BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 15 2012, pages 2008–2015
doi:10.1093/bioinformatics/bts314

Genetics and population analysis

Advance Access publication May 28, 2012

Blockwise HMM computation for large-scale population genomic inference

Joshua S. Paul[1] and Yun S. Song[1,2,*]

[1]Computer Science Division and [2]Department of Statistics, University of California, Berkeley, CA 94720, USA

Associate Editor: Jeffrey Barrett

ABSTRACT

Motivation: A promising class of methods for large-scale population genomic inference use the conditional sampling distribution (CSD), which approximates the probability of sampling an individual with a particular DNA sequence, given that a collection of sequences from the population has already been observed. The CSD has a wide range of applications, including imputing missing sequence data, estimating recombination rates, inferring human colonization history and identifying tracts of distinct ancestry in admixed populations. Most well-used CSDs are based on hidden Markov models (HMMs). Although computationally efficient in principle, methods resulting from the common implementation of the relevant HMM techniques remain intractable for large genomic datasets.

Results: To address this issue, a set of algorithmic improvements for performing the exact HMM computation is introduced here, by exploiting the particular structure of the CSD and typical characteristics of genomic data. It is empirically demonstrated that these improvements result in a speedup of several orders of magnitude for large datasets and that the speedup continues to increase with the number of sequences. The optimized algorithms can be adopted in methods for various applications, including the ones mentioned above and make previously impracticable analyses possible.

Availability: Software available upon request.

Supplementary Information: Supplementary data are available at *Bioinformatics* online.

Contact: yss@eecs.berkeley.edu

Received on March 3, 2012; revised on April 18, 2012; accepted on May 22, 2012

1 INTRODUCTION

With the cost of genomic sequencing rapidly decreasing, there is a growing need for statistical methodologies that can efficiently accommodate genomic-scale data for many individuals while accounting for complex patterns of variation (e.g. linkage disequilibrium) caused by evolutionary processes such as mutation and recombination. The applicable underlying statistical model is the coalescent with recombination, which describes the distribution of genealogical histories for a collection of individuals. Known methods for inference under this model are generally intractable at the genomic scale, so practicable methods must realize a balance between computational efficiency and approximating key properties of the coalescent with recombination. A promising class of such methods use the conditional sampling distribution (CSD).

A CSD approximates the probability (under the coalescent with recombination) of sampling an individual with a particular DNA sequence, given that a collection of sequences from the population has already been observed. Intuitively, recombination partitions the newly sampled sequence into segments, each of which is a copy of the corresponding segment in a previously sampled sequence, with imperfections introduced by mutation. For computational efficiency, this construction is often cast as a hidden Markov model (HMM). The hidden state at a site indicates the previously sampled sequence being copied and the associated observed state the allele of the new sample.

Even within this framework there are alternatives, as it is possible to trade-off fidelity to the underlying coalescent process for computational efficiency. The CSD of Paul *et al.* (2011), for example, is the most accurate but is a constant factor slower than CSDs proposed by Fearnhead and Donnelly (2001) and by Li and Stephens (2003), with the latter being the fastest, but least accurate. CSDs for more complex models, incorporating gene conversion (Gay *et al.*, 2007; Yin *et al.*, 2009), diploidy (Marchini *et al.*, 2007), demography (Davison *et al.*, 2009; Hellenthal *et al.*, 2008) and admixture (Price *et al.*, 2009; Sundquist *et al.*, 2008), have also been proposed.

Methods incorporating the CSD generally fall into one of several categories. Likelihoods can be approximated using CSD-based importance sampling (De Iorio and Griffiths, 2004a, b; Fearnhead and Donnelly, 2001; Griffiths *et al.*, 2008; Stephens and Donnelly, 2000) coupled with composite methods (Fearnhead and Donnelly, 2002; Hudson, 2001) or directly as a product of CSDs (Li and Stephens, 2003). In conjunction with expectation–maximization or Markov chain Monte Carlo, these methods have been used for estimation of fine-scale recombination rates (Crawford *et al.*, 2004; Fearnhead and Smith, 2005; Li and Stephens, 2003; McVean *et al.*, 2004), gene conversion parameters (Gay *et al.*, 2007; Yin *et al.*, 2009), population demography (Davison *et al.*, 2009; Hellenthal *et al.*, 2008) and population structure (Lawson *et al.*, 2012). It is also possible to infer and use the hidden states in the HMM CSD computation. This has been used for admixture inference (Price *et al.*, 2009; Sundquist *et al.*, 2008; Wegmann *et al.*, 2011), in which genomic segments corresponding to ancestral populations are identified and also within a pseudo-Gibbs sampling framework to phase genotype sequence data into haplotype sequence data and to impute missing data (Howie *et al.*, 2009; Li *et al.*, 2010; Marchini *et al.*, 2007; Stephens and Scheet, 2005).

*To whom correspondence should be addressed.

Nearly, all of these methods rely on iterative Monte Carlo or expectation–maximization techniques. As a result, they are computationally intensive, often requiring several hours, or, in some cases, days, to produce a result, even for modest non-genomic datasets (Howie *et al.*, 2009); directly extending the methods to large genomic datasets is thus often impractical. Moreover, nearly all of the running time is expended on CSD computation, and so the choice of CSD is often made on the basis of efficiency and (arguably) at the expense of accuracy (Browning and Browning, 2007; Howie *et al.*, 2009; Li and Stephens, 2003; Scheet and Stephens, 2006; Stephens and Scheet, 2005).

In this article, we help to overcome these limitations by proposing two related optimizations to the relevant HMM-based CSD computations. Consider sampling a large number of sequences from a population. If the sampled sequences are very long, it is likely that nearly all of them will be unique. However, for most relatively short regions, the number of unique subsequences will be reduced. This intuition forms the basis of the first optimization, which locally reduces the complexity of the CSD computation, thereby improving efficiency. The collection of locally unique subsequences on which this optimization depends are formalized as a partition $\mathcal{C}$ of the sampled sequences; we characterize the optimal partition given the sampled sequences and provide a fast algorithm for approximating this optimum.

A second common feature of the sampled sequences is an abundance of non-polymorphic sites. These sites are informative—for example, a local over-abundance of non-polymorphic sites indicates a recent common ancestor, which in turn indicates a low propensity for recombination—and should be included in the analysis. Indeed, Li and Durbin (2011) used the physical distribution of polymorphic and non-polymorphic sites between a pair of sequences to infer past population sizes of humans. Using the fact that non-polymorphic sites do not differentiate the sequences, we show that it is possible to reduce the complexity of the CSD computation at non-polymorphic sites. We stress that our solution is different from simply ignoring non-polymorphic sites; we are proposing algorithmic improvements to incorporate non-polymorphic sites into the analysis in an efficient way.

In formally describing and evaluating our optimizations, we restrict attention to the most accurate HMM-based CSD, $\hat{\pi}_{\text{SMC}}$, proposed by Paul *et al.* (2011) and consider the problem of computing the conditional sampling probability (CSP), denoted $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$, of a particular individual α given a collection $\mathbf{n}$ of previously sampled individuals. (Incidentally, in the case the size n of the previously observed sample $\mathbf{n}$ is 1, the HMM underlying $\hat{\pi}_{\text{SMC}}$ is equivalent to the HMM used in the aforementioned work of Li and Durbin; we anticipate that $\hat{\pi}_{\text{SMC}}$ provides one way of extending this line of work to many sequences). On simulated data, our algorithmic improvement leads to a speedup of about $550\times$ for $n = 5000$ previously sampled individuals; by making regularity assumptions on mutation and recombination rates, this speedup increases to about $1850\times$. Importantly, we show that the empirically observed speed-up increases with the number n of previously sampled individuals.

Although we describe our optimizations in the context of computing $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$, they are more generally applicable. We provide two sufficient conditions for our optimizations and use them to show the applicability to other HMM-based CSDs, including those of Fearnhead and Donnelly (2001) and Li and Stephens (2003), as well as CSDs for more complex demographic models and population genetic HMMs. Also, in the Supplementary Material, we describe extending our algorithms to allow for efficient inference of hidden states, often termed posterior decoding.

