MOLECULAR ECOLOGY
RESOURCES
Molecular Ecology Resources (2015) doi: 10.1111/1755-0998.12437

IMa2p – parallel MCMC and inference of ancient demography under the Isolation with migration (IM) model

ARUN SETHURAMAN and JODY HEY
Center for Computational Genetics and Genomics, Department of Biology, Temple University, Philadelphia, PA 19102, USA

Abstract

IMa2 and related programs are used to study the divergence of closely related species and of populations within species. These methods are based on the sampling of genealogies using MCMC, and they can proceed quite slowly for larger data sets. We describe a parallel implementation, called IMa2p, that provides a nearly linear increase in genealogy sampling rate with the number of processors in use. IMa2p is written in OpenMPI and C++, and scales well for demographic analyses of a large number of loci and populations, which are difficult to study using the serial version of the program.

Keywords: Isolation with migration, MCMC, Metropolis coupling, message passing interface, Parallelization

Received 28 January 2015; revision received 3 June 2015; accepted 4 June 2015

Introduction

Isolation with migration (IM) models are used to study the divergence of species and populations in situations where investigators are interested in the roles played by both the time since separation and the rate of gene exchange between populations (Wakeley & Hey 1998; Nielsen & Wakeley 2001). As implemented in the program IMa2, analyses proceed by running a Markov chain Monte Carlo (MCMC) simulation that generates samples of genealogies, G, and splitting times, t, from the posterior distribution $p(G,t \mid data)$. However, with larger data sets, the exploration of the state space $\{G,t\}$ by MCMC can proceed very slowly To shorten runtimes, it is common to use a method in which an unheated (cold) chain is run simultaneously with multiple heated chains, and a Metropolis update is used to swap state spaces between chains. This method was developed independently by investigators in different fields (Swendsen & Wang 1986; Geyer 1991; Kimura & Taki 1991; Hansmann 1997) and goes by several names – here we will use Metropolis-coupled MCMC, or *MC*3 (Geyer 1991).

As implemented in IMa2, *MC*3 can decrease the overall run time, even as the computation rate per chain is reduced, because of the improved mixing that occurs with the addition of multiple chains. However, because the IMa2 program runs on a single processor, run times are often still quite long, and the method is often not practical for large data sets. Here, we describe a parallel implementation of IMa2 that distributes chains over processors using the Message Passing Interface (MPI). Some caveats with parallelization using a synchronization algorithm are also discussed.

Correspondence: Arun Sethuraman, Fax: +215-204-6646;
E-mail: arun@temple.edu

Methods

IMa2 algorithm

Hey & Nielsen (2007) described a version of the Felsenstein equation (Felsenstein 1988) for the posterior probability of the parameters of a two-population Isolation with migration model, including population sizes, migration rates and divergence times. Under a two-population model, the parameters to be estimated are the splitting time, t, and a set of coalescent and migration rates $\tau = \{\Theta_1, \Theta_2, \Theta_a, m_{12}, m_{21}\}$ where Θ_1, Θ_2 and Θ_a, are the population mutation rates for the sampled populations and the ancestral population, respectively. The splitting time t is when the ancestral population of size Θ_a split into populations 1 and 2. Bidirectional migration rates between the populations are described by the parameters m_{12} and m_{21}. The posterior probability distribution of the rate parameters, given the data, X, can then be written as:

$$p(\tau|X) = \int_{\Psi} p(\tau|G)p(G|X)dG \qquad \text{eqn 1}$$

where G is a coalescent genealogy with migration events, and Ψ is the space of possible genealogies. Expression

(1) can be approximated by first running a Markov chain simulation over splitting times and genealogies, with updates decided by a Metropolis–Hastings (MH) criterion:

$$\min\left\{1, \frac{p(X|G^*,t^*)p(G^*,t^*)g(G^*,t^* \to G,t)}{p(X|G,t)p(G,t)g(G,t \to G^*,t^*)}\right\} \qquad \text{eqn 2}$$

Then, using a sample of genealogies and splitting times $p(G,t \mid X)$, expression (1) can be approximated as:

$$p(\tau|X) \approx \frac{1}{k}\sum_{i=1}^{k} p(\tau|G_i,t_i) \qquad \text{eqn 3}$$

In practice, an IMa2 analysis has two steps or modes, including 'M' (MCMC), during which genealogies are sampled from the posterior distribution $p(G,t \mid X)$, and 'L' (load genealogies), during which the posterior distribution of all rate parameters (τ), given the data (X) are obtained by the approximation in (eqn 3).

Metropolis-coupled MCMC

In *MC*3, there are n Markov chains with heating parameters, or 'temperatures', $\beta_1, \beta_2, \ldots, \beta_n$, where $0 < \beta \leq 1$. For the unheated chain 1, $\beta_1 = 1$ and the stationary distribution is the desired posterior density distribution, while all other chains track stationary distributions that are proportional to that of the unheated chain, raised to the power of their corresponding temperature. In each iteration, the state spaces of all chains are updated using eqn 3, after which MC3 updates are attempted by swapping the state spaces (or, equivalently, the β values) of two chains at random with acceptance probability

$$\min\left\{1, \frac{p(G_x,t_x|X)^{\beta_x}\, p(G_y,t_y|X)^{\beta_y}}{p(G_y,t_y|X)^{\beta_x}\, p(G_x,t_x|X)^{\beta_y}}\right\}, \qquad \text{eqn 4}$$

where x and y are two randomly chosen chains.

