $\alpha_k^{\mathbf{v}}/s^2$ that $\mathbf{z}$ falls in the intersection of k lineages. Finally, with probability u_k^*/u_h^* we move on to step N4 and at least one lineage jumps. Taking the product of the rates of these events we obtain

$$w\left(\frac{\lambda s^2}{L^2}\right)\left(\frac{|Q_h|\, u_h^*}{w}\right)\left(\frac{1}{|Q_h|}\right)\left(\frac{\alpha_k^{\mathbf{v}}}{s^2}\right)\left(\frac{u_k^*}{u_h^*}\right).$$

Cancelling terms, we see that the rate at which an event falls in the intersection of k lineages and a lineage jumps in a pixel of occupancy h is $\lambda\alpha_k^{\mathbf{v}}u_k^*/L^2$. Since pixels are disjoint, we can find the total rate by summing over all pixels with occupancy h and over all occupancy values. This gives us

$$\frac{\lambda u_k^*}{L^2}\sum_{h=1}^{n}\sum_{\mathbf{v}\in Q_h}\alpha_k^{\mathbf{v}} = \lambda\alpha_k u_k^*/L^2$$

as required. □

Lemma 7 proves that the rejection sampling step of N4 correctly adjusts the rate at which lineages jump to account for the approximated areas, resulting in an exact algorithm to simulate the spatial coalescent. The remainder of the algorithm can be easily verified; for example, we can immediately see that step N4 chooses a subset of the available lineages according to Lemma 6.

Fig. 2 shows the probability of identity in state calculated via an implementation of Algorithm N and via numerical methods outlined in the Appendix. There is an excellent agreement between the results. Although identity in state is a pairwise measure, numerical methods still provide a good means of verifying the correctness of the implementation. We begin with a sample of regularly spaced locations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, simulate the history of the sample, and then calculate the probability of identity between the pairs $(\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_1, \mathbf{x}_3), \ldots, (\mathbf{x}_1, \mathbf{x}_n)$. These probabilities are then aggregated over many replicates to obtain an estimate of the mean probability of identity in state over these distances.

3.3. Analysis

The previous subsection assures us that Algorithm N simulates the coalescent process correctly for any pixel size s, but does not give us any indication of what the value of this parameter should be. It is clear that s has an important part to play in the amount of computational effort that is required to simulate the coalescent. If s is too large we will generate many events that miss all of the lineages, and so spend a great deal of time looping around step N3. On the other hand, if s is too small, we spend our time in steps N5 and N6 updating the indexing structures P and Q.

One approach to choosing the value of s is to simply run Algorithm N for a variety of s values and make a choice based on the value which minimises the running time. While this is an effective method, it is unsatisfactory for several reasons. The most obvious problem is that there is no indication of the generality of the results obtained: the optimum value of s may depend on the model parameters, and a great deal of computer time might be invested to explore the effects of L, u and r. A more subtle problem is that this approach can only find the optimum value of s for a given *implementation* of the algorithm. Ideally, each implementation should repeat this time consuming process to determine its own optimal pixel sizes for varying model parameters.

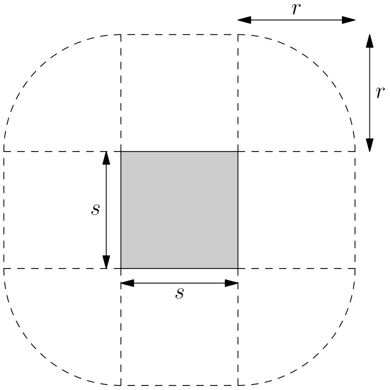


Fig. 3. The catchment area of a pixel is $s^2 + 4rs + \pi r^2$. For a disc of radius r to intersect with the shaded pixel, its centre must fall within the illustrated region.

A much more satisfactory method is to analyse the algorithm, using theoretical methods to estimate the optimum value of s and assess the effects of model parameters on this optimum. In principle, the analysis of an algorithm involves counting the number of primitive operations (such as assignments, arithmetic operations, etc.) incurred during its execution. Using this detailed model of the computational cost in terms of its input parameters, we can then derive the value of s that minimises this function. In practice, however, we are almost never interested in such detailed information, and an approximate broad-brush stroke model of the running time of the algorithm provides all the information that we need.

In this section we apply this methodology to analyse Algorithm N. We derive a simple approximation to quantify the expected computational cost of the algorithm as a function of the pixel size s, which provides us with a useful prediction for its optimum value. The analysis of the algorithm is slightly complicated by the stochastic nature of the coalescent process. The most common means of analysing an algorithm is to derive an expression to approximate its total running time as a function of the input parameters. This would be very difficult in this case, as it would first involve deriving the coalescence time of a sample of size n in the extinction/recolonisation model. We are not interested in the overall running time however, just minimising the cost of simulating the process. Since the basic unit of work in the process is a single lineage jump, we are therefore interested in the value of s that minimises the average computational effort required to effect one lineage jump. Coalescence can be ignored in this analysis, as there are at most $n - 1$ coalescences, and the number of lineage jumps is much, much larger than the sample size.