We stress that the work presented here is fundamentally different from previous works on '*approximating*' CSD-based population genetic inference. Howie *et al.* (2009) consider a fixed-size subset of the haplotype configuration $\mathbf{n}$, chosen using a measure of 'closeness' to the sampled haplotype α, in order to reduce the state space of the HMM-based CSD and speed up computation. Similarly, Scheet and Stephens (2006) and Browning and Browning (2007) consider an HMM with reduced state space by compacting the configuration $\mathbf{n}$ into a substantially smaller haplotype model. More recently, Delaneau *et al.* (2012) have proposed an approximate HMM formulation relying on a partition of the sampled sequences similar to that proposed herein (Section 2.6). As described earlier, using such heuristics improves computational efficiency, but ultimately at the expense of accuracy. The purpose of this article is to provide highly optimized '*and exact*' computation for a large class of approximate CSDs, rather than to introduce additional approximations to the underlying models.

2 METHODS

Herein, we describe the HMM formulation of $\hat{\pi}_{\text{SMC}}$, the algorithms that are currently being used to compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ and the optimizations we are proposing to improve the running time.

We remark that the theoretical analysis of our algorithms is limited to asymptotic time (and space) complexity. As a measure of real-world performance, asymptotic analyses often leave much to be desired. Consider, for example, a sample in which 1 out of every 1000 sites is polymorphic. If we denote by k the total number of sites and k_{p} the number of polymorphic sites, then formally $O(k) = O(k_{\text{p}})$. Nevertheless, we would like to distinguish between an algorithm that operates on each of the k sites and an algorithm that operates only on the k_{p} polymorphic sites, as the the latter will be some $1000\times$ faster; we thus write the complexities for the two algorithms as $O(k)$ and $O(k_{\text{p}})$, respectively.

2.1 Notation

We consider haplotypes in the finite-locus finite-alleles setting. Throughout, we suppose that there are k loci, and that recombination may occur between any adjacent pair of loci $(\ell, \ell+1)$ where $1 \leq \ell < k$.

The space of all such haplotypes is denoted by $\mathcal{H}$, and given a haplotype $h \in \mathcal{H}$, the allele at locus ℓ is denoted by $h[\ell]$ and the sub-haplotype for a range of loci $\ell \leq \ell'$ is denoted by $h[\ell:\ell']$. A sample configuration of haplotypes is specified by a vector $\mathbf{n} = (n_h)_{h \in \mathcal{H}}$, with n_h being the number of haplotypes of type h in the sample. The set of '*unique*' haplotypes associated with configuration $\mathbf{n}$ is denoted by $\mathcal{H}_{\mathbf{n}} = \{h \in \mathcal{H} : n_h > 0\}$. Finally, the total number of haplotypes is denoted by $|\mathbf{n}| = n$, the number of unique haplotypes by $|\mathcal{H}_{\mathbf{n}}| = n_{\text{u}}$ and the number of polymorphic loci, which generally depends on the sample $\mathbf{n}$, by k_{p}.

2.2 A brief description of $\hat{\pi}_{\text{SMC}}$

Suppose that, conditioned on having already observed a haplotype configuration $\mathbf{n}$, we wish to sample a new haplotype α. By generalizing the technique of Griffiths *et al.* (2008) based on the diffusion process, Paul and Song (2010) introduced the CSD $\hat{\pi}_{\text{PS}}$ intended to approximate key properties of the coalescent with recombination, the model under which inference is to be performed. The central idea of $\hat{\pi}_{\text{PS}}$ is to fix the unknown ancestry of $\mathbf{n}$ to be the '*trunk genealogy*' $\mathcal{A}^*(\mathbf{n})$, in which lineages associated with the haplotypes do not mutate, recombine, or coalesce with one another, but rather

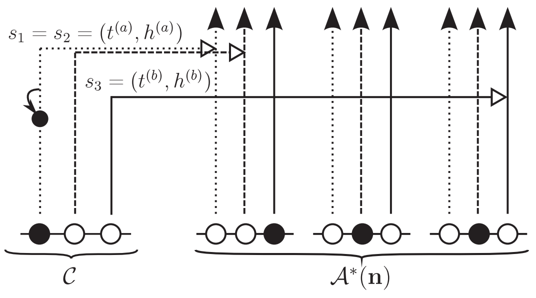
$s_1 = s_2 = (t^{(a)}, h^{(a)})$
$s_3 = (t^{(b)}, h^{(b)})$
$\mathcal{C}$
$\mathcal{A}^*(\mathbf{n})$

Fig. 1. Illustration of a sequentially Markov framework for the conditional genealogy $\mathcal{C}$. The trunk genealogy $\mathcal{A}^*(\mathbf{n})$ is indicated. The three loci of each haplotype are each represented by a circle, with the shading indicating the allelic type at that locus. Time is represented vertically, with the present (time 0) at the bottom. The marginal genealogies at the first, second and third loci are shown as dotted, dashed and solid lines, respectively. Mutation event, along with resulting allele, is indicated by small arrow. Absorption events at each locus, and the corresponding absorption time ($t^{(a)}$ and $t^{(b)}$) and haplotype ($h^{(a)}$ and $h^{(b)}$, respectively), are indicated by horizontal lines

extend infinitely into the past. Having fixed the ancestry of $\mathbf{n}$, a conditional genealogy $\mathcal{C}$ associated with haplotype α is sampled; within $\mathcal{C}$, lineages evolve backwards in time subject to mutation, recombination, coalescence and '*absorption*' into one of the lineages of $\mathcal{A}^*(\mathbf{n})$. When every lineage of $\mathcal{C}$ has been absorbed, the process terminates; the type of every lineage of $\mathcal{C}$ is now determined and a sample for α is generated.

Although a recursion for computing the CSP $\hat{\pi}_{\mathrm{PS}}(\alpha|\mathbf{n})$ is known, it is computationally intractable for all but the smallest datasets. To remedy this limitation, Paul *et al.* (2011) adopted a sequentially Markov framework (McVean and Cardin, 2005) for the conditional genealogy $\mathcal{C}$. The central idea is to consider the '*marginal*' conditional genealogy s_ℓ associated with each locus ℓ, which is described, disregarding mutation events, by the pair $s_\ell = (t_\ell, h_\ell)$, where $t_\ell \in [0,\infty)$ is the absorption time and $h_\ell \in \mathcal{H}_{\mathbf{n}}$ is the absorption haplotype. The conditional genealogy $\mathcal{C}$ can thus be represented as a sequence of marginal conditional genealogies $\{s_\ell\}$. See Figure 1 for an illustration.

In general, the random sequence of marginal conditional genealogies is not Markov, due to the potential for coalescence events within the conditional genealogy. Nonetheless, it is possible to '*approximate*' this sequence as Markov by using a two-locus transition distribution. Mutation can then be realized at each locus independently as a Poisson process on the marginal conditional genealogy, thereby generating a sample for α. The resulting CSD is denoted by $\hat{\pi}_{\mathrm{SMC}}$. Paul *et al.* (2011) found that the Markov approximation underlying $\hat{\pi}_{\mathrm{SMC}}$ has minimal effect on accuracy compared with $\hat{\pi}_{\mathrm{PS}}$, coinciding with findings in similar domains (Marjoram and Wall, 2006; McVean and Cardin, 2005).

Owing to its Markov structure, the CSD $\hat{\pi}_{\mathrm{SMC}}$ can be cast as an HMM wherein the ℓth hidden state is the marginal conditional genealogy $s_\ell = (t_\ell, h_\ell)$ and the ℓth observed state the conditionally sampled allele $\alpha[\ell]$. In order to use standard dynamic programming methodologies for inference, the state space of the HMM must be finite, and so absorption time is discretized by partitioning $[0,\infty)$ into a (possibly large) finite number m of intervals I and considering an absorption interval, denoted by $d_\ell \in I$, rather than an absorption time. The discretized hidden state at the ℓth locus is then $s_\ell = (d_\ell, h_\ell)$, and the initial, transition and emission distributions for the discretized $\hat{\pi}_{\mathrm{SMC}}$ HMM are denoted by $\zeta(\cdot)$, $\phi_\ell(\cdot|\cdot)$ and $\xi_\ell(\cdot|\cdot)$, respectively. Interested readers should consult Paul and Song (2010) and Paul *et al.* (2011) for details.