IMa2p algorithm

The *MC*3 algorithm lends itself well to parallelization, with different chains running on different processors. Altekar *et al.* (2004) describe two synchronization schemes ('Global' and 'Point-to-point') that can be used to ensure that chains are synchronized, and that swaps are only attempted between chains that are in the same iteration. We implemented a combination of both schemes to maintain synchrony throughout the run. Under the 'global' scheme, processors are synchronized by calling a 'barrier' operation (MPI::Barrier in the Open-MPI framework), which forces processors to wait until parallel communication is complete before proceeding to the next iteration of MCMC. Under the 'Point-to-point' scheme, exchange sequence is shared among processors, thus allowing processors not involved in an MPI operation to proceed onto the next iteration without having to wait. We used both of these by exchanging information between processors (i.e. 'Point-to-point') for heat swapping, and using MPI::Barrier operations (i.e. 'Global') for obtaining information about the progress of the run.

In brief, updated genealogies and divergence times are proposed on each chain in the MCMC. All chains then either accept or reject the proposed genealogy and divergence time using the Metropolis–Hastings criterion, defined in eqn 3. After the update step in every chain, two chains are randomly chosen to attempt a temperature swap. One of the chosen chains computes the swap acceptance MH term (eqn 4), notifies the other of success or failure of the swap attempt, upon which both chains swap their temperatures from temporary swap holders (to prevent race conditions). For every other chain that is not involved in a swap operation, MCMC operations continue until the next iteration in which that chain is involved in a swap operation.

Mixing and convergence

At regular intervals, an MPI barrier is imposed so that summary information on acceptance of swaps between chains and on overall convergence of the cold chain can be collated by the head node and to ensure synchrony such that all chains are in the same generation (see Algorithm 2).

We assessed mixing and the approach to stationarity by estimating the autocorrelation and effective sample sizes (ESS) of quantities sampled from the state space of the unheated chain (Geyer 1991). See Appendix 1, Algorithm 3 for details on autocorrelation assessment.

Synchronization

Two common problems associated with synchronization of *MC*3 are (i) deadlocks, wherein a processor is left waiting for swap information from another processor that will not be sent, and (ii) race conditions, wherein two processors attempt to access swap information from the same memory location. We avoid deadlocks by deciding the sequence of attempted swaps ahead of time (Altekar *et al*. 2004). Adhering to the exchange rule, all nodes use the same sequence and two chains only attempt to swap if they are both in the swap sequence at that iteration, while other chains continue with their updates. Race conditions are avoided by the use of swap holders as proposed by Altekar *et al*. (2004). Prior to swapping in any iteration, temperatures are stored in temporary swap

holders, and chains of temperatures are swapped with these holders upon swap acceptance.

Simulations

Our primary question, in assessing performance, is how the overall speed of calculations changes as a function of the number of processors. Because of the need to maintain synchrony among chains, we expect a less than linear response as the number of chains increases. If the time required to update individual chains varies little among chains, then the synchronization requirement may add little overhead. This last point raises a second question, if we run a model that is expected to give a high variance in the time to update individual chains, will we see a corresponding departure in performance (as measured as a departure from a linear relationship between rate of calculation and number of processors)? All simulations were performed using MS (Hudson 2002) under a two-population model (see Simulations in Appendix 1).

For the first set of simulations (designed to analyse computational speed-ups using IMa2p), we varied the number of loci (5, 50 and 300 loci) for a model with 15 gene copies sampled from each of two populations. Model parameters were set as follows: $\Theta_1 = \Theta_2 = \Theta_a = m_{12} = m_{21} = 1$; and $t = 0.01$. Five independent runs were made for each data set, each using 60 Metropolis-coupled chains, distributed among 1–20 processors.

The second set of simulations was designed to assess the circumstances when the departure from a linear improvement may be greatest. We expect this to occur for models that invoke a high variance among chains in time required to complete a genealogy update. A single locus was simulated with 50 gene copies from each population, and the analyses conducted with a large upper bound on the migration rate prior distribution ($m = 1, 10, 100$), and the likelihood ($p(X \mid G)$) set to a constant. This has the effect of causing the data to be ignored and causes the target posterior density to equal the prior distribution of the model parameters. In this way, we cause the number of migration events in the genealogies to vary widely, and because calculations require more time for genealogies with many migration events, we increase the variance among chains in required computing time per update. As processors are synchronized at the end of each iteration, we expect that the greater the number of chains, the more time spent in waiting for another chain to finish its computations. Five independent runs were performed, and the number of processors varied between 1 and 10.