Throughout this analysis we assume a moderately sized sample of n lineages on a large torus of diameter $L \gg r$, and this torus is tessellated into pixels of edge $s \ll L$ such that L/s is an integer. We also assume a large neighbourhood size, so $u \ll 1$. These assumptions are reasonable, since they reflect the biological reality that we wish to model. The following lemmas provide us with the key quantities for our analysis.

Lemma 8. *Let* $\mathbf{z}$ *be a point chosen uniformly at random on a torus of diameter L tessellated into pixels of edge s. Let* $\sigma_r(s)$ *be the expected number of pixels that a disc of radius r centred at* $\mathbf{z}$ *intersects with. Then,*

$$\sigma_r(s) = \frac{s^2 + 4rs + \pi r^2}{s^2}. \tag{3}$$

Proof. Let $N = (L/s)^2$, and consider the disc of radius r centred on $\mathbf{z}$. The area in which $\mathbf{z}$ can fall such that it intersects with a given pixel is $s^2 + 4rs + \pi r^2$, as shown in Fig. 3. Therefore, the probability that the disc intersects with a given pixel is $(s^2 + 4rs + \pi r^2)/(Ns^2)$ since the total area of the torus is Ns^2. Then, summing this probability over all N pixels gives us the expected number of pixels the disc intersects with, and the required result. □

Lemma 9. *Let* $\mathbf{z}$ *be a point chosen uniformly at random on a torus of diameter* L *tessellated into pixels of edge* s*, and let* V *be the set of pixels that a disc of radius* r *centred at* $\mathbf{z}$ *intersects with. Then, let* $\mathbf{v}$ *be a pixel chosen uniformly from* V *and* $\mathbf{x}$ *be a point chosen uniformly from the pixel* $\mathbf{v}$*. Let* $\psi_r(s)$ *be the probability that* $\mathbf{x}$ *is within the disc* $D_r(z)$*. Then,*

$$\psi_r(s) \geq \frac{\pi r^2}{s^2 + 4rs + \pi r^2}. \tag{4}$$

Proof. The expected number of covered pixels is $\sigma_r(s)$, and therefore the area of these pixels is $s^2\sigma_r(s)$. The area of the disc is πr^2, and therefore the probability that a point falling uniformly at random in a covered pixel is within the disc is $\geq \pi r^2/(s^2\sigma_r(s))$ by Jensen's inequality. □

We analyse Algorithm N by counting the approximate number of basic operations required per lineage jump over a large number of jumps. This gives us an expression $T_N(s)$ that should be proportional to the time required by an implementation to simulate a single jump when averaged over a large number of lineage jumps.

Lemma 10. *Over a large number of lineage jumps, the expected computational cost of a single jump in Algorithm N is*

$$T_N(s, n) \propto \log_2(n\sigma_r(s)) \left(2\sigma_r(s) + \frac{1}{\psi_r(s)}\right). \tag{5}$$

Proof. We proceed by considering the expected contribution of each algorithm step when averaged over a large number of lineage jumps. Over long timescales, lineages are uniformly distributed over the torus, regardless of the initial state of the sample (Barton et al., 2010a, Lemma 6.9). Therefore, the probability that two lineages occupy the same pixel in a given event is negligible and we therefore know that the expectation of the maximum occupancy h^* is equal to 1.

Step N2 is executed once for each jump, and since $\mathbb{E}[h^*] = 1$, the total contribution of this step is constant.

Step N3 loops a certain number of times, and involves non-constant time operations. Since $\mathbb{E}[h^*] = 1$, we know that $\mathbb{E}[|Q_1|] = n\sigma_r(s)$, since there are n lineages covering $\sigma_r(s)$ pixels each. Assuming a balanced tree data structure (Knuth, 1998, Section 6.2.3) for Q_h, each pass through N3 requires $\log_2(n\sigma_r(s))$ time. Since the expected occupancy of a pixel is 1, we therefore have $|S| = 1$ with probability $\psi_r(s)$, and consequently expect to repeat this step $1/\psi_r(s)$ times for each lineage jump. Therefore, we have a contribution of $\log_2(n\sigma_r(s))/\psi_r(s)$ per jump for this step.

Upon reaching N4, the expected value of $|S| = 1$, and this step therefore requires a constant amount of time per lineage jump. In step N5 we remove each lineage in C from the $\sigma_r(s)$ pixels that it covers. Since $\mathbb{E}[h^*] = 1$, the time required to update P_v is constant, and the time required to update Q_h is $\log_2(n\sigma_r(s))$ as before. Then, since $\mathbb{E}[|C|] = 1$, the contribution of this step is $\sigma_r(s)\log_2(n\sigma_r(s))$. Similarly, step N6 adds the parental lineage to $\sigma_r(s)$ pixels, and therefore also requires $\sigma_r(s)\log_2(n\sigma_r(s))$ time.

Step N7 is executed at most $n - 1$ times, and requires a constant number of operations. It does not contribute significantly to the running time over a large number of jumps.