2.3 Computation with $\hat{\pi}_{\mathrm{SMC}}$

The CSP $\hat{\pi}_{\mathrm{SMC}}(\alpha|\mathbf{n})$ under the discrete-space HMM can be efficiently computed using the '*forward algorithm*', a dynamic program associated with the HMM forward recursion:

$$\hat{\pi}_{\mathrm{SMC}}(\alpha|\mathbf{n}) = \sum_{d\in I}\sum_{h\in\mathcal{H}_{\mathbf{n}}} F_k(d,h), \tag{1}$$

where

$$F_\ell(d,h) = \xi_\ell(\alpha[\ell]|d,h)\sum_{d'\in I}\sum_{h'\in\mathcal{H}_{\mathbf{n}}} \phi_{\ell-1}(d,h|d',h')F_{\ell-1}(d',h'), \tag{2}$$

with base case

$$F_0(d,h) = \zeta(d,h). \tag{3}$$

Importantly, this recursion, and the associated dynamic program with time complexity $O(k(n_{\mathrm{u}}m)^2)$, is applicable to a general HMM. In the following, we examine the transition and emission distributions more carefully and obtain a series of refined recursions and the associated dynamic programs.

2.4 Improving efficiency through the transition distribution

Consider the description of $\hat{\pi}_{\mathrm{SMC}}$ given above and more rigorously defined in Paul *et al.* (2011). If a recombination does not occur between loci $\ell-1$ and ℓ, then $(d_{\ell-1}, h_{\ell-1}) = (d_\ell, h_\ell)$; moreover, if recombination does occur, the absorbing haplotype h_ℓ is independent of $h_{\ell-1}$ and uniformly distributed. As a result, we have the following property (We remark that Property 1 is a sufficient, though not necessary, condition for the algorithmic optimizations described in this and subsequent sections. For example, as stated, the transition distribution imposes a uniform distribution on the absorbing haplotype in the case of a recombination; in fact, the algorithms can be generalized to accommodate an arbitrary distribution over haplotypes that does not depend on d' or h'. In Section 4, we discuss the applicability of these optimizations to more general (and more specialized) classes of approximate CSDs.):

PROPERTY 1. *The initial and transition probabilities ζ and ϕ, take the following functional form*:

$$\zeta(d,h) = x^{(d)}\cdot\frac{n_h}{n},$$

$$\phi_\ell(d,h|d',h') = y_\ell^{(d')}\cdot\delta_{d',d}\delta_{h',h} + z_\ell^{(d',d)}\cdot\frac{n_h}{n},$$

where $x^{(d)}$, $y_\ell^{(d')}$ and $z_\ell^{(d',d)}$ are known analytic expressions.

Using Property 1 in conjunction with definitions

$$Q_\ell(d) = \sum_{h\in\mathcal{H}_{\mathbf{n}}} F_\ell(d,h) \quad\text{and}\quad U_\ell(d) = \sum_{d'\in I} z_\ell^{(d',d)} Q_\ell(d'), \tag{4}$$

we can express equations (1)–(3) as

$$\hat{\pi}_{\mathrm{SMC}}(\alpha|\mathbf{n}) = \sum_{d\in I} Q_k(d), \tag{5}$$

where

$$F_\ell(d,h) = \xi_\ell(\alpha[\ell]|d,h)\left[y_{\ell-1}^{(d)}F_{\ell-1}(d,h) + \frac{n_h}{n}U_{\ell-1}(d)\right], \tag{6}$$

with base case

$$F_0(d,h) = x^{(d)}\cdot\frac{n_h}{n}. \tag{7}$$

Using these recursions, the dynamic program in Algorithm 1 can be used to compute $\hat{\pi}_{\mathrm{SMC}}(\alpha|\mathbf{n})$. Within the pseudocode description, the time complexity of Lines 6, 7 and 8 are $O(m)$, $O(n_{\mathrm{u}})$ and $O(n_{\mathrm{u}})$, respectively. As a result, the time complexity of Lines 5–9, and for the algorithm as a whole, is $O(km(m+n_{\mathrm{u}}))$. This represents a substantial improvement over the quadratic dependence on n_{u} in the naive forward algorithm for HMMs. This simple optimization has already been generally adopted (Fearnhead and Donnelly, 2001; Li and Stephens, 2003; Paul *et al.*, 2011) and serves as a baseline for improvement.

Algorithm 1 Compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ using a forward-type recursion improved by considering Property 1

```
1.  for all $d \in I$ do
2.      Compute $F_0(d,h)$ by (7), $\forall h \in \mathcal{H}_\mathbf{n}$
3.      Compute $Q_0(d)$ using (4)
4.  end for
5.  for $\ell = 1 \to k$ and $d \in I$ do
6.      Compute $U_{\ell-1}(d)$ using (4)
7.      Compute $F_\ell(d,h)$ using (6), $\forall h \in \mathcal{H}_\mathbf{n}$
8.      Compute $Q_\ell(d)$ using (4)
9.  end for
10. Compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ using (5)
```

2.5 Improving efficiency through the emission distribution

The state of the $\hat{\pi}_{\text{SMC}}$ HMM at locus ℓ is a tuple (d_ℓ, h_ℓ). However, the emission probability of $\alpha[\ell]$ is governed only by the time interval d_ℓ and the allele $h_\ell[\ell]$. As a result, the following property holds:

PROPERTY 2. *Consider a subset $\mathcal{B} \subset \mathcal{H}_\mathbf{n}$ such that there exists an allele a with $h[\ell]=a$ for all $h \in \mathcal{B}$. Then, for each absorption interval $d \in I$, the emission distribution $\xi_\ell(\cdot|d,h)$ is identical for all $h \in \mathcal{B}$. We indicate this fact by writing $\xi_\ell(\cdot|d,h) = \xi_\ell(\cdot|d,\mathcal{B})$ for all $h \in \mathcal{B}$.*

With this in mind, define a '*partition*' $\mathcal{C}$ of the haplotype configuration $\mathbf{n}$ to be a collection of blocks of the form $(\mathcal{B}, \ell_s, \ell_e)$, such that

- For every $(\mathcal{B}, \ell_s, \ell_e) \in \mathcal{C}$, there exists a sub-haplotype x such that $h[\ell_s : \ell_e] = x$ for all $h \in \mathcal{B}$.
- For every haplotype $h \in \mathcal{H}_\mathbf{n}$ and $1 \le \ell \le k$, there exists '*exactly*' one $(\mathcal{B}, \ell_s, \ell_e) \in \mathcal{C}$ with $h \in \mathcal{B}$ and $\ell_s \le \ell \le \ell_e$.

For a given locus ℓ, a configuration partition $\mathcal{C}$ induces a partition of the haplotypes $\mathcal{H}_\mathbf{n}$, denoted by $\mathcal{C}_\ell$, and Property 2 applies to each $\mathcal{B} \in \mathcal{C}_\ell$. In the next sections, we present new sets of recursions and dynamic programming algorithms valid for an arbitrary partition $\mathcal{C}$. The computational complexity of these algorithms will depend on $\mathcal{C}$ through two functions, namely $\Psi(\mathcal{C})$ and $\Omega(\mathcal{C})$, defined as follows: For locus ℓ, define $\psi_\ell(\mathcal{C}) = |\mathcal{C}_\ell|$, the number of blocks in $\mathcal{C}_\ell$ and define $\omega_\ell(\mathcal{C})$ to be the total number of haplotypes in blocks of the configuration partition '*ending*' at locus ℓ. Then,

$$\Psi(\mathcal{C}) = \sum_{\ell=1}^{k} \psi_\ell(\mathcal{C}) = \sum_{\ell=1}^{k} |\mathcal{C}_\ell|,$$

$$\Omega(\mathcal{C}) = \sum_{\ell=1}^{k} \omega_\ell(\mathcal{C}) = \sum_{(\mathcal{B},\ell_s,\ell_e)\in\mathcal{C}} |\mathcal{B}|.$$

In some cases, we are primarily concerned with polymorphic loci, and so we define $\Psi_p(\mathcal{C})$ to be the analog of $\Psi(\mathcal{C})$ summed over '*only*' polymorphic loci.