Additional simulations were conducted to confirm that varying the number of processors does not affect the target density. Here, we show the results for one data set and a varying number of processors. A data set was generated with 2 loci and 15 individuals from each population of size with parameters as follows: $\Theta_1 = \Theta_2 = \Theta_a = 5$, $m_{12} = m_{21} = 2$; and $t = 0.2$. Parameter estimates from parallel (2–20 processors) and serial (1 processor) runs were compared by plotting marginal posterior density estimates of parameters across duplicate MCMC runs. Two tailed Kolmogorov–Smirnov (KS) tests were performed between cumulative posterior densities obtained for each parameter to assess whether the distributions obtained varied with the number of processors. The null hypothesis (H_0) for KS tests of equality was that the cumulative posterior density distributions of each parameter estimated using x processors was the same as the distribution estimated using y processors, where x and y were one of 1–20 processors. The alternate hypothesis (H_a) was that two distributions using x and y processors were not equal. Analyses were run using a burn-in period of 100,000 iterations, and 20,000 genealogies were sampled in steps of 1000 after burn-in (total of 2×10^7 iterations after burn-in). Bonferroni correction for multiple tests (across all pairs of number of processors – total of 15 pairwise comparisons) was performed on thus obtained P-values at a threshold of $P = 0.05$.

Empirical data

To demonstrate IMa2p on empirical data, and to compare run times against the serial version of IMa2, we measured computational times across replicate runs of IMa2p and IMa2 using a data set of 48 loci from two chimpanzee subspecies (*Pan troglodytes troglodytes* and *P. t. verus*) (Won & Hey 2005). We used the identical settings as originally published for prior distributions, number of chains in MCMC and number of genealogies saved (see Appendix 1). We varied the number of processors between 1 and 10, with five replicate runs of IMa2p, and measured the computational time required for the entire run in each case. We also compared computational time required by IMa2p against that of a serial replicate run of IMa2.

All simulation analyses were performed on the HIGHMEM (Dell R610, 8 nodes, 2× Intel Xeon X5677, 3.5 GHz, 8 cores/node) or BIGMEM (Supermicro H8QG6, 3× AMD Opteron 6238, 2.6 GHz, 42 cores/node) nodes of Temple University's Owlsnest HPC server. Analyses of the empirical data were performed on a Dell Precision T5610 desktop (INTEL XEON E5-2620, 2.10 GHz, 12 cores).

Results

Computation time and number of processors

Plots of computational time (number of MCMC iterations per minute – see Figs 1 and 2) depart modestly from a linear relationship with the number of processors,

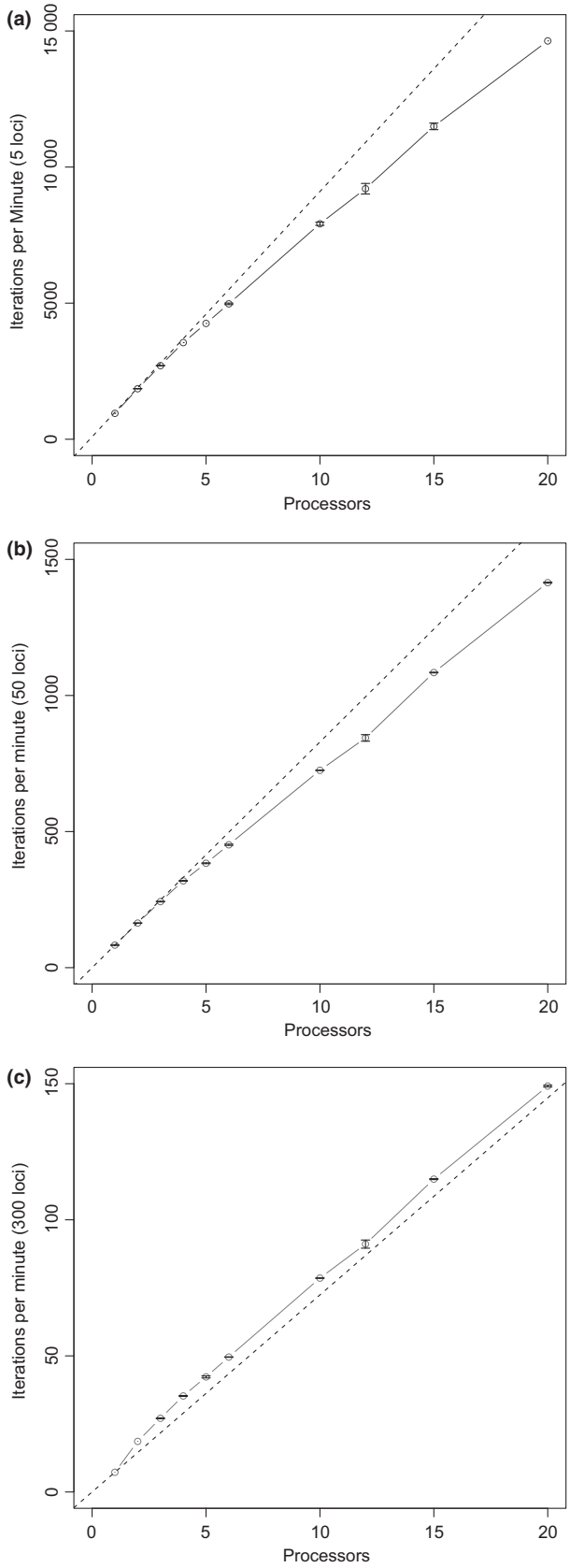

Fig. 1 Computation speeds measured in mean number of MCMC iterations per minute (over five replicates), versus number of processors. Error bars across replicate runs of IMa2p on the same data set with different random number seeds are shown over 95% confidence intervals, ignoring negligible standard errors (<0.001). Panels a, b and c show results computed for 5 loci, 50 loci and 300 loci data, respectively.

