Applied Mathematics & Information Sciences
An International Journal

Parallel Metropolis Coupled Markov Chain Monte Carlo for Isolation with Migration Model

Chunbao Zhou[1], *Xianyu Lang*[1], *Yangang Wang*[1], *Chaodong Zhu*[2], *Zhonghua Lu*[1] *and Xuebin Chi*[1]

[1]Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
[2]Key Laboratory of Systematics and Evolution, Institute of Zoology, Chinese Academy of Sciences, Beijing 100101, China

Received: 13 Aug. 2012, Revised: 1 Nov. 2012, Accepted: 10 Nov. 2012
Published online: 1 Feb. 2013

Abstract: Isolation with Migration model (IM), which jointly estimates divergence times and migration rates between two populations from DNA sequence data, can capture many phenomena when one population splits into two. The parameters inferences for IM are based on Markov Chain Monte Carlo method (MCMC). Standard implementations of MCMC are prone to fall into local optima. Metropolis Coupled MCMC [$(MC)^3$] as a variant of MCMC can more readily explore multiple peaks in posterior distribution of trees. Expensive execution time has limited the application of $(MC)^3$. This paper proposes a Parallel Metropolis Coupled Markov Chain Monte Carlo for IM. The proposed parallel algorithm retains the ability of $(MC)^3$ and maintains a fast execution time. Performance results indicate nearly linear speed improvement. This paper provides researcher with rapider and more high-efficiency methods to study population genetics and molecular ecology problems aided with super computer.

Keywords: Isolation with migration model, metropolis coupled markov chain monte carlo, MPI

1. Introduction

The analysis of population divergence is a major focus in population genetics and molecular ecology. There are two extreme assumptions for most models which are designed to do population divergence. The migrations are at a constant rate for an infinitely long time and the populations are descended from a common ancestral population and diverged without gene flow [1]. Most models are different from the actual world because of these two assumptions.

The aim of Isolation with Migration model (IM) is jointly estimating divergence times and migration rates between two populations from DNA sequence data. There are six parameters in IM: the population sizes for the ancestral population and two descended populations, two different migration rates for two descended populations and the time of population splitting [2]. With these six parameters, IM can capture many phenomena that can occur when one population splits into two. IM has been successfully used for population genetics [2–4].

The parameters inferences for IM are based on Markov Chain Monte Carlo method (MCMC). MCMC is a sampling method with the probability distributions based on a Markov chain that has the desired distribution as its equilibrium distribution. The state of the chain is used as a sample of the desired distribution [5]. MCMC produces the correct distribution as the length of the Markov chain increases [6]. MCMC has been successfully used for gaining posterior probability distribution for phylogenetic tree [7–11].

If the desired distribution for Markov chain has multiple peaks which are separated by low valleys, MCMC is prone to fall into local optima. Many strategies have been proposed to solve this problem, among which Metropolis coupled MCMC [$(MC)^3$] is the successful one. In $(MC)^3$, multiple chains run simultaneously with different stationary distributions. Only one is the target chain called cold chain and the others are hot chains for better mixing by swapping the state information of the chains. The hot chains can more readily cross valleys in the trees. If the cold chain successfully swaps state information with a hot chain, it is likely to cross a valley and find another peak in the tree [12].

The use of multiple chains incurs a significant performance cost for $(MC)^3$. Specifically, only the posterior probability distribution for cold chain is the desired distribution. In $(MC)^3$, each chain requires the

* Corresponding author e-mail: zhoucb@sccas.cn

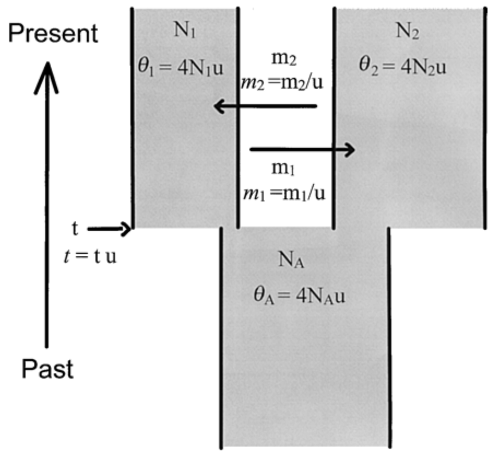

Figure 1 The isolation with migration model [2] .