Finally, we define the trivial partition C_T for haplotype configuration $\mathbf{n}$ as the partition containing a single block $(\{h\}, 1, k)$ for each $h \in \mathcal{H}_\mathbf{n}$. Note that $\Psi(C_T) = k \cdot n_u$ and $\Omega(C_T) = n_u$. See Figure 2 for an illustration of both C_T and two non-trivial configuration partitions.

2.5.1 A general configuration partition Let $\mathcal{C}$ be a configuration partition of $\mathbf{n}$. Begin by defining

$$Q_\ell(d,\mathcal{B}) = \sum_{h\in\mathcal{B}} F_\ell(d,h) \quad \text{so that} \quad Q_\ell(d) = \sum_{\mathcal{B}\in\mathcal{C}_\ell} Q_\ell(d,\mathcal{B}). \tag{8}$$

Now suppose $(\mathcal{B}, \ell_s, \ell_e) \in \mathcal{C}$. Applying Definition (8) and Property 2 to equation (6), then for $\ell_s \le \ell \le \ell_e$,

$$Q_\ell(d,\mathcal{B}) = \xi_\ell(\alpha[\ell]|d,\mathcal{B}) \left[y_{\ell-1}^{(d)} Q_{\ell-1}(d,\mathcal{B}) + \frac{n_\mathcal{B}}{n} U_{\ell-1}(d) \right], \tag{9}$$

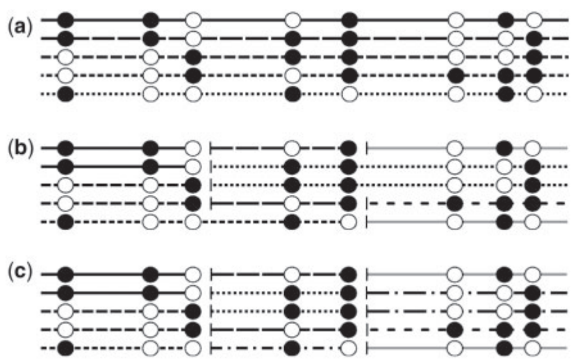

Fig. 2. Illustration of three alternative configuration partitions. Each row represents a haplotype, with white and black circles representing the allele at each of eight polymorphic loci. The line style of each sub-haplotype indicates the block to which it belongs. (**a**) The trivial configuration partition C_T; $\Psi_p(C_T) = 40$ and $\Omega(C_T) = 5$. (**b**) A non-trivial configuration partition, C; $\Psi_p(C) = 24$ and $\Omega(C) = 12$. (**c**) The configuration partition C_s found by the algorithm described in Section 2.6 for $s = 3$; $\Psi_p(C_s) = 24$ and $\Omega(C_s) = 15$

where $n_\mathcal{B} = \sum_{h\in\mathcal{B}} n_h$. Similarly, by induction and making use of equations (6) and (9), it is possible to show that, for $\ell_s \le \ell \le \ell_e$ and $h \in \mathcal{B}$,

$$F_\ell(d,h) = T_\ell(d,\mathcal{B}) \cdot F_{\ell_s-1}(d,h) + \frac{n_h}{n_\mathcal{B}} \Big(Q_\ell(d,\mathcal{B}) - T_\ell(d,\mathcal{B}) Q_{\ell_s-1}(d,\mathcal{B}) \Big), \tag{10}$$

where $T_\ell(d,\mathcal{B}) = \prod_{\ell'=\ell_s}^{\ell} \xi_{\ell'}(\alpha[\ell']|d,\mathcal{B}) \cdot y_{\ell'-1}^{(d)}$, and solves the recursion,

$$T_\ell(d,\mathcal{B}) = \xi_\ell(\alpha[\ell]|d,\mathcal{B}) \cdot y_{\ell-1}^{(d)} \cdot T_{\ell-1}(d,\mathcal{B}), \tag{11}$$

for $\ell_s \le \ell \le \ell_e$, with base case $T_{\ell_s-1}(d,\mathcal{B}) = 1$.

For each block $(\mathcal{B}, \ell_s, \ell_e) \in \mathcal{C}$, we take advantage of equations (9) and (11) to directly compute $Q_\ell(d,\mathcal{B})$ and $T_\ell(d,\mathcal{B})$ at every locus $\ell_s \le \ell \le \ell_e$. At the end of each block, when $\ell = \ell_e$, the finer-grain values $F_\ell(d,h)$ are computed for each $h \in \mathcal{B}$ using equation (10), and subsequently used to compute initial values for blocks beginning at locus $\ell + 1$. The associated dynamic program to compute the CSP $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ is given in Algorithm 2.

Within Algorithm 2, the time complexity of Line 7 is $O(m)$; of Line 8 is $O(\psi_\ell(\mathcal{C}))$ and of Lines 9 and 10 is $O(\omega_\ell(\mathcal{C}))$. Thus, the time complexity of Lines 6–11, and the dynamic program, is $O(km^2 + m(\Psi(\mathcal{C}) + \Omega(\mathcal{C})))$. Observe that Algorithm 1 is a special case of Algorithm 2 for $\mathcal{C} = C_T$. Thus, if it is possible to obtain a configuration partition $\mathcal{C}$ for $\mathbf{n}$ such that $\Psi(\mathcal{C}) + \Omega(\mathcal{C})$ is substantially less than $\Psi(C_T) + \Omega(C_T) = kn_u + n_u$, our new algorithm may be considerably faster than Algorithm 1; constructing such a configuration partition is the subject of Section 2.6.

2.5.2 The absence of polymorphism In many reasonable evolutionary scenarios, a great many loci will not be polymorphic. Accommodating such loci in the analysis is important and can be done efficiently making use of Property 2. In particular, for a non-polymorphic locus ℓ, Property 2 applies to the trivial set $\mathcal{B}_0 = \mathcal{H}_\mathbf{n}$, and therefore the emission distribution can be written $\xi_\ell(\cdot|d,\mathcal{B}_0) = \xi_\ell(\cdot|d)$ and moreover, $Q_\ell(d,\mathcal{B}_0) = Q_\ell(d)$.

Suppose consecutive loci $\ell_s^*, \ldots, \ell_e^*$ are not polymorphic. Rewriting equations (9) and (10) for block $(\mathcal{B}_0, \ell_s^*, \ell_e^*)$ yields, for $\ell_s^* \le \ell \le \ell_e^*$,

$$Q_\ell(d) = \xi_\ell(\alpha[\ell]|d) \cdot \left[y_{\ell-1}^{(d)} Q_{\ell-1}(d) + U_{\ell-1}(d) \right], \tag{12}$$

and for $\ell_s^* \le \ell \le \ell_e^*$ and $h \in \mathcal{B}_0 = \mathcal{H}_\mathbf{n}$,

$$F_\ell(d,h) = T_\ell(d) \cdot F_{\ell_s^*-1}(d,h) + \frac{n_h}{n} \Big(Q_\ell(d) - T_\ell(d) Q_{\ell_s^*-1}(d) \Big), \tag{13}$$

where $T_\ell(d) = \prod_{\ell'=\ell_s^*}^{\ell} \xi_{\ell'}(\alpha[\ell']|d) \cdot y_{\ell'-1}^{(d)}$ and solves the recursion

$$T_\ell(d) = \xi_\ell(\alpha[\ell]|d) \cdot y_{\ell-1}^{(d)} \cdot T_{\ell-1}(d), \tag{14}$$

for $\ell_s^* \le \ell \le \ell_e^*$, with base case $T_{\ell_s^*-1}(d) = 1$.