for simulated data sets with 5, 50 and 300 loci, and for the chimpanzee data set from Won & Hey (2005). The departure from linearity is less for data sets with more loci and is essentially absent for the largest data sets. On average, larger data sets would be expected to have lower communication-to-computation time ratios (and larger ratios for smaller data sets). However, as the number of processors increases, the communication time (time spent on MPI operations) also increases. Correspondingly, we would expect a communication-to-computation time ratio closer to 1 at larger number of processors for larger data sets. The total times required for the analyses in Fig. 1 varied strongly with the number of loci. The 5-locus data set completed its run in ~10.5 min on a single processor, as against 41 s using 20 processors. The 50-locus data set completed its run in ~2 h on a single processor and ~7 min using 20 processors. The 300-locus data set on the other hand required ~23 h on a single processor, but ~1 h on 20 processors.

Analysis of the data set from Won & Hey (2005) for a total of 200,000 iterations using 30 chains required ~4 h using the serial IMa2 program and on a single processor using IMa2p, while the same computation was completed in ~33 min when using 10 processors. The number of MCMC iterations per minute completed by the serial IMa2 program was also nearly identical to that completed by the parallelized IMa2p program using a single processor (see Fig. 2).

Departure from linearity under high maximal migration rates

We expect a greater departure from a linear relationship between computational time (here measured as total number of MCMC iterations per minute) and the number of processors under models with a high variance among processors in computational time, because of the increased waiting time for synchronization. In other words, synchronization costs would be expected to be greater during the 'global' exchange routines of the IMa2p algorithm (where data pertaining to mixing and convergence are collated onto the head node). We observe a greater cost of synchronization, as a departure from a linear relationship between iterations per minute and number of processors, for a model designed to have a high variance among loci in computation time (see

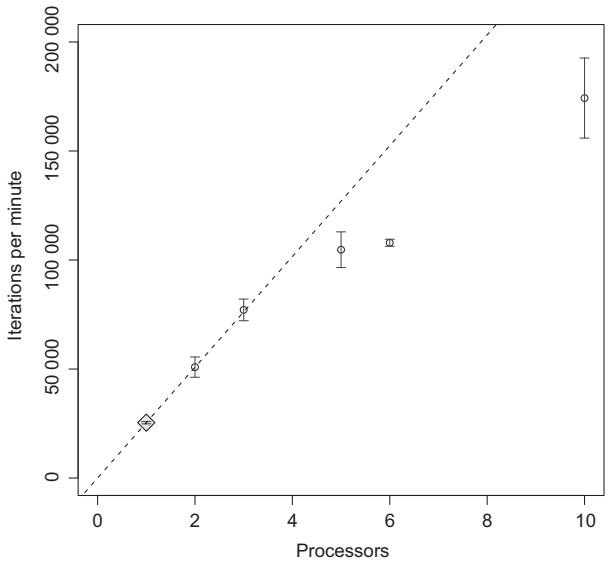

Fig. 2 Number of processors versus total number of updates of MCMC (number of iterations × number of chains) per minute. Error bars are show over five replicate runs of IMa2p on 47 genomic loci, obtained from Won & Hey (2005). The square point also indicates iterations per minute from five replicate runs of IMa2 on a single processor. These coincide with iterations per minute measured using 1 processor with the parallelized IMa2p.

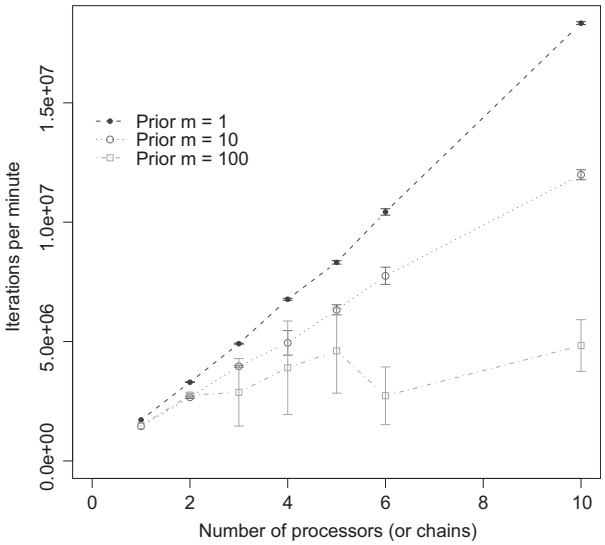

Fig. 3 Number of chains (also number of processors here) versus total number of updates of MCMC (number of iterations x number of chains) per minute. Error bars are shown over 5 replicate runs over the same single locus data set comprising 50 gene copies (individuals) sampled from one of two populations.

Fig. 3). This figure also reveals a higher variance in iterations per minute, as measured across replicate runs, with higher maximal migration rates, which is also consistent with a higher variance among chains within runs.