Summing the significant contributions from the steps above gives us (5), as required. □

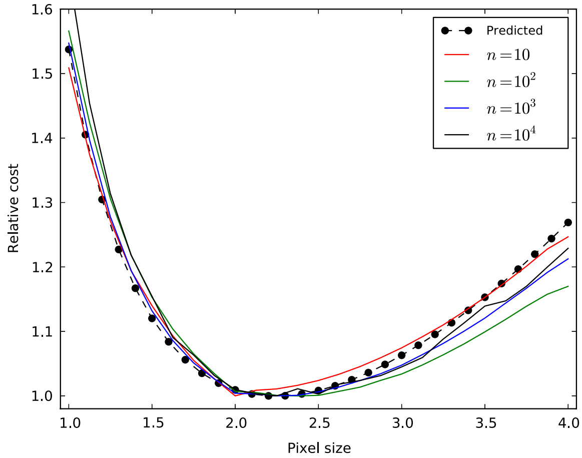


Fig. 4. Comparison of the predicted computational cost of a lineage jump in Algorithm N and the observed cost in a C implementation of the algorithm for varying pixel size s. The cost of a single jump is found by taking the mean CPU time required over 10^8 jumps, using parameters $r = 1$, $u = 1/800$ and $L = s10^4$. To facilitate comparison, the costs reported are relative to the minimum cost; for example, the predicted value is obtained by calculating $T_N(s)/T_N(2.24)$.

Lemma 10 provides us with a prediction for the expected time required to perform a single lineage jump in Algorithm N in terms of the sample size n. It is more useful, however, to consider the limit of large sample sizes, and to derive a prediction that is independent of n. In general, we expect $s > r$, and so $\sigma_r(s)$ is between 1 and 10; hence, for realistic sample sizes, the dominant term is $\log_2(n)(2\sigma_r(s) + 1/\psi_r(s))$. As a result, the optimal pixel size is approximately independent of n and can be found by minimising $2\sigma_r(s) + 1/\psi_r(s)$. Performing this minimisation numerically we find a minimum at $s \approx 2.24r$, providing us with a useful prediction for the optimal pixel size in Algorithm N. Fig. 4 shows an excellent agreement between the predicted and observed optimal pixel size over a wide range of sample sizes.

4. Multilocus coalescent

There are two practical approaches to including the effects of recombination in the coalescent algorithm. The first, pioneered by Hudson (1983), is to track the state of m-locus individuals and build m genealogies as we proceed backwards in time, generating common ancestor and recombination events and applying the effects to the ancestral population. Recombination events increase the size of the ancestral population, but do not increase the amount of ancestral material; common ancestor events reduce the size of the ancestral population, and potentially reduce the amount of ancestral material present through coalescence. Coalescence occurs when two or more ancestors trace back to a single parent, and both have ancestral material at one or more loci. Each time a coalescence occurs we update the genealogy at the loci involved, and the simulation terminates when all genealogies are complete. This approach is the basis of the classical `ms` program (Hudson, 2002).

The second approach, introduced by Wiuf and Hein (1999), involves first generating the genealogy for the left-most locus and then moving rightwards along the sequence, generating recombination breakpoints. At each breakpoint the current genealogy is modified, and the algorithm terminates when the rightmost point of the sequence has been reached. Wiuf and Hein's algorithm is considerably more complicated than Hudson's, but the time complexity of the methods is similar (Wiuf and Hein, 1999). The approach of working along the genome from left to right, however, has led to an approximation that makes simulating the history of large genomic regions much more efficient. In the sequentially Markov coalescent (McVean and Cardin, 2005), the coalescent with recombination is approximated by assuming that the genealogy at a breakpoint depends only on the immediately previous genealogy. This approximation has been used and extended in many simulation algorithms (Marjoram and Wall, 2006; Chen et al., 2009; Excoffier and Foll, 2011), as the time and space requirements of simulating the history of a sample are greatly reduced by this simplification. We do not use the sequentially Markov coalescent here, as the necessary theory to extend the method to a two-dimensional continuum has not been developed.

In this section we extend the single locus coalescent algorithm of Section 3 by adapting Hudson's approach to the setting of a spatial continuum. We incorporate recombination by letting each individual in the sample consist of m linearly arranged loci, and extending the model such that we have two parents at each event. At an event, there is a probability ρ that a recombination event occurs between locus ℓ and $\ell+1$, and a child derives from different parents at these loci. See Etheridge and Véber (2013) and Barton et al. (2013a) for more details on this model of recombination.

In this section we describe Algorithm M, the extension of Algorithm N to simulate the ancestry of m-locus individuals. The algorithms share the same overall structure, and, besides the process of transferring ancestral material from offspring to parents, do not differ in significant ways. Rather than writing out a full listing of Algorithm M, which would obscure the key points and result in needless notational complexity, we concentrate on describing the differences between the two algorithms. The first change we require is to generalise the data structures used to track the history of m-locus individuals. We also require an algorithm to transfer the ancestral material from children to parents under the effects of recombination, recording any coalescences that occur. The latter part is far more complex, and we therefore examine the process in detail.