Algorithm 2 Compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ using a forward-type recursion improved by considering Properties 1 and 2, for a configuration partition $\mathcal{C}$

```
1. for all $d \in I$ do
2.     Compute $F_0(d,h)$ using (7), $\forall h \in \mathcal{H}_\mathbf{n}$
3.     Compute $Q_0(d,\mathcal{B})$ using (8) and $T_0(d,\mathcal{B})=1$, $\forall(\mathcal{B},1,\ell_e)\in\mathcal{C}$
4.     Compute $Q_0(d)$ using (8)
5. end for
6. for $\ell = 1 \to k$ and $d \in I$ do
7.     Compute $U_{\ell-1}(d)$ using (4)
8.     Compute $Q_\ell(d,\mathcal{B})$ and $T_\ell(d,\mathcal{B})$ using (9) and (11), $\forall(\mathcal{B},\ell_s,\ell_e)\in\mathcal{C}$ such that $\ell_s \le \ell \le \ell_e$; compute $Q_\ell(d)$ using (8)
9.     Compute $F_\ell(d,h)$ using (10), $\forall h \in \mathcal{B}$ and $\forall(\mathcal{B},\ell_s,\ell)\in\mathcal{C}$
10.    Compute $Q_\ell(d,\mathcal{B})$ using (8) and $T_\ell(d,\mathcal{B})=1$, $\forall(\mathcal{B},\ell+1,\ell_e)\in\mathcal{C}$
11. end for
12. Compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ using (5)
```

Algorithm 3 Computation of $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ improved by considering Properties 1 and 2, and a special case for non-polymorphic loci, for a configuration partition $\mathcal{C}$ such that $\forall(\mathcal{B},\ell_s,\ell_e)\in\mathcal{C}$, ℓ_e is polymorphic

```
1. Algorithm 2, lines 1–5; and set $T_0(d)=1$ $\forall d \in I$ and $\ell_s^*=1$
2. for $\ell = 1 \to k$ and $d \in I$ do
3.     if locus $\ell$ is polymorphic then
4.         if locus $\ell-1$ is not polymorphic then
5.             Compute $Q_{\ell-1}(d,\mathcal{B})$ and $T_{\ell-1}(d,\mathcal{B})$ using (15) and (16)
6.         end if
7.         Algorithm 2, lines 7–10
8.         Set $T_\ell(d)=1$ and $\ell_s^* = \ell+1$
9.     else
10.        Compute $U_{\ell-1}(d)$, $Q_\ell(d)$, and $T_\ell(d)$ using (4), (12), and (14)
11.    end if
12. end for
13. Compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ using (5)
```

Now, let $\mathcal{C}$ be a configuration partition with $(\mathcal{B},\ell_s,\ell_e)\in\mathcal{C}$. Suppose that there is a stretch of non-polymorphic loci $\ell_s^*,\dots,\ell_e^*$ and that $\ell_s \le \ell_s^* \le \ell_e^* \le \ell_e$. Applying Definition (8) to equation (13), yields, for $\ell_s^* \le \ell \le \ell_e^*$,

$$Q_\ell(d,\mathcal{B}) = T_\ell(d) Q_{\ell_s^*-1}(d,\mathcal{B}) + \frac{n_\mathcal{B}}{n}\Big[Q_\ell(d) - T_\ell(d) Q_{\ell_s^*-1}(d)\Big]. \qquad (15)$$

Similarly, considering the definition of $T_\ell(d,\mathcal{B})$ along with equation (14),

$$T_\ell(d,\mathcal{B}) = T_\ell(d)\cdot T_{\ell_s^*-1}(d,\mathcal{B}). \qquad (16)$$

Algorithm 2 can be modified to accommodate such stretches of non-polymorphic loci as a special case, making use of equations (12) and (14) to directly compute the values of $Q_\ell(d)$ and $T_\ell(d)$ at each non-polymorphic locus ℓ. If we then assume (without loss of generality) that each $(\mathcal{B},\ell_s,\ell_e)\in\mathcal{C}$ has ℓ_e at a polymorphic locus, then at the final non-polymorphic locus, for which $\ell = \ell_e^*$, equations (15) and (16) may be used to compute $Q_\ell(d,\mathcal{B})$ and $T_\ell(d,\mathcal{B})$ for each $\mathcal{B}\in\mathcal{C}_\ell$. This modification is detailed in Algorithm 3.

Within Algorithm 3, the time complexity of Lines 5 and 8 is $O(1)$, of Line 7 is $O(m+\psi_\ell(\mathcal{C})+\omega_\ell(\mathcal{C}))$ and of Line 10 is $O(m)$. As a result, the time complexity of Lines 2–12, and of the dynamic program, is $O(km^2 + m(\Psi_p(\mathcal{C})+\Omega(\mathcal{C})))$. Relative to Algorithm 2, less computation needs to be done for non-polymorphic loci; thus, in the typical case of many non-polymorphic loci, this dynamic program will have a decreased running time. For $\mathcal{C}=C_T$, the time complexity is $O(km^2 + k_p m n_u)$.

Algorithm 4 Computation of $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ improved by considering Properties 1 and 2, and a special '*optimized*' case for non-polymorphic loci, for a configuration partition $\mathcal{C}$ such that $\forall(\mathcal{B},\ell_s,\ell_e)\in\mathcal{C}$, ℓ_e is polymorphic

```
1. Algorithm 3, line 1
2. for polymorphic $\ell \in \{2 \to k\}$ and $d \in I$ do
3.     if locus $\ell-1$ is not polymorphic then
4.         Compute $Q_{\ell-1}(d)$ and $T_{\ell-1}(d)$ using (17)
5.     end if
6.     Algorithm 3, lines 4–8
7. end for
8. Compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ using (5)
```

2.5.3 An optimization for non-polymorphic loci The key recursions (12) and (14) for non-polymorphic loci can be written in matrix form as $\mathcal{Q}_\ell = \mathcal{E}_\ell(\mathcal{Y}_\ell+\mathcal{Z}_\ell)\mathcal{Q}_{\ell-1}$ and $\mathcal{T}_\ell = \mathcal{E}_\ell\mathcal{Y}_\ell\mathcal{T}_{\ell-1}$, where $\mathcal{Q}_\ell$ and $\mathcal{T}_\ell$ are m-dimensional column vectors having dth entry $Q_\ell(d)$ and $T_\ell(d)$, respectively; $\mathcal{E}_\ell$ and $\mathcal{Y}_\ell$ are $(m\times m)$-dimensional diagonal matrices having $(d\times d)$th entry $\xi_\ell(\alpha[\ell]|d)$ and $y_{\ell-1}^{(d)}$, respectively; and $\mathcal{Z}_\ell$ is an $(m\times m)$-dimensional matrix having (d,d')th entry $z_{\ell-1}^{(d',d)}$.

Now, suppose that the mutational model is symmetric and the mutation rate constant for all loci. Then, $\mathcal{E}_\ell=\mathcal{E}$ does not depend on ℓ, for all non-polymorphic loci ℓ. Similarly, if the recombination rate between each pair of loci is constant, then $\mathcal{Y}_\ell=\mathcal{Y}$ and $\mathcal{Z}_\ell=\mathcal{Z}$ do not depend on ℓ. With these assumptions, for $\ell_s^* \le \ell \le \ell_e^*$,

$$\begin{aligned}\mathcal{Q}_\ell &= \mathcal{E}(\mathcal{Y}+\mathcal{Z})\mathcal{Q}_{\ell-1} = (\mathcal{E}(\mathcal{Y}+\mathcal{Z}))^{\ell-\ell_s^*+1}\mathcal{Q}_{\ell_s^*-1},\\ \mathcal{T}_\ell &= \mathcal{E}\mathcal{Y}\mathcal{T}_{\ell-1} = (\mathcal{E}\mathcal{Y})^{\ell-\ell_s^*+1}\mathcal{T}_{\ell_s^*-1},\end{aligned} \qquad (17)$$

and the values of $(\mathcal{E}(\mathcal{Y}+\mathcal{Z}))^r$ and $(\mathcal{E}\mathcal{Y})^r$ can be pre-computed (either by eigenvalue decomposition or repeated multiplication) for a relevant range of r-values. Using this technique for explicitly computing only the necessary values of $Q_\ell(d)$ and $T_\ell(d)$, stretches of non-polymorphic loci can be '*analytically*' skipped.