Equality of distributions for varying number of processors

Estimates of marginal posterior density distributions of all parameters were visibly consistent across separate MCMC runs across 1–20 processors, while maintaining the same number of chains (Figs 4 and 5). Kolmogorov–Smirnov (KS) tests (after Bonferroni correction for multiple tests) show congruence of all cumulative posterior density distributions ($P = 1.0$) of population sizes, divergence times and migration rates. Similarly, estimates of parameters obtained by IMa2p for the data of Won & Hey (2005) by varying the number of processors used were not distinguishable from those reported by Won & Hey (2005) (see Table A1).

Discussion

IMa2 and its precursors can be slow to converge on the target density and sometimes require lengthy runtimes, particularly with larger data sets. To shorten the time required for analyses, we have implemented a parallelized version of IMa2. For a wide range of data set sizes, we show that there is a considerable speed-up in computation achieved with increasing the number of processors (Figs 1 and 2). For the larger data sets, when parallelization is most needed, the number of MCMC updates increases nearly linearly with the number of processors.

The observation of a nearly linear relationship with larger sample sizes was also observed by Altekar *et al.* (2004) in their analyses of Leviviridae (small data set), and *Astragalus* (large data set). In smaller data sets, the computation time for the likelihood is less and so the wait times for swapping is relatively larger, particularly as the number of processors is increased. As the number of processors increases, the number of swaps per processor decreases, increasing the variance in the total number of swaps per processor, leading to longer wait times. In effect, there is a larger communication-to-computation time ratio for smaller data sets and large numbers of processors.

For data sets and models that introduce a wide variance in computation time among chains, we observed a greater departure from a linear increase in the update rate, and thus a reduced benefit of having additional processors (Fig. 3). However, the conditions used to generate this observation were those expected to generate a very large variance in numbers of migration events in the genealogies being simulated. For analyses with low upper bounds on the migration rate, or for data sets that dominate the prior distributions, this should not be an issue. Users of the program who

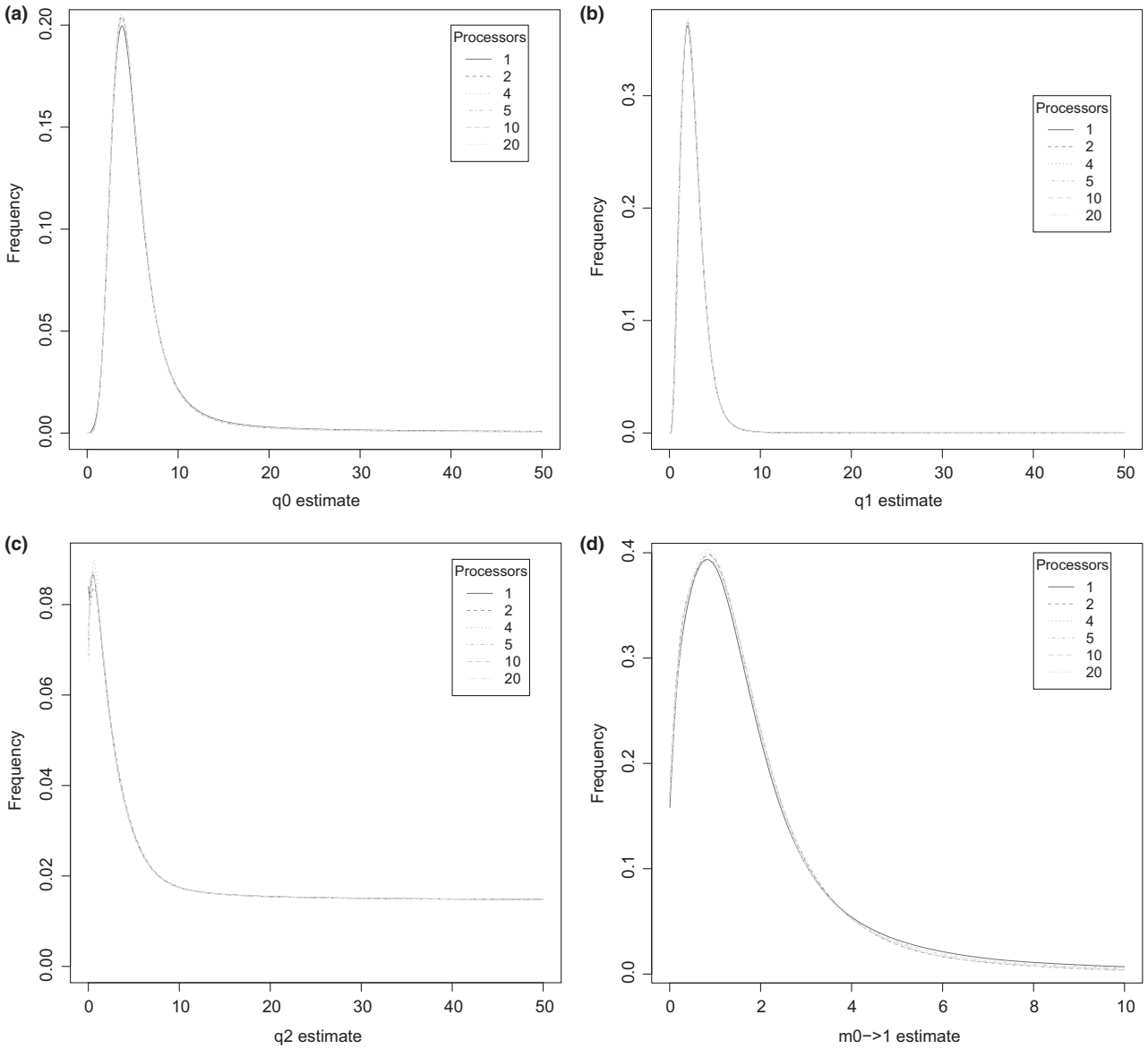

Fig. 4 Posterior density histograms of parameters (q_0 (a), q_1 (b), q_2 (c) are mutation scaled population sizes, $m_{0\to1}$ (d) is the mutation scaled migration rate from population 0 to population 1).