Algorithm M operates in the same manner as N, where we have n individuals sampled on the torus and simulate until the history of the sample is complete. Each locus ℓ has an oriented tree π_ℓ and list of node times τ_ℓ. We also have a sequence η such that η_ℓ is the next available node in the oriented tree π_ℓ. The state of the ancestral population is most simply represented as set of tuples $(\mathbf{x}, a_1 \ldots a_m)$, where $\mathbf{x}$ is the location of an individual and the sequence $a_1 \ldots a_m$ records the node the individual occupies in the genealogy at each locus. If $a_\ell = 0$ (i.e., the null node), then there is no ancestral material present in this individual at locus ℓ, and so we are not interested in tracking its history. Termination of the algorithm is controlled by letting κ track the total amount of ancestral material in the sample; initially, $\kappa = nm$, and we know that all loci have coalesced when $\kappa = n$.

The same method of spatial indexing via the pixels P and occupancy Q applies in Algorithm M, and so steps N2–N5 are essentially unchanged. Once we have selected the set of children, however, we must decide how the ancestral material of these individuals is distributed among the parents of the event, taking into account the effects of possible recombination and coalescence. This process is described in detail in Algorithm G. Afterwards, we have two parents with the ancestral material from the children shared among them via their node mappings, and for each parent that has ancestral material we insert it into the sample using a similar method to N6. Parents without ancestral material are discarded.

For notational convenience, we let a be a matrix of node assignments for the children of an event, such that $a_{j,\ell}$ is the node that the jth child occupies in the genealogy at locus ℓ. Similarly, we let p be the matrix of node assignments for the parents such that $p_{k,\ell}$ is the node that the kth parent occupies in the genealogy at the ℓth locus. Initially, parents have nonancestral material at every locus (and so $p_{k,\ell} = 0$ for all k and ℓ). Then, as the node assignments in a are distributed among the parents, taking into account the effects of coalescence and recombination, some of these loci are assigned ancestral material.

The purpose of Algorithm G is to provide an explicit method by which recombination can be incorporated into the spatial coalescent simulation of the previous section. It is a straightforward approach and can be readily adapted to more general recombination models (for example, allowing an arbitrary number of parents ν or a vector of per-locus recombination rates). The algorithm assumes that the set of child individuals has already been chosen, and then generates the parent individuals, taking into account the effects of coalescence and recombination.

Algorithm G (*Generate Parents*)**.** Given the matrix of node mappings of n child individuals a, generate the node mappings of two parental individuals p, and update π, τ, η and κ to record coalescence events. Recombination occurs with probability ρ between adjacent loci.

```
G1. [Initialisation.] Set $p_{j,\ell} \leftarrow 0$ for $1 \le j \le 2$ and $1 \le \ell \le m$.
    Then set $c_\ell \leftarrow 0$ for $1 \le \ell \le m$, and $j \leftarrow 1$.
G2. [Choose first parent] Set $k \leftarrow \mathcal{R}_U(\{1, 2\})$ and set $\ell \leftarrow 1$.
G3. [Ancestral?] Set $\alpha \leftarrow a_{j,\ell}$. If $\alpha = 0$ go to G7.
G4. [Inheritance.] If $p_{k,\ell} = 0$, set $p_{k,\ell} \leftarrow \alpha$ and go to G7. Otherwise,
    if $c_\ell \ne 0$, go to G6.
G5. [Single coalescence] Set $c_\ell \leftarrow 1$, $\beta \leftarrow p_{k,\ell}$ and $\gamma \leftarrow \eta_\ell$. Then
    set $\pi_{\ell,\beta} \leftarrow \gamma$, $\tau_{\ell,\gamma} \leftarrow t$, $p_{k,\ell} \leftarrow \gamma$ and $\eta_\ell \leftarrow \gamma + 1$.
G6. [Multiple coalescence.] Set $\pi_{\ell,\alpha} \leftarrow \eta_\ell - 1$ and $\kappa \leftarrow \kappa - 1$.
G7. [Next locus.] If $\mathcal{R}_U([0, 1)) < \rho$, set $k \leftarrow \mathcal{R}_U(\{1, 2\} \setminus \{k\})$. Then
    set $\ell \leftarrow \ell + 1$ and if $\ell \le m$ go to G3.
G8. [Next child.] Set $j \leftarrow j + 1$, and if $j \le n$ go to G2.
```

Algorithm G proceeds by considering each child j and each locus ℓ in turn. If child j has ancestral material at locus ℓ we transfer this ancestral material to a parent. Each time such an assignment is made we test for coalescence, which occurs when more than one child descends from a particular parent at a given locus, and update the data structures to record this event.

For each child j we first choose a parent uniformly at random for the first locus in step G2. We then consider each locus $1 \le \ell \le m$ in turn; if $a_{j,\ell} \ne 0$ then there is ancestral material and we proceed on to G4. In G4 we test to see if there has already been ancestral material assigned to this parent k at locus ℓ. If this is the case, then a coalescence has occurred at this locus, and we must record this fact. Because we can have more than two children in an event that have ancestral material at a given locus, there may be several children descending from a given parent. The first time we encounter a coalescence at a locus, we update π, τ and η to register this event in G5. For subsequent coalescence events at this locus we skip directly to G6, where we record that the parent of node α at locus ℓ is $\eta_\ell - 1$ and decrement κ, accounting for the loss of one piece of ancestral material. Recombination between adjacent loci occurs in step G7, where we choose a new parent for locus $\ell + 1$ with probability ρ.