The modified dynamic program associated with this optimization is given in Algorithm 4. The time complexity of Line 4 is $O(m)$ and of Line 6 is $O(m+\psi_\ell(\mathcal{C})+\omega_\ell(\mathcal{C}))$. Thus, the time complexity for the dynamic program is $O(k_p m^2 + m(\Psi_p(\mathcal{C})+\Omega(\mathcal{C})))$.

This refinement once again reduces the computation required for non-polymorphic loci, and so we might expect substantial improvements in performance over Algorithms 2 and 3. For the choice $\mathcal{C}=C_T$, the time complexity is $O(k_p m(m+n_u))$. Note that the assumptions necessary for Algorithm 4, namely a symmetric mutation model and uniform mutation and recombination rates, can be relaxed, but at the expense of additional pre-computation. For example, given non-uniform, but locally similar, recombination rates, pre-computation might be performed for each of several rates; each stretch of non-polymorphic loci could then use the pre-computed values associated with the closest recombination rate.

2.6 A fast algorithm for configuration partitions

In Section 2.5, we assumed a configuration partition $\mathcal{C}$ to be specified and showed that, for Algorithms 2–4, the time complexity depends on $\mathcal{C}$ through the functions $\Psi(\mathcal{C})$ (or $\Psi_p(\mathcal{C})$) and $\Omega(\mathcal{C})$ and more particularly their sum. It is intuitively clear that a configuration partition minimizing Ω will maximize Ψ (as in C_T), and vice versa, and in general these quantities are inversely related; minimizing a convex combination of these quantities is therefore difficult. In this section, we propose a fast and simple parametrized algorithm for constructing reasonably good configuration partitions.

Given a configuration $\mathbf{n}$, the algorithm proceeds sequentially over the loci: initially, let $\ell_s=1$. Given ℓ_s, find the largest polymorphic locus ℓ_e such that $\ell_s \le \ell_e \le k$, and the number of unique sub-haplotypes between loci ℓ_s and ℓ_e is at most some threshold parameter s. Then, for each unique sub-haplotype x between ℓ_s and ℓ_e, group all $h\in\mathcal{H}_\mathbf{n}$ such that $h[\ell_s:\ell_e]=x$ into the same

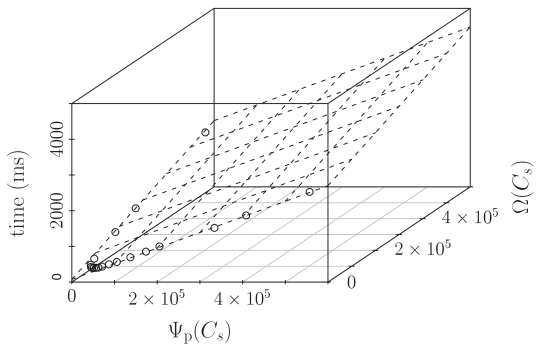

Fig. 3. Empirically observed running time of Algorithm 4 used to compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$, for a particular configuration $\mathbf{n}$ and an arbitrary $\alpha \in \mathcal{H}_{\mathbf{n}}$. Several values of $s \in (2,\ldots,500)$ were used, and each circle corresponds to a particular value of s. The curve of circles demonstrates the trade-off between small Ψ_p (small s-values) and small Ω (large s-values). As predicted by the asymptotic time complexity results, running time appears to depend linearly on both Ψ_p- and Ω-values, and fitting a linear model indicates the constant associated with Ψ_p is $\sim$1.5 times greater than the constant associated with Ω. The configuration $\mathbf{n}$ was generated using coalescent simulation for 500 individuals, each having 10^5 bi-allelic loci, using population-scaled mutation rate $\theta = 0.005$ per locus and population-scaled recombination rate $\rho = 0.001$ between each pair of adjacent loci, and resulting in $k_p = 1724$ polymorphic loci and $n_u = 324$ unique haplotypes

block $\mathcal{B}$ and add $(\mathcal{B}, \ell_s, \ell_e)$ to the configuration partition. Set $\ell_s = \ell_e + 1$ and repeat until locus k is reached. An example configuration partition resulting from this algorithm is shown in Figure 2c.

Applying this procedure to configuration $\mathbf{n}$ with threshold parameter s results in a configuration partition which we denote C_s. Observe that for $s = |\mathcal{H}_{\mathbf{n}}|$, we obtain $C_s = C_T$, which minimizes Ω. On the other hand, for $s = 2$ (in the bi-allelic case), the algorithm produces a configuration partition that minimizes Ψ_p. Intermediate values of s produce the intermediate results that are of interest. In order to gauge the effect of different combinations of Ψ_p and Ω on the running time, the CSP $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ was computed using the configuration partition C_s for each of several values of s, and the times recorded; the results are plotted in Figure 3. As our intuition suggested, the running time depends substantially on the choice of $\mathcal{C}$ and, in accordance with the asymptotic time complexity results depends linearly on both Ψ_p and Ω. Furthermore, as anticipated, the values of Ψ_p and Ω are inversely related.

By fitting a linear model to the data, we can deduce the constants associated with Ψ_p and Ω, which the asymptotic results cannot provide. Although these constants will depend on the implementation and hardware, their ratio should be relatively stable, and therefore informative for choosing an optimal trade-off between Ψ_p and Ω. We find that the constant associated with Ψ_p is $\sim$1.5 times that associated with Ω, suggesting that running time is minimized for a choice of $\mathcal{C}$ that minimizes $1.5 \cdot \Psi_p(\mathcal{C}) + \Omega(\mathcal{C})$. Furthermore, making use of the above algorithm, we define

$$s^* = \underset{s}{\operatorname{argmin}} \left\{ 1.5 \cdot \Psi_p\big(C_s\big) + \Omega\big(C_s\big) \right\},$$

and $C^* = C_{s^*}$. In practice the value s^* is found using binary search and determining C^* is very fast. This definition will be used frequently in Section 4, as C^* (and the analogous result for Algorithm 1, using Ψ in place of Ψ_p) provides a good, though not necessarily optimal, choice for $\mathcal{C}$.

3 RESULTS

We have presented three optimized algorithms for computing the conditional sampling probability (CSP) $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$. Briefly, Algorithms 2–4 rely on a partition $\mathcal{C}$ of the configuration $\mathbf{n}$. We have characterized optimal such partitions and proposed a simple and fast method for constructing good partitions $\mathcal{C} = C^*$ (cf, Section 2.6). For the sake of comparison, we also consider the trivial partition $\mathcal{C} = C_T$. Relative to Algorithm 2, Algorithms 3 and 4 represent successive improvements in efficiency for non-polymorphic loci. In this section, we provide an empirical analysis of these algorithms and demonstrate that our optimizations yield a substantial speedup.

Table 1. A summary of the proposed algorithms along with their asymptotic time complexities

	$\mathcal{C} = C_T$	General $\mathcal{C}$
Algorithm 2	$O(km(m+n_u))$	$O(km^2 + m(\Psi(\mathcal{C}) + \Omega(\mathcal{C})))$
Algorithm 3	$O(km^2 + k_p m n_u)$	$O(km^2 + m(\Psi_p(\mathcal{C}) + \Omega(\mathcal{C})))$
Algorithm 4	$O(k_p m(m+n_u))$	$O(k_p m^2 + m(\Psi_p(\mathcal{C}) + \Omega(\mathcal{C})))$

Recall that Algorithm 2 with $\mathcal{C} = C_T$ is equivalent to Algorithm 1.