same amount of computation per iteration and interacts to the others only when swap happens, so it is ideally suited for implementation on parallel machines. CPU and GPU are two parallel ways now and widely used for parallel computing [13,14]. OpenMP for shared memory system and Message Passing Interface (MPI) for distributed storage system are two important ways for parallel computing for CPU [15,16]. MPI is used for parallel processing in this article. MPI is an Application Programming Interface (API) specification for parallel program implementation based on processes. Processes with local memory can not access the memory for the other processes. Processes can only communicate with the other processes by sending and receiving messages. The critical costs for MPI are messages passing and processes synchronization [17].

In this article, we proposed a Parallel Metropolis Coupled Markov Chain Monte Carlo [P(MC)^3] for IM, which improves the performance of IM through the parallel processing. Our proposed method achieved a nearly linear speedup over the sequential version of IM. P(MC)^3 provided new opportunities to IM for population genetics.

2. Model and Algorithm

2.1. Isolation with migration model

The IM model with two sets of parameters is shown in Figure 1. The implement of IM is based on MCMC. The state of the Markov chain is a genealogy (i.e. a gene tree) for each locus in the data set. The length of the Markov chain is measured by steps. Each step is the transfer of the state for Markov chain which picks new values for genealogies and parameters that are included in the state space [2].

The posterior distribution $f(\Theta|X)$ is as follow:

$$f(\Theta|X) = cf(\Theta)\int_{G\in\Gamma} f(X|\Theta,G)f(G)dG \qquad (1)$$

Where Γ refers to the set of all possible gene genealogies, $\Theta = \{\theta, M_1, M_2, \gamma, \alpha, T\}$ is the set of parameters $\theta = 4N_1u$, $\gamma = N_2/N_1$, $\alpha = N_A/N_1$, $M_1 = 2N_1m_1$, $M_2 = 2N_2m_2$, $T = t/2N_1$ and X are genomic sequences [1].

To improve mixing and convergence, metropolis coupling are used for MCMC. Multiple chains can run simultaneously. If the posterior probability density distribution of the phylogenetic parameters for cold chain is $f(\Theta|X)$, then for hot chain is $f(\Theta|X)^\beta$, where $\beta(0<\beta<1)$ is the heat parameter. When the Markov chain with state Θ obtains a proposed new state Θ', the probability of accepting Θ' for cold chain is

$$R = min\left[1, \frac{f(X|\Theta')}{f(X|\Theta)} \times \frac{f(\Theta')}{f(\Theta)} \times \frac{q(\Theta)}{q(\Theta')}\right] \qquad (2)$$

The probability of accepting Θ' for hot chains is

$$R = min\left[1, \left(\frac{f(X|\Theta')}{f(X|\Theta)} \times \frac{f(\Theta')}{f(\Theta)}\right)^\beta \times \frac{q(\Theta)}{q(\Theta')}\right] \qquad (3)$$

The heat parameter increases the acceptance probability of new states for hot chains. In other words, hot chains tend to accept more states than a cold chain and more readily cross valleys in the trees [18]. So successfully swapping between cold chain and hot chains can make cold chain reach a new peak crossing valleys and the cold chain can more easily traverse the space of trees. The obvious disadvantage for IM is the execution time, especially the Metropolis coupled IM. Multiple chains run simultaneously and execution time multiply increase.

2.2. Parallel Metropolis coupled Markov Chain Monte Carlo for IM

In Metropolis coupled IM, multiple chains run simultaneously. The chains require the same amount of computation per iteration and are independent between others only except when swap happens. So it is ideally suited for implementation on parallel machines.

There are N processors used in P(MC)^3, each processor performs all computation associated with its assigned chain(s). Processor N generates random numbers in pairs and sends them to the other processors. The random numbers are the labels of chains for swap. Each processor performs MCMC for chain(s) in them and checks the random numbers to determine whether it involves in swap or not. The processors which are not involved in swap continue to do the next step. If the chains for swap are in the same processor, they swap state information. Otherwise, the processors have to communicate to each other. The details of P(MC)^3 are as follows.