Algorithm G is a simple and effective method of transferring ancestral material from children to parents for small numbers of loci ($m < 10$, say). It requires only one pass through the set of children and transfers ancestral material directly from children to parents, without requiring any intermediate data structures. It is, however, extremely inefficient for large m, wasting large amounts of time and space by storing and iterating over the large tracts of non-ancestral material that enter the sample as we progress backwards in time.

This inefficiency can be easily resolved by changing the representation of the node mappings for an individual from the 'dense' sequence $a_1 \ldots a_m$ to a sequence of pairs (ℓ, α) mapping loci to nonzero nodes. The first benefit of this 'sparse' representation is that the amount of memory required to store the state of the ancestral population is far smaller. Using the dense representation the amount of memory required grows with the size of the ancestral population. Since each individual requires $O(m)$ space to store its node mappings and the ancestral population can grow up to a limit of nm, we require $O(nm^2)$ space. For large m, this is prohibitive. The sparse representation, on the other hand, requires $O(nm)$ space, since the amount of ancestral material in the sample is nonincreasing as we proceed backwards in time.

The second benefit of storing pairs that map loci to non-null tree nodes is that the time required to distribute ancestral material from children to parents is greatly reduced. Algorithm G requires $O(m)$ time, regardless of the distribution of ancestral material among children. This behaviour is particularly poor for large m, since even moderate recombination rates result in a rapidly growing ancestral population, implying that the majority of loci in a randomly chosen individual do not contain ancestral material. We therefore spend the majority of our time looping between G3 and G7 doing nothing other than choosing parents for loci with nonancestral material. This can be greatly improved using the sparse representation, and we outline the ways to modify Algorithm G to generate parents under this approach.

Assuming that the mapping pairs (ℓ, α) are sorted in increasing order of locus number, we begin by choosing the parent k to be a uniformly distributed element of $\{1, 2\}$ as before. We then examine the first locus pair (ℓ, α) and assign the ancestral material α to locus ℓ of parent k. (Note that the parent for the first locus with ancestral material ℓ is still uniformly distributed, even if $\ell > 1$.) We then examine the next pair (ℓ', α'), and calculate the probability that a recombination event has occurred given the gap between the two loci $\ell' - \ell$. If recombination does occur, we choose a new parent as before, and then move on to the next pair. Since the gaps between loci with ancestral material may be arbitrarily large, it is important that we can calculate the probability of recombination without iterating over the intervening loci. The following lemma allows us to accomplish this. (The lemma is expressed in terms of a general number of parents ν; we are usually interested in the biparental case where $\nu = 2$.)

Lemma 11. *Given a gap of length k between two loci with ancestral material in a system with ν parents and a probability ρ of recombination between adjacent loci, the probability that they descend from different parents is*

$$\phi(k) = \frac{\nu - 1}{\nu}\left(1 - \left(1 - \frac{\nu\rho}{\nu - 1}\right)^k\right). \tag{6}$$

Proof. Consider the following analogue of the recombination process over a gap of length k. Suppose we set $a_0 \leftarrow 1$ and then for each $1 \le j \le k$ set $a_j \leftarrow \mathcal{R}_U(\{1, \dots, \nu\} \setminus \{a_{j-1}\})$ with probability ρ or set $a_j \leftarrow a_{j-1}$ otherwise. Let $\phi(k)$ be the probability that $a_k \neq 1$. Clearly, $\phi(1) = \rho$. Suppose that $\phi(k-1)$ is the probability that $a_{k-1} \neq 1$ and consider $\phi(k)$.

Three possibilities exist in which $a_k \neq 1$: $a_{k-1} = 1$ and recombination occurs; $a_{k-1} \neq 1$ and recombination does not occur; or, $a_{k-1} \neq 1$ and we recombine to a value not equal to 1. Writing these probabilities down, we have

$$\begin{aligned}\phi(k) &= \rho\,(1 - \phi(k-1)) \\ &\quad + (1-\rho)\phi(k-1) + \rho\left(1 - \frac{1}{\nu - 1}\right)\phi(k-1)\end{aligned}$$

with $\phi(1) = \rho$. Solving this recurrence gives us (6), as required. □

Using Lemma 11 we can avoid looping over nonancestral material to determine the parent of the next locus with ancestral material. In this way we can visit each piece of ancestral material in turn, assigning it to the correct parent and updating the data structures to account for coalescence events, as required. Therefore, the overall cost of the process of transferring ancestral material from descendants to ancestors in an event is proportional to the amount of ancestral material that the children carry. Since this is quite small when recombination is fast, the algorithm is an effective and efficient method of transferring ancestral material under these conditions.

This approach may seem wasteful, however, when modelling lower levels of recombination, where we expect to see many adjacent loci sharing the same ancestry. In this case, storing the ancestry for each locus in each individual and storing a tree for each locus is unnecessary. A more compact approach is to store a tuple holding the first and the last locus of each segment, and to only store trees for each unique genealogy. Unfortunately, this method has some major pitfalls. Without the use of specialised data structures, maintaining the segments and calculating overlaps becomes a serious burden and negates any advantages of a more compact representation.