desire a largely uninformative prior on migration rate, and who are working with data that show some evidence of divergence, should avoid having high upper bounds (e.g. such that the maximum population migration rate is substantially >1) particularly for smaller data sets.

In our analyses, we considered including up to 20 processors, and it is possible that additional processors may show a reduced additional benefit. Feng *et al.* (2003), for instance, in their analysis of speedups in *P*(*MC*)3 algorithms in Bayesian phylogenetics note that linear speedups can be achieved using up to 28 processors. This trade-off has been previously attributed to the computational burden of interprocessor communications that is inherent to the MPI framework itself (see Altekar *et al.* (2004)). Optimizing the number of MPI communications between processors could offer a possible solution to this, as explored by Feng *et al.* (2006). Related to this is the computational overhead in swapping across processors when only one chain is run per processor, which results in greater computational time for MPI communication than with distributing swaps between and within processors, that is performing Metropolis coupling with >1 chain per processor. Users are advised to distribute more than 1 chain per processor when running IMa2p in parallel.

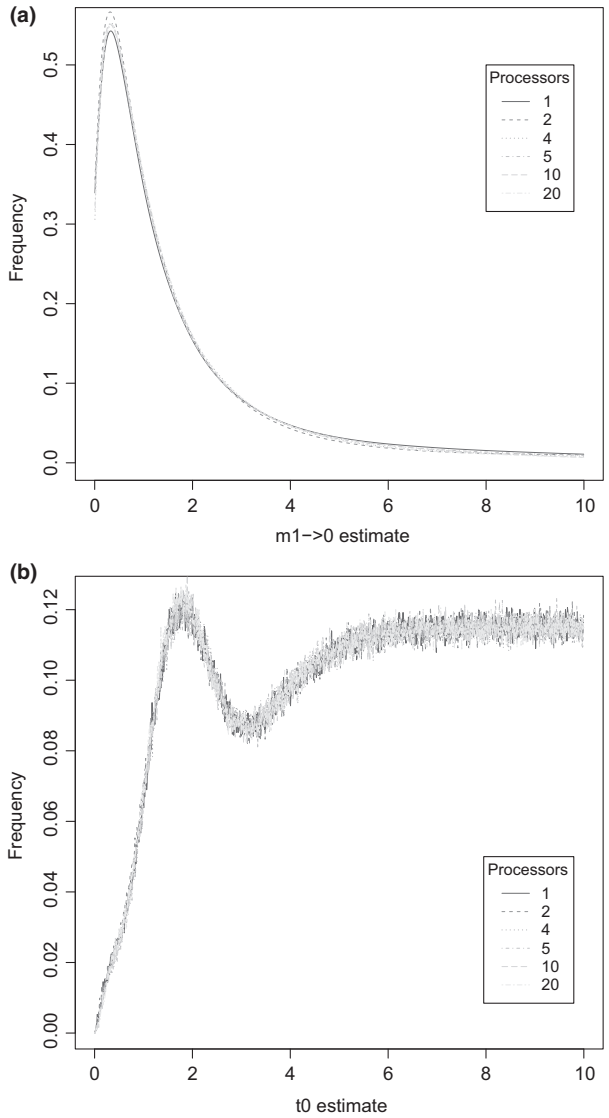

Fig. 5 Posterior density histograms of parameters ($m_{1\to 0}$ (a) is the mutation scaled migration rate from population 1 to population 0, and t_0 (b) is the divergence time (number of generations × mutation rate) between populations 0 and 1).

Acknowledgements

This research was supported by NIH Grant: R01GM078204 to Jody Hey. We thank Bryan Carstens, three anonymous reviewers and members of the Hey laboratory and the CCGG for their inputs on early versions of the manuscript and tool.

References

Altekar G, Dwarkadas S, Huelsenbeck JP, Ronquist F (2004) Parallel metropolis coupled Markov Chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, **242**, 407–415.

Felsenstein J (1988) Phylogenies from molecular sequences: inference and reliability. *Annual Review of Genetics*, **22**, 521–565.

Feng X, Buell DA, Rose JR, Waddell PJ (2003) Parallel algorithms for Bayesian phylogenetic inference. *Journal of Parallel and Distributed Computing*, **63**, 707–718.

Feng X, Cameron KW, Buell DA (2006) Pbpi: a high performance implementation of bayesian phylogenetic inference. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 75, ACM.

Geyer CJ (1991) *Markov chain Monte Carlo maximum likelihood*.