The optimized algorithms, along with their asymptotic time complexities, are summarized in Table 1. Intuitively, for a fixed number of haplotypes n, and assuming coarse homogeneity across the genome, the runtimes of each of these algorithms should be linear in the number of loci. We are interested in determining the constants associated with this linear behaviour for each algorithm. Note, however, that for the cases when $\mathcal{C} = C_T$, the time complexities do not depend on n directly, but rather the number of unique haplotypes n_u. For a particular value of n, the quantity n_u will increase with the number of loci under consideration until $n_u = n$; only at this point do the runtimes become linear in the number of loci. A similar argument can be made for a more general configuration partition $\mathcal{C}$. In order to attain and analyse the linear behaviour for the modestly sized configurations that are considered, we formally interpret even non-unique haplotypes to be unique, thereby forcing $n_u = n$.

We produce data using coalescent simulation: we assume a symmetric 2-allele model and with population-scaled mutation rate $\theta = 0.005$ per locus and population-scaled recombination rate $\rho = 0.001$ between each pair of adjacent loci. For each of several values of n, we thus simulate an n-haplotype 2×10^5-locus configuration $\mathbf{n}$. We compute the partitions C_T and C^* and subsequently record the running time of each algorithm in computing $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$, for an arbitrary haplotype $\alpha \in \mathcal{H}_{\mathbf{n}}$. Throughout, we use a time discretization consisting of $m = 16$ intervals. The running times are plotted, on a logarithmic scale, as a function of n in Figure 4a and 4b, for $\mathcal{C} = C_T$ and $\mathcal{C} = C^*$, respectively.

From Figure 4a, for which $\mathcal{C} = C_T$, it is clear that our refinements for non-polymorphic loci have practical benefits, as Algorithms 3 and 4 perform substantially better than Algorithm 2. The asymptotic results summarized in Table 1 suggest the running time of Algorithm 4 is a factor of k/k_p faster than Algorithm 2. This factor is roughly reflected in the logarithmic plot of Figure 4a as a vertical shift, with deviations occurring because k_p increases (slowly) with n. Similarly, as n increases, the asymptotic results indicate that computation is dominated by the $O(k_p nm)$ term for both Algorithms 3 and 4; this is reflected in Figure 4a by a near identity in running times for these algorithms for larger values of n.

Comparing Figure 4b to a, the benefits of taking $\mathcal{C} = C^*$ can be observed. For each algorithm, this optimization improves

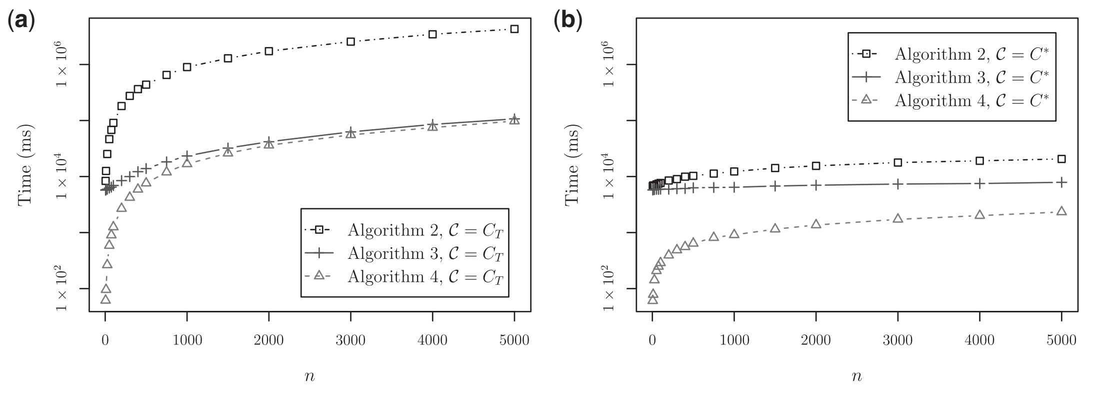

Fig. 4. Log-scaled plots of the running time (in milliseconds) required to compute $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ for $\mathbf{n}$ with 2×10^5 loci and $|\mathbf{n}| = n$, as a function n, for each of Algorithms 2–4. Configurations were generated using coalescent simulation as described in the text and results obtained on a single core of a MacPro with dual quad-core 3.0 GHz Xeon CPUs. (**a**) $\mathcal{C} = C_T$, the trivial configuration partition. (**b**) $\mathcal{C} = C^*$, the configuration partition described in Section 2.6

Table 2. A summary of several key statistics from Figure 4

	$\mathcal{C} = C_T$		
	$n = 100$	$n = 2000$	$n = 5000$
Algorithm 2	45 (1.0×)	870 (1.0×)	2153 (1.0×)
Algorithm 3	3.5 (13×)	21 (41×)	54 (40×)
Algorithm 4	0.63 (71×)	18 (48×)	49 (44×)
	$\mathcal{C} = C^*$		
	$n = 100$	$n = 2000$	$n = 5000$
Algorithm 2	3.8 (12×)	7.8 (110×)	10.3 (208×)
Algorithm 3	3.0 (15×)	3.5 (250×)	3.9 (546×)
Algorithm 4	0.14 (320×)	0.68 (1300×)	1.17 (1845×)

The table indicates the time (in seconds) required to compute the CSP $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$ for $|\mathbf{n}| = n$, per 1×10^5 loci. The speed-up versus Algorithm 2 with $\mathcal{C} = C_T$, equivalent to the commonly used Algorithm 1, is given in parentheses.

performance substantially, particularly as the number of haplotypes n increases. Given the results for Algorithm 3 in particular, it is clear that the key quantity $\Psi_{\text{p}}(\mathcal{C}) + \Omega(\mathcal{C})$, taken from Table 1, increases more slowly with n for $\mathcal{C} = C^*$ than for $\mathcal{C} = C_T$. Finally, as in the previous case, the asymptotic results for general $\mathcal{C}$ indicate that computation is dominated by the $O(m(\Psi_{\text{p}}(\mathcal{C}) + \Omega(\mathcal{C})))$ term for both Algorithms 3 and 4; the associated convergence of running times appears to be occurring in Figure 4b, though more slowly than in Figure 4a; thus, Algorithm 4 is a practically useful alternative to Algorithm 3, even for larger values of n.

Although general trends are clear from Figure 4, the logarithmic scale makes it difficult to appreciate the magnitude of the effects of the optimizations. As mentioned earlier, assuming rough homogeneity over the genome, the computation time increases linearly with the number of loci. In Table 2, we summarize the constant associated with this linear behaviour as the time required to process 1×10^5 loci, along with the speedup relative to the baseline, Algorithm 1. Observe that Algorithm 3, with $\mathcal{C} = C^*$, which can be applied in complete generality, provides a speedup of 15×, 250× and 546× for conditional sample sizes $n = 100, 2000$, and 5000, respectively; and in most cases, Algorithm 4 can be applied, which increases these speedups to 320×, 1300× and 1845×. Importantly, the speedup increases with the conditional sample size n.

4 DISCUSSION

We have presented a number of optimized algorithms for computing the CSP $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$. Our optimizations are based on two intuitive observations: first, the number of unique haplotypes in a genomic sample is dramatically reduced within relatively short regions and second, the large number of non-polymorphic loci in a genomic sample, though informative, do not distinguish between haplotypes. These observations are formalized and leveraged to refine the recursive equations for computing $\hat{\pi}_{\text{SMC}}(\alpha|\mathbf{n})$, yielding optimized, yet exact, algorithms.