The `ms` program (Hudson, 2002) uses this segment approach. It maintains a list of segments representing the distinct genealogies that have been created through recombination, and each ancestor maintains a list of segments mapping their ancestral material to these trees. Each recombination event results in a new segment being created, and each common ancestor event conducts a linear scan of these segments to detect overlaps between the segments the individuals carry and the segments defining the extant trees.

The expected number of recombination events within ancestral material in a simulation of the standard coalescent with a sample of size n is RH_{n-1} (Hudson and Kaplan, 1985), where H_n is the nth Harmonic number and $R = 4N_e\rho(m-1)$ is the scaled recombination rate. (N_e is the effective population size, m the number of loci and ρ the between-locus recombination probability.) Since $H_n \approx \ln n + \gamma$, where γ is the Euler-Mascheroni constant, the expected number of segments is approximately R for large R and small n.

Therefore, in the worst case, each common ancestor event in `ms` costs $O(R)$ time. For large R, this represents an extremely heavy cost. If we follow the example of Chen et al. (2009) and assume $N_e = 12\,500$ and $\rho = 1.2 \times 10^{-8}$, we have a scaled recombination rate of $R = 12\,000$ for a 20 Mb region. Profiling `ms` under these parameters with a sample of two individuals reveals that over 90% of the simulation time is spent performing common ancestor events, compared to around 2% of the time executing recombination events. Since the number of these events is similar (otherwise the simulation would quickly end, or never finish), this demonstrates a major flaw in `ms`, at least in the case of large R and m.

A thorough analysis of the different approaches to maintaining ancestry in coalescent simulations is beyond the scope of this article. The comparison with `ms` is intended to illustrate that, although our approach is simple, it is effective when the expected amount of ancestral material per ancestor is small. For low levels of recombination, the `ms` approach is very efficient. It is not clear what type of approach is suitable for large genomic regions with intermediate levels of recombination. This is an important question for the efficiency of coalescent simulation.

Ideally, we would like to extend the analysis of Section 3.3 and derive the optimal pixel size s for a given neighbourhood size and recombination rate. Such an analysis, however, would require a detailed understanding of the spatial distribution of ancestors and the ancestral material they carry. Since very little is currently known about these distributions, we must defer a full analysis of the multilocus algorithm to future work. Before this analysis is performed, we can make some basic recommendations about the choice of pixel size for multilocus simulations. Firstly, the optimal value for s derived for the single locus case in Section 3.3 is certainly an upper bound on the value of s that should be used for multilocus simulations. Any simulations in which we expect many ancestors to be in close proximity for extended periods of time should have $s < 2.24r$. On the other extreme, for $s < 1$ memory requirements for maintaining the spatial indexes increase sharply. In this case, any potential advantages of faster simulation may be overwhelmed by excessive memory usage, preventing replication over available CPU resources.

5. Conclusion

A central goal of the spatial Λ-Fleming–Viot continuum model is to provide a means of incorporating events over different scales, ranging from the steady process of local reproduction to large-scale demographic shifts. Without such events, we cannot explain the patterns that we observe in nature over large spatial scales since the process of diffusion is too slow (Barton et al., 2010b). The continuum model incorporates these fluctuations in a flexible and elegant manner, by allowing events of different radii and impacts to occur at different rates. For example, the effect of rare large events modelling demographic shifts along with frequent small events modelling reproduction is analysed in detail by Barton et al. (2010a).

In the interest of simplicity we have given detailed algorithm listings for a single class of event occurring at a fixed rate, and outlined the changes required to generalise to an arbitrary number of event classes from the disc model. Incorporating events from different models, however, is not so straightforward, as our methods here depend entirely on the geometry of the disc model. Including events from the Gaussian replacement model (Barton et al., 2010b), for example, would require a different approach. This is not a major drawback, however. It is only the most frequent reproduction events that must be from the disc model; rarer events can have any geometry that we wish, provided we follow the method outlined in Section 3.1 for incorporating large-scale events. Since these events are rare with respect to regular reproduction, there is little point in conditioning on the events affecting individuals and is in fact more efficient to simulate their effects directly.

We have discussed and analysed the algorithms in this article in more detail than is customary in the literature. Given the centrality of coalescent simulation in modern population genetics, it is important that the algorithms used are both correct and as efficient as possible, and this can only be achieved via detailed algorithm listings and analyses. The consequences of hiding these important details are well illustrated by the `ms` program. As our brief analysis in Section 4 revealed, `ms` is very efficient for small numbers of loci, but scales extremely poorly for large genomic regions. However, this fact is not widely known and `ms` is usually regarded as being very efficient (Carvajal-Rodríguez, 2008; Chen et al., 2009). The performance of `ms` is in effect taken to be a measure of the inherent difficulty of simulating the coalescent with recombination. Consequently, researchers have abandoned the full coalescent and resorted to various approximations (Liang et al., 2007; Padhukasahasram et al., 2008; McVean and Cardin, 2005; Chen et al., 2009) in order to simulate larger genomic regions. An analysis of the algorithm and the use of the appropriate data structures would make simulating the full coalescent many times faster than is possible with `ms` without the need for approximation.