In each step, each processor p ($p \in \{1,2,3,...N\}$, where N is the total number of processors used) does the following:

1.Do MCMC for chain(s) based on parameter Θ. Where $\Theta = \{\theta, M_1, M_2, \gamma, \alpha, T\}$ is the set of parameters of IM.
2.Processor N generates random labels of chains in pairs for swap and sends them to the other processors.
3.Checks the labels and determines if it involves in swap.
4.If processor involves in swap, it change the heat parameter of the chains according to the labels, otherwise it continues to do the next step.
5.Update parameter based on all the loci for chain(s) in processor.
6.The processor involves the chain with heat parameter $\beta = 1$ records information for final results according to the statement information of the chain.

At the end of the run, gather the information from chains with information recorded for final results and summarize the results.

In optimum, we only need change the heat parameter. But some other variables in specific for IM have to be swapped for correct implement. Even so heat parameter is better than the state information of chains for swap.

2.3. Heat parameter

When swap happens between different processors, communication between processors is necessary. High communication costs can severely degrade the performance of P(MC)^3. For IM, state information is tree data structures and associated conditional likelihoods account. Communication costs for such structures are huge. The only difference between the cold chain and the hot chains is the heat parameter. Different stationary distributions are based on the heat parameter. So the heat parameter is suit to communication rather than chain state information and their associated data structures. Once heat parameter is swapped, the chains will accept new states based on their newly acquired heat parameter. Heat parameter swapping is not only an efficient way but also easy to implement [18]. There are also some variables for communication in order to right implement of IM, e.g. static variables. However, the heat parameters swap also calls for communication. In P(MC)^3, each processor holds all the heat parameters and only need decide to use one depending on the swap. The heat parameters swap change to heat parameters and the communication is needless.

2.4. Synchronization

Heat parameter swap calls for synchronization. Success or failure of swap depends on the current state information of chains for swap. If chains are in the same processor, it is easy to decide. But if chains are in different processors, because processor can not access the memory of other processors, the current state information of chains has to be communicated and the cost is huge. In P(MC)^3 each processor for swap does all the computation about itself and only one processor receives the result from the other one, makes a decision and informs the other one.

When the swap is successful, if all the processors want to know it, the synchronization is necessary. If each processor only knows the swap which is relative to it, the synchronization is needless. But there is another problem, if each processor does not know the labels of chains in the other processors and the chains for swap are in different processors, the swap can not happen. So a processor is necessary which informs each other mutual label of process as a middleman. Processors which are not involved in swap can continue to do next step until they are involved in swap. So the processors are selfish and synchronization happens between the processors for swap.

Some Information based on current state information of cold chain is recorded for final results. When heat parameters swap, the cold chain may be different depending on the heat parameter. So the information is dispersedly recorded in different processor and the gather of the information calls for synchronization.

3. Results and Conclusion

We evaluate P(MC)^3 on a server with 40 cores in 4 Intel Xeon E7-4870 processors with frequency 2.4GHz running Red Hat Enterprise Linux Server 6.0. The data set includes a segment (ca. 970 bp) of the mitochondrial gene NADH dehydrogenase 5 subunit (ND5) from 131 individuals of mudskipper (Periophthalmus modestus) [19]. These sequences were deposited in GenBank with the accession numbers HQ453212-HQ453269 and AB257605-AB257625. The mutation model for IM is HKY. The number of swap attempts per step is 1. Duration of run is 1000 and duration of burn is 1000. Population size and splitting parameter can change through implement. The number of genealogy updates per step is 10. The swap takes place in every step. We measure speedup with 2, 4, 8, 16 and 32 processors for 32, 64 and 128 chains. We also ran IM with the same configurations for comparisons.

The speedups are shown in Figure 2. All speedup figures are based on execution times of the sequential version of IM. P(MC)^3 achieves nearly linear speedup for both implementations. The best speedups are 17.42, 20.96 and 22.34 on 32 processors for 32, 64, and 128 chains over the sequential version of IM. The speedup for

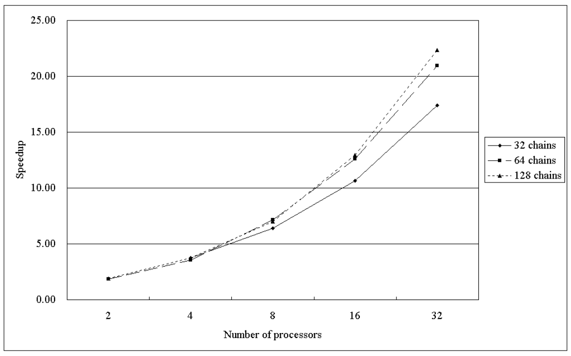

Figure 2 The parallel speedup of P(MC)^3 for different number of chains .