We have described our optimization algorithms in the context of the HMM associated with the CSD $\hat{\pi}_{\text{SMC}}$ proposed by Paul *et al.* (2011). It is natural to question whether similar optimizations are applicable to related CSDs, such as those proposed by Fearnhead and Donnelly (2001) and by Li and Stephens (2003). In Section 2, we described two sufficient conditions: Property 1, which stipulates that, upon recombination, a new hidden haplotype is chosen independently and uniformly at random and Property 2, which stipulates that the emission distribution depends only on the allele at the current locus of the hidden haplotype. The aforementioned CSDs do satisfy both of these properties; in particular, stronger forms of Property 1 hold for both CSDs, enabling additional optimizations. We have not empirically analysed the resulting optimized algorithms, but by considering the resulting asymptotic time complexities, analogous to those in Table 1, we anticipate that the speedups obtained will be qualitatively comparable to those observed for $\hat{\pi}_{\text{SMC}}$, though the corresponding magnitudes are difficult to estimate.

It is also interesting to consider CSDs for more complex demographic scenarios. A theoretically straightforward extension of $\hat{\pi}_{\text{SMC}}$ to variable population size, for example, will continue to satisfy both Properties 1 and 2 and will therefore be amenable to very similar optimizations. On the other hand, extension to structured populations, populations that are divided into several demes between which there is limited migration, will not satisfy Property 1 as the new hidden haplotype chosen upon recombination depends on the deme in which recombination occurs. Nonetheless, a relaxed version of Property 1 will be satisfied along with Property 2, and we anticipate analogous optimizations will be possible. The outcome is similar if $\hat{\pi}_{\mathrm{SMC}}$ is extended to conditionally sampling diploid, rather than haploid, individuals.

Related optimizations may be possible in other contexts as well. For example, Li and Durbin (2011) make use of a population genetic HMM which satisfies conditions that correspond to Properties 1 and 2 and is used to analyse genomic data. It is interesting to note that, in order to make their method practicable, Li and Durbin consider non-overlapping 100 bp windows as their set of loci; using the optimization detailed in this article may render such compromises unnecessary. It is less clear whether our optimizations are applicable to other population genetic HMMs, such as those considered by Hobolth *et al.* (2007) and Dutheil *et al.* (2009); nonetheless, we hope that our work will foster progress in this area.

We conclude by recalling that a broad range of population genetic methods have been developed and will continue to be developed, based on the CSD. These methods are generally computationally intensive, and approximations are often made on the basis of efficiency and at the expense of accuracy; with the advent of inexpensive genomic sequencing, such computational problems will be compounded. We have introduced several optimizations for CSD computation that can potentially speed up this computation by several orders of magnitude without introducing additional approximations. We believe that these optimizations will enable analyses that were previously impracticable, particularly for large genomic datasets. We also hope that the optimizations will encourage more accurate methods, and in particular more accurate CSDs, to be developed and used for population genomic inference.

Acknowledgements

We thank Anand Bhaskar and Matthias Steinrücken for fruitful discussion on both the theoretical and practical aspects of this work.

Funding: NIH National Research Service Award Trainee appointment (T32-HG00047) to JSP and an NSF CAREER Grant (DBI-0846015) and a Packard Fellowship for Science and Engineering to YSS.

Conflict of Interest: none declared.

REFERENCES

Browning,B.L. and Browning,S.R. (2007) Rapid and accurate haplotype phasing and missing data inference for whole genome association studies using localized haplotype clustering. *Am. J. Hum. Genet.*, **81**, 1084–1097.

Crawford,D.C. *et al.* (2004) Evidence for substantial fine-scale variation in recombination rates across the human genome. *Nat. Genet.*, **36**, 700–706.

Davison,D. *et al.* (2009) An approximate likelihood for genetic data under a model with recombination and population splitting. *Theor. Popul. Biol.*, **75**, 331–345.

De Iorio,M. and Griffiths,R.C. (2004a) Importance sampling on coalescent histories. I. *Adv. Appl. Prob.*, **36**, 417–433.

De Iorio,M. and Griffiths,R.C. (2004b) Importance sampling on coalescent histories. II: Subdivided population models. *Adv. Appl. Prob.*, **36**, 434–454.

Delaneau,O. *et al.* (2012) A linear complexity phasing method for thousands of genomes. *Nat. Methods*, **9**, 179–181.

Dutheil,J.Y. *et al.* (2009) Ancestral population genomics: the coalescent hidden markov model approach. *Genetics*, **183**, 259–274.

Fearnhead,P. and Donnelly,P. (2001) Estimating recombination rates from population genetic data. *Genetics*, **159**, 1299–1318.

Fearnhead,P. and Donnelly,P. (2002) Approximate likelihood methods for estimating local recombination rates. *J. Royal Stat. Soc. B*, **64**, 657–680.

Fearnhead,P. and Smith,N.G. (2005) A novel method with improved power to detect recombination hotspots from polymorphism data reveals multiple hotspots in human genes. *Am. J. Hum. Genet.*, **77**, 781–794.

Gay,J. *et al.* (2007) Estimating meiotic gene conversion rates from population genetic data. *Genetics*, **177**, 881–894.

Griffiths,R.C. *et al.* (2008) Importance sampling and the two-locus model with subdivided population structure. *Adv. Appl. Probab.*, **40**, 473–500.

Hellenthal,G. *et al.* (2008) Inferring human colonization history using a copying model. *PLoS Genet.*, **4**, e1000078.

Hobolth,A. *et al.* (2007) Genomic relationships and speciation times of human, chimpanzee, and gorilla inferred from a coalescent hidden markov model. *PLoS Genet*, **3**, e7.

Howie,B.N. *et al.* (2009) A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. *PLoS Genet.*, **5**, e1000529.

Hudson,R.R. (2001) Two-locus sampling distributions and their application. *Genetics*, **159**, 1805–1817.

Lawson,D. *et al.* (2012) Inference of population structure using dense haplotype data. *PLoS Genet.*, **8**, e1002453.

Li,H. and Durbin,R. (2011) Inference of human population history from individual whole-genome sequences. *Nature*, **475**, 493–496.

Li,N. and Stephens,M. (2003) Modelling linkage disequilibrium, and identifying recombination hotspots using SNP data. *Genetics*, **165**, 2213–2233.

Li,Y. *et al.* (2010) Mach: Using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genet. Epidemiol.*, **34**, 816–834.

Marchini,J. *et al.* (2007) A new multipoint method for genome-wide association studies by imputation of genotypes. *Nat. Genet.*, **39**, 906–913.

Marjoram,P. and Wall,J.D. (2006) Fast "coalescent" simulation. *BMC Genet.*, **7**, 16.

McVean,G.A.T. *et al.* (2004) The fine-scale structure of recombination rate variation in the human genome. *Science*, **304**, 581–584.

McVean,G.A. and Cardin,N.J. (2005) Approximating the coalescent with recombination. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, **360**, 1387–1393.

Paul,J.S. and Song,Y.S. (2010) A principled approach to deriving approximate conditional sampling distributions in population genetics models with recombination. *Genetics*, **186**, 321–338.

Paul,J.S. *et al.* (2011) An accurate sequentially markov conditional sampling distribution for the coalescent with recombination. *Genetics*, **187**, 1115–1128.

Price,A.L. *et al.* (2009) Sensitive detection of chromosomal segments of distinct ancestry in admixed populations. *PLoS Genet.*, **5**, e1000519.

Scheet,P. and Stephens,M. (2006) A fast and flexible method for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *Am. J. Hum. Genet.*, **78**, 629–644.

Stephens,M. and Donnelly,P. (2000) Inference in molecular population genetics. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **62**, 605–655.

Stephens,M. and Scheet,P. (2005) Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. *Am. J. Hum. Genet.*, **76**, 449–462.

Sundquist,A. *et al.* (2008) Effect of genetic divergence in identifying ancestral origin using HAPAA. *Genome Res.*, **18**, 676–682.

Wegmann,D. *et al.* (2011) Recombination rates in admixed individuals identified by ancestry-based inference. *Nat. Genet.*, **43**, 847–853.

Yin,J. *et al.* (2009) Joint estimation of gene conversion rates and mean conversion tract lengths from population SNP data. *Bioinformatics*, **25**, i231–i239.