Substantial efforts have been made in this article to avoid approximation and to ensure that the process that we simulate is precisely equal to the well-defined coalescent process of the continuum model. A sceptical reader might argue that this is wasted effort, since some approximation to a model which is (at best) a cartoon of reality can do little harm. However, this ignores one of the most important applications of stochastic simulations: assessing the accuracy of analytical methods. Without exact simulations, it is very difficult to measure the accuracy of putative analytical approximations; deviations from simulation results may be due to either approximations of the model in simulations or to errors in the approximate calculations. Exact simulations also allow us to be confident that the data we observe are a result of interesting phenomena arising from the underlying model, and not some unexpected consequence of an ad-hoc approximation.

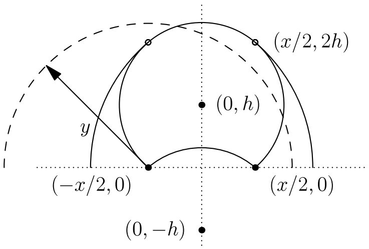
Acknowledgments

We would like to thank Amandine Véber for discussions and comments on the manuscript.

The first author's work was supported by EPSRC grant EP/I013091/1. The second author's work was supported in part by EPSRC grant EP/I01361X/1. The third author's work was supported by European Research Council grant 250152.

Appendix. Calculating identity

Let $A_r(x, \mathbf{z})$ be the area of the intersection of three discs of radius r, centred at $(-x/2, 0)$, $(x/2, 0)$ and $\mathbf{z}$, and let $h = \sqrt{r^2 - x^2/4}$. If we consider $\mathbf{z}$ in the upper half plane, the domain of $A_r(x, \mathbf{z})$ is the union of four sets, indicated by the solid arcs:



. (A.1)

If $\mathbf{z}$ is in the region bounded by the x-axis and the arc of radius r centred at $(0, -h)$, $A_r(x, \mathbf{z})$ is constant and equal to $A_r(x)$. Otherwise, if $\mathbf{z}$ falls in the region bounded by the arc of radius r centred on $(0, h)$, $A_r(x, \mathbf{z})$ is the area of a circular triangle (Fewell, 2006). Finally, if $\mathbf{z}$ falls within the region bounded by the arc of radius $2r$ centred on $(-x/2, 0)$, $A_r(x, \mathbf{z}) = A_r(\sqrt{(z_1 + x/2)^2 + z_2^2})$; a similar rule applies to the region bounded by the arc centred on $(x/2, 0)$. For all $\mathbf{z}$ outside of these regions, $A_r(x, \mathbf{z}) = 0$.

Let $F_\mu(x)$ be the probability of identity in state of two genes sampled at distance x, under the infinitely many alleles model with mutation at rate μ (Barton et al., 2010b, 2013a). After rephrasing Eq. (A.8) of (Barton et al., 2013a) in terms of the A_r functions, converting to polar coordinates and simplifying, we obtain

$$\phi(x)F_\mu(x) = \frac{u^2}{\nu}A_r(x) + \int_0^\infty K(x, y)F_\mu(y)\, dy \tag{A.2}$$

where $\phi(x) = 2\mu/\Lambda + 2u\pi r^2 - u^2 A_r(x)$, the kernel $K(x, y)$ is given by

$$K(x, y) = f\left(\frac{y}{r}\right)\left(1 - \frac{1}{\nu}\right)\frac{A_r(x)}{r} + \frac{4uy}{\pi r^2}\int_0^\pi Q_1(x, y, \theta) - uQ_2(x, y, \theta)\, d\theta,$$

with

$$Q_1(x, y, \theta) = A_r\left(\sqrt{x^2 - 2xy\cos\theta + y^2}\right)$$
$$Q_2(x, y, \theta) = A_r\left(x, (-x/2 + y\cos\theta, y\sin\theta)\right).$$

Here, $f(x)$ is the probability density function of the distance between two points sampled independently and uniformly at random within the unit disc (Alagar, 1976),

$$f(x) = \frac{x}{\pi}\left(4\arccos(x/2) - \sqrt{4-x^2}\right), \quad x \in [0, 2],$$

with $f(x) = 0$ for $x > 2$. The value of $\phi(x)$ is found by observing that $\int_{\mathbb{R}^2} A_r(|\mathbf{z}|)\, d\mathbf{z} = (\pi r^2)^2$ and $\int_{\mathbb{R}^2} A_r(x, \mathbf{z})\, d\mathbf{z} = \pi r^2 A_r(x)$.

Eq. (A.2) is a Fredholm equation of the second kind, and can be solved numerically using a number of methods (Atkinson, 1997), with the Nyström method being most convenient. Solving the equation in this manner requires the repeated evaluation of $\int_0^\pi Q_1(x, y, \theta) - uQ_2(x, y, \theta)\, d\theta$, which can be time consuming and inaccurate if not done with care. In particular, integrating $Q_2(x, y, \theta)$ can be troublesome. Performing the integration from 0 to π piecewise, however, as the arc of radius y centred at $(-x/2, 0)$ intersects with the arcs defining the domain of $A_r(x, \mathbf{z})$, as indicated in (A.1), gives us a fast and accurate numerical method.