Hansmann UH (1997) Parallel tempering algorithm for conformational studies of biological molecules. *Chemical Physics Letters*, **281**, 140–150.

Hey J, Nielsen R (2007) Integration within the Felsenstein equation for improved Markov Chain Monte Carlo methods in population genetics. *Proceedings of the National Academy of Sciences*, **104**, 2785–2790.

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

Kimura K Taki K, (1991) Time-homogeneous parallel annealing algorithm. In: *Proceedings of the 13th IMACS World Congress on Computation and Applied Mathematics (IMACS'91)* (eds Vichnevetsky R, Miller JJH), **2**, pp. 827–828.

Nielsen R, Wakeley J (2001) Distinguishing migration from isolation: a Markov Chain Monte Carlo approach. *Genetics*, **158**, 885–896.

Swendsen RH, Wang JS (1986) Replica Monte Carlo simulation of spinglasses. *Physical Review Letters*, **57**, 2607.

Wakeley J, Hey J (1998) Testing speciation models with DNA sequence data. In: *Molecular Approaches to Ecology* (eds DeSalle R, Schierwater B), pp. 157–175. Birkhäuser-Verlag, Basel.

Won YJ, Hey J (2005) Divergence population genetics of chimpanzees. *Molecular Biology and Evolution*, **22**, 297–307.

Yu N, Jensen-Seaman MI, Chemnick L, *et al*. (2003) Low nucleotide diversity in chimpanzees and bonobos. *Genetics*, **164**, 1511–1518.

J.H. conceived the project. A.S. and J.H. designed the study. AS designed and wrote the IMa2p program based on the existing IMa2 code and carried out the analyses.

Data Accessibility

IMa2p uses OpenMPI and C++, and the source code, and simulated IMa2p input files from this manuscript can be downloaded from https://bio.cst.temple.edu/~hey/software/software.htm or www.github.com/arunsethuraman/ima2p. IMa2p was beta tested using the same standard testing modules as IMa2. Other functionality added to IMa2p that are different from the previous version includes (i) reloading of results of parallel runs to restart M mode/L mode runs, (ii) reporting autocorrelations and swap rates across processors, (iii) combining genealogies saved on different processors (as the cold or sampling chain can move around), (iv) option to swap entire chain state during swaps within a processor, as against swapping only temperatures. Instructions on compiling using OpenMPI and running the software in parallel are described in the user manual that can be accessed inside the code repository. For further details on IMa2, please see:https://bio.cst.temple.edu/~hey/program_files/IMa2/Using_IMa2_8_24_2011.pdf.

Sequence data from Won & Hey (2005) are available on GenBank under Accession nos AY275957 to AY277244 and AY463943 to AY463951.

Appendix 1

Algorithm 1: P(MC)3 algorithm

begin
  repeat
    Let current state of Markov Chain be ψ_i, for all n chains.
    for *all chains* ϵ (1, 2, …, n) **do**
      Randomly propose new genealogy, G^* and divergence time t^*.
      Calculate acceptance probability
      $R_i = min\left\{1, \frac{p(X|G^*,t^*)^{\beta} p(G^*,t^*)^{\beta} g(G^*,t^*\rightarrow G,t)}{p(X|G,t)^{\beta} p(G,t)^{\beta} g(G,t\rightarrow G^*,t^*)}\right\}$
      Generate uniform random variable, $U \epsilon$ (0, 1).
      if $U < R_i$ **then**
        Accept G^*, t^*
      end
      else
        Reject G^*, t^*
      end
      Randomly choose two chains x and y, $x, y \epsilon$ (1, …, n) to swap heats.
      if x and y are not on the same processor **then**
        Send swap information of β_x and β_y across processors.
      end
      Send swap probability $S_i = min\left\{1, \frac{p(G_x,t_x|X)^{\beta_x} p(G_y,t_y|X)^{\beta_y}}{p(G_y,t_y|X)^{\beta_x} p(G_x,t_x|X)^{\beta_y}}\right\}$
      Generate uniform random variable $U \epsilon$ (0, 1)
      if $U < S_i$ **then**
        Swap β_x and β_y
      end
      if *Sampling in this iteration* **then**
        Collect genealogy on head node
      end
    end
  until *N MCMC generations*
end

Algorithm 2: P(MC)3 swap counting algorithm

```
begin
    repeat
        Randomly choose two chains x and y to swap.
        if Swapping in this generation then
            if x < y then
                swap[x][x]++.
            end
            else
                swap[y][x]++.
            end
            Attempt swap between x and y.
            if swap accepted then
                if x < y then
                    swap[y][x]++.
                else
                    swap[x][y]++.
                end
            end
        end
    end
    if thinning then
        Collate/Reduce swap matrix onto head node.
        Call barrier to synchronize.
    end
    until n MCMC generations
end
```

Algorithm 3: P(MC)3 autocorrelations algorithm

```
begin
    repeat
        for All n chains do
            if cold chain then
                Compute autocorrelations of different summary statistics
                if not on head node then
                    Collate/Reduce autocorrelations to head node
                    Call barrier to synchronize
                end
            end
        end
    until Every lag interval
end
```

1. Data simulations

Multilocus genotype (SNP) data was simulated using *ms* (Hudson 2002) under the isolation with migration model using the following command lines for Simulations 1, 2 and 3, respectively. Sequence data was then generated using a Jukes Cantor model of nucleotide substitution, where each base is equally likely to mutate into any other base. Lengths of sequence alignments were varied across loci and replicates.