P(MC)^3 depends on the number of chains used. So for larger number of chains, P(MC)^3 can yield larger performance improvement.

We also evaluate P(MC)^3 on supercomputer Lenovo DeepComp 7000 running Red Hat Enterprise Linux Server release 5.1. It has 1140 blade server nodes with Intel Xeon processors and 2 fat nodes with Intel Itanium2 processors. We use blade server nodes for evaluation. Each blade server node has two quad-core Intel Xeon E5450 processors with frequency 3.00GHz and share memory 32GB. The cluster nodes were connected to Infiniband with 1 Gbps Ethernet.

We only evaluate 128 chains for P(MC)^3 on supercomputer Lenovo DeepComp 7000 and the speedup is shown in Figure 3. The speedup is nearly linear, but slows down when we use 128 processors. The speedup for 128 processors is not ideal.

Processor *N* need send random labels of chains for swap to the other processors. When number of processors increases, the communication cost increases meanwhile. More chains are in the same processor, larger probability of chains for swap is in the same processor. When chains for swap are in different processors, the communication cost increases. Heat parameter swap calls for synchronization. The computational imbalance among the chains for swap increases wait time for synchronization. The communication cost and the wait time increase, the speedup decreases definitely.

In this article, we proposed P(MC)^3 of IM for concurrency and to minimize communication costs. The performance of implementations has been analyzed and found to achieve nearly linear speedup. P(MC)^3 opens up the possibility of running a large number of chains for better mixing. The reduction in time achieved by P(MC)^3 provided new opportunities to IM for population genetics.

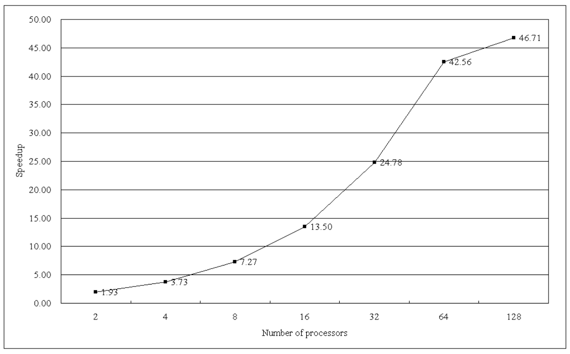

Figure 3 The parallel speedup of P(MC)^3 .

4. Discussion

IM is very useful for population genetics and molecular ecology. IM will provide further insight into evolution with higher accuracy but will also consume more computation time. Expensive execution time has limited the application of IM. Current implementation of P(MC)^3 only exploits chain level parallelism for (MC)^3 of IM and its parallelization is limited by the number of chains used in the analysis. As a next step, we will exploit the intra-chain level parallel model of IM for more speedup such as PBPI [20].

If whole mitochondrial protein sequences are to be analyzed, a single MCMC chain will require memory beyond the capacity of a single CPU [21]. Such problems are common when biologists want to analyze complex models. So the parallel of data for IM is also the focus.

Acknowledgement

The authors acknowledge the financial support by Informatization Special Program of Chinese Academy of Sciences ”Construction and Application for the Supercomputing Environment of Cloud Oriented Service”, the National High Technology Research and Development Program of China (863 Program) (No. 2010AA012301), the National Natural Science Foundation of China (No. 30870268, 31172048) and the Knowledge Innovation Program of Chinese Academy of Sciences (KSCX2-EW-B-02).

References

[1] R. Nielsen and J. Wakeley. Distinguishing migration from isolation: a markov chain monte carlo approach. *Genetics*, 158(2):885–96, 2001.