References

Alagar, V., 1976. The distribution of the distance between random points. J. Appl. Probab. 13, 558–566.

Atkinson, K., 1997. The Numerical Solution of Integral Equations of the Second Kind. Cambridge University Press.

Barton, N.H., Etheridge, A.M., Kelleher, J., Véber, A., 2013a. Inference in two dimensions: allele frequencies versus lengths of shared blocks. Theor. Popul. Biol. 87, 105–119.

Barton, N.H., Etheridge, A.M., Véber, A., 2010a. A new model for evolution in a spatial continuum. Electron. J. Probab. 15, 7.

Barton, N.H., Etheridge, A.M., Véber, A., 2013b. Modelling evolution in a spatial continuum. J. Stat. Mech. P01002.

Barton, N.H., Kelleher, J., Etheridge, A.M., 2010b. A new model for extinction and recolonisation in two dimensions: quantifying phylogeography. Evolution 64 (9), 2701–2715.

Carvajal-Rodríguez, A., 2008. Simulation of genomes: a review. Curr. Genomics 9 (3), 155–159.

Charlesworth, B., Charlesworth, D., 2010. Elements of Evolutionary Genetics. Roberts and Company, Greenwood Village, Colorado.

Chen, G.K., Marjoram, P., Wall, J.D., 2009. Fast and flexible simulation of DNA sequence data. Genome Res. 19, 136–142.

Crawford, T.J., 1984. What is a population? In: Shorrocks, B. (Ed.), Evolutionary Ecology. Blackwell Scientific Publications.

Etheridge, A.M., 2008. Drift, Draft and Structure: Some Mathematical Models of Evolution, 80. Banach Center Publ., pp. 121–144.

Etheridge, A.M., Véber, A., 2013. The spatial Λ-Fleming–Viot process on a large torus: genealogies in the presence of recombination. Ann. Appl. Probab. 22 (6), 2165–2209.

Excoffier, L., Foll, M., 2011. Fastsimcoal: a continuous-time coalescent simulator of genomic diversity under arbitrarily complex evolutionary scenarios. Bioinformatics 27 (9), 1332–1334.

Felsenstein, J., 1975. A pain in the torus: some difficulties with the model of isolation by distance. Am. Nat. 109, 359–368.

Fewell, M., 2006. Area of common overlap of three circles. Australian Dept. Defense.

Hudson, R.R., 1983. Properties of a neutral allele model with intragenic recombination. Theor. Popul. Biol. 23, 183–201.

Hudson, R.R., 2002. Generating samples under a Wright–Fisher neutral model of genetic variation. Bioinformatics 18 (2), 337–338.

Hudson, R.R., Kaplan, N., 1985. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. Genetics 111 (1), 147–164.

Kelleher, J., Barton, N.H., Etheridge, A.M., 2013. Coalescent simulation in continuous space. Bioinformatics 29 (7), 955–956.

Knuth, D.E., 1997a. Fundamental Algorithms, third ed. In: The Art of Computer Programming, vol. 1. Addison-Wesley, Reading, Massachusetts.

Knuth, D.E., 1997b. Seminumerical Algorithms, third ed. In: The Art of Computer Programming, vol. 2. Addison-Wesley, Reading, Massachusetts.

Knuth, D.E., 1998. Sorting and Searching, second ed. In: The Art of Computer Programming, vol. 3. Addison-Wesley, Reading, Massachusetts.

Knuth, D.E., 2011. Combinatorial Algorithms, Part 1. In: The Art of Computer Programming, vol. 4A. Addison-Wesley, Upper Saddle River, New Jersey.

Liang, L., Zöllner, S., Abecasis, G.R., 2007. GENOME: a rapid coalescent-based whole genome simulator. Bioinformatics 23 (12), 1565–1567.

Malécot, G., 1948. Les mathématiques de l'hérédité. Masson et Cie, Paris.

Marjoram, P., Wall, J.D., 2006. Fast coalescent simulation. BMC Genet. 7, 16.

McVean, G.A.T., Cardin, N.J., 2005. Approximating the coalescent with recombination. Philos. Trans. R. Soc. Lond. Ser. B 360, 1387–1393.

Morjan, C.L., Rieseberg, L.H., 2004. How species evolve collectively: implications of gene flow and selection for the spread of advantageous alleles. Mol. Ecol. 13 (6), 1341–1356.

Padhukasahasram, B., Marjoram, P., Wall, J.D., Bustamante, C.D., Nordborg, M., 2008. Exploring population genetic models with recombination using efficient forward-time simulations. Genetics 178 (4), 2417–2427.

Wiuf, C., Hein, J., 1999. Recombination as a point process along sequences. Theor. Popul. Biol. 55 (3), 248–259.

Wright, S., 1943. Isolation by distance. Genetics 28 (2), 114–138.

Wright, S., 1946. Isolation by distance under diverse systems of mating. Genetics 31 (1), 39–59.

Wright, S., 1978. Variability Within and Among Natural Populations. In: Evolution and the Genetics of Populations, vol. 4. The University of Chicago Press, Chicago.