(1) ms 30 5/50/300 –t 5 –I 2 15 15 –n 1 1.0 –n 2 1 –m 1 2 5 –m 2 1 5 –en 0.002 1 1 –ej 0.002 2 1

(2) ms 100 1 t 5 –I 2 50 50 –n 1 1.0 –n 2 1 –m 1 2 5 –m 2 1 5 –en 0.2 1 1 –ej 0.2 2 1

(3) ms 30 2 –t 5 –I 2 15 15 –n 1 1.0 –n 2 1 –m 1 2 2 –m 2 1 2 –en 0.2 1 1 –ej 0.2 2 1

All simulated IMa2p input files can be downloaded from the git repository under "Simulations".

Runs of IMa2p for Simulation 1 were then performed by setting prior limits on parameters as instructed in the IMa2 (Hey & Nielsen 2007) user manual. In short, Watterson's estimates of the population mutation rate, $\theta = 4Nu$, were computed based on the number of segregating sites at each locus. The geometric mean (x) of each estimate was computed, then the upper bound on θ was set to $5x$, the splitting time upper bound was set to $2x$, and the migration rate upper bound was set to $2/x$. For details on this rule of thumb, see Hey & Nielsen 2007. The random number seed was varied across replicate runs (using the –s flag). The IMa2p command line used in each run is shown below:

Simulation 1

IMa2p –i Sim1_5loci.u –o Sim1_5loci.out –q2 –m1 –t3 –hfg –hn60 –ha0.98 –hb0.75 –r245 –b10000 –l100

IMa2p –i Sim1_50loci.u –o Sim1_50loci.out –q27.49 –m0.36 –t10.99 –hfg –hn60 –ha0.98 –hb0.75 –r245 –b10000 –l100

IMa2p –i Sim1_300loci.u –o Sim1_300loci.out –q48.57 –m0.21 –t19.43 –hfg –hn60 –ha0.98 –hb0.75 –r245 –b10000 –l100

Simulation 2

IMa2p –i Sim2.u –o Sim2.out –c0 –q50 –m1 –t20 –hfg –hn5 –ha0.9 –hb0.8 –b100000 –l0.5 –r245

IMa2p –i Sim2.u –o Sim2.out –c0 –q50 –m10 –t20 –hfg –hn5 –ha0.9 –hb0.8 –b100000 –l0.5 –r245

IMa2p –i Sim2.u –o Sim2.out –c0 –q50 –m100 –t20 –hfg –hn5 –ha0.9 –hb0.8 –b100000 –l0.5 –r245

Simulation 3

IMa2p –i Sim3.u –o Sim3.out –q50 –m10 –t10 –s123 –hfg –hn20 –ha0.97 –hb0.5 –b100000 –l200000

2. Empirical data

Won & Hey (2005) used a data set from 26 primates (Yu *et al*. 2003), comprising genomic DNA sequences across 50 loci, of varying lengths (~480 bp long on average) to infer ancestral demography under the IM model. Parallel runs using IMa2p were performed using the same limits on priors for population sizes, divergence time and migration rates as described in Won & Hey (2005). M mode runs comprised a total of 30 chains distributed across 1–10 processors, and a burn-in period of 100 000 iterations, and 1000 genealogies were saved at every 100th iteration post burn-in (total of 200,000 MCMC iterations).

M mode

IMa2p –i wonhey.u –o wonhey.out –b100000 –L1000 –hn30 –ha0.97 –hb0.9 –hfg –q4 –m5 –t1 –p356 –s1234

L mode

IMa2p –i wonhey.u –o wonhey.out –r0 1000 –v wonhey.out –q4 –m5 –t1 –p245

Table A1 Mean parameter estimates of population sizes (q), migration rates (m) and divergence time (t) between *P. t. troglodytes* and *P. t. verus*, using 47 genomic loci from Won & Hey (2005), estimated by varying the number of processors in IMa2p. Also shown are comparable estimates reported in Table 1 of Won & Hey (2005) using the serial (single processor) version of IMa2

Processors	q_0	q_1	q_2	$m_{0\to1}$	$m_{1\to0}$	t
1	0.76	0.21	0.06	1.736*	0.013ns	0.15
2	0.76	0.19	0.10	1.528*	0.00ns	0.12
3	0.88	0.22	0.15	1.539*	0.381ns	0.18
5	0.73	0.20	0.21	1.378*	0.00ns	0.13
6	0.86	0.23	0.10	1.143*	0.163ns	0.19
10	0.82	0.21	0.05	1.133*	0.114ns	0.23
MLE (Won & Hey 2005)	0.87	0.24	0.16	1.179*	0.002ns	0.22
Lower 90% HPD	0.61	0.16	0.00	0.314	0.002	0.13
Higher 90% HPD	1.27	0.33	0.35	2.541	1.226	0.33

*Indicates statistical significance at a P-value of 0.05 in an LLR test of migration. For details of the LLR test, see Nielsen & Wakeley (2001).