[2] J. Hey and R. Nielsen. Multilocus methods for estimating population sizes, migration rates and divergence time, with applications to the divergence of drosophila pseudoobscura and d. persimilis. *Genetics*, 167(2):747–60, 2004.
[3] J. Hey. On the number of new world founders: a population genetic portrait of the peopling of the americas. *PLoS Biol*, 3(6):e193, 2005.
[4] Y. J. Won and J. Hey. Divergence population genetics of chimpanzees. *Mol Biol Evol*, 22(2):297–307, 2005.
[5] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21:1087, 1953.
[6] R.M. Neal and University of Toronto. Department of Computer Science. Probabilistic inference using markov chain monte carlo methods. 1993.
[7] J. P. Huelsenbeck and F. Ronquist. Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–5, 2001.
[8] B. Larget and D.L. Simon. Markov chain monte carlo algorithms for the bayesian analysis of phylogenetic trees. *Molecular Biology and Evolution*, 16:750–759, 1999.
[9] S. Li, D.K. Pearl, and H. Doss. Phylogenetic tree construction using markov chain monte carlo. *Journal of the American Statistical Association*, pages 493–508, 2000.
[10] B. Mau, M. A. Newton, and B. Larget. Bayesian phylogenetic inference via markov chain monte carlo methods. *Biometrics*, 55(1):1–12, 1999.
[11] Z. Yang and B. Rannala. Bayesian phylogenetic inference using dna sequences: a markov chain monte carlo method. *Mol Biol Evol*, 14(7):717–24, 1997.
[12] C.J. Geyer. *Markov chain Monte Carlo maximum likelihood*. Defense Technical Information Center, 1992.
[13] B. Kim, K.J. Kim, and J.K. Seong. Gpu accelerated molecular surface computing. *Appl. Math*, 6(1S):185S–194S, 2012.
[14] W. Kim and C. Hong. A distributed hybrid algorithm for composite stock cutting. *Appl. Math*, 6(2S):661S–667S, 2012.
[15] J. Wang, C. Hu, J. Zhang, and J. Li. Message scheduling for array re-decomposition on distributed memory systems. *Future Generation Computer Systems*, 26(2):281–290, 2010.
[16] J. Wang, C.J. Hu, J.L. Zhang, and J.J. Li. Openmp compiler for distributed memory architectures. *SCIENCE CHINA Information Sciences*, 53(5):932–944, 2010.
[17] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface, seconde dition*. the MIT Press, 1999.
[18] G. Altekar, S. Dwarkadas, J. P. Huelsenbeck, and F. Ronquist. Parallel metropolis coupled markov chain monte carlo for bayesian phylogenetic inference. *Bioinformatics*, 20(3):407–15, 2004.
[19] T. Mukai and M. Sugimoto. Genetic population structure of the mudskipper, periophthalmus modestus, in japan inferred from mitochondrial dna sequence variations. *Japanese Journal of Ichthyology*, 53(2):151, 2006.
[20] X. Feng, K.W. Cameron, and D.A. Buell. Pbpi: a high performance implementation of bayesian phylogenetic inference. page 75. ACM, 2006.
[21] P. J. Waddell, Y. Cao, M. Hasegawa, and D. P. Mindell. Assessing the cretaceous superordinal divergence times within birds and placental mammals by using whole mitochondrial protein sequences and an extended statistical framework. *Syst Biol*, 48(1):119–37, 1999.

Chunbao Zhou is an assistant researcher in Supercomputing Center of Chinese Academy of Sciences. His research interests include Bioinformatics, Computational Biology and High Performance Computing.

Xianyu Lang is an associate researcher in Supercomputing Center of Chinese Academy of Sciences. Her research interests include bioinformatics and high performance computing.

Yangang Wang is an associate researcher in Supercomputing Center of Chinese Academy of Sciences. His research interests include computational mathematics and high performance computing.

Chaodong Zhu is an researcher in Institute of Zoology of Chinese Academy of Sciences. His research interests include DNA Taxonomy, Bee Systematics and Parasitoid Systematics.

Zhonghua Lu is an researcher in Supercomputing Center of Chinese Academy of Sciences. Her research interests include computational mathematics and high performance computing.

Xuebin Chi is an researcher in Supercomputing Center of Chinese Academy of Sciences. His research interests include computational mathematics and high performance computing.