Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

OPTIMAL PARALLELIZATION OF A SEQUENTIAL APPROXIMATE BAYESIAN COMPUTATION ALGORITHM

Jean-Michel Marin

I3M (UMR CNRS 5149)
Université Montpellier 2, FRANCE

Pierre Pudlo

I3M (UMR CNRS 5149)
Université Montpellier 2, FRANCE
and CBGP (UMR 1062)
INRA Montpellier, FRANCE

Mohammed Sedki

I3M (UMR CNRS 5149)
Université Montpellier 2, FRANCE

ABSTRACT

Approximate Bayesian Computation (ABC) methods have a lot of success to accomplish Bayesian inference on the parameters of models for which the calculation of the likelihood is intractable. These algorithms consists in comparing the observed dataset to many simulated datasets. These ones can be generated in different ways. Typically, the rejection ABC scheme consists first of simulating parameters using independent calls to the prior distribution and then, given these values, generating the datasets using independent calls to the model. For such a method, the computation time needed to get a suitable approximation of the posterior distribution can be very long. Also, there exist some sequential Monte Carlo methods replacing simulations from the prior by using successive approximations to the posterior distribution. Here, we recall a sequential simulation algorithm and we compare different parallelization strategies. We notably shown that the parallelization of the sequential ABC sampler is useless when using more than four threads per instance of the program and that the standard rejection ABC sampler has to be used when facing a large number of cores. Indeed, in such a case, the cost of the sequential ABC sampler's parallelization prohibits its use.

1 INTRODUCTION

Approximate Bayesian Computation (ABC) methods, also known as likelihood-free techniques, have appeared in the past ten years as the most satisfactory approach to intractable likelihood problems, first in population genetics (Beaumont, Zhang, and Balding 2002) then in a broader spectrum of applications. Many papers are presenting the current state of art in ABC, one can see for instance Csillèry et al. (2010), Beaumont (2010) and Marin et al. (2012). All are presenting stochastic algorithms targeted for some approximations to the bayesian posterior distribution, but none of them is focused on the parallelization issue.

Assume that we face a bayesian inference problem on some parameter $\theta \in \Theta$ with prior $\pi(\theta)$ and model distribution (or likelihood) $\ell(\mathbf{x}|\theta)$. ABC methods consist of simulating a large number N of particles $(\theta_i, \mathbf{x}_i)$, $i = 1, \ldots N$, from the joint distribution $\pi(\theta)\ell(\mathbf{x}|\theta)$ and approximating the posterior distribution using non-parametric conditional density estimate of θ knowing that $\mathbf{x}$ is the observed dataset $\mathbf{x}_{\text{obs}}$. The basic Monte Carlo sampler of Algorithm 1 is easy to parallelize since all particles are independent.

Algorithm 1 Likelihood-free sampler

1: **for** $i = 1$ to N **do**
2: Generate θ_i from the prior distribution $\pi(\cdot)$
3: Generate $\mathbf{x}_i$ from the likelihood $\ell(\cdot|\theta_i)$
4: **end for**
5: From all particles $(\theta_i, \mathbf{x}_i)$, estimate the conditional density of θ given $\mathbf{x} = \mathbf{x}_{\text{obs}}$

This first sampler produces many useless particles $(\theta_i, \mathbf{x}_i)$ for the conditional density estimate: any particle whose simulated dataset $\mathbf{x}_i$ is far from the observed dataset $\mathbf{x}_{\text{obs}}$ does not contribute to the posterior density estimate. Sampling the parameter space with a (possibly non informative) prior distribution $\pi(\theta)$ is inefficient because this might generate datasets far from the observed dataset, hence irrelevant for the posterior density estimate. So, the standard ABC algorithm (Algorithm 2) performs rejection (which includes sorting the particles with respect to their distances to the observation) before doing any conditional density estimate.

Algorithm 2 Likelihood-free rejection sampler

1: Fix a given quantile $\alpha \in]0,1[$
2: **for** $i = 1$ to N **do**
3: Generate θ_i from the prior distribution $\pi(\cdot)$
4: Generate $\mathbf{x}_i$ from the likelihood $\ell(\cdot|\theta_i)$
5: **end for**
6: Sort all particles $(\theta_i, \mathbf{x}_i)$ with respect to their distances to the observation $\mathbf{x}_{\text{obs}}$ in increasing order, eventually through the use of a summary statistics $S(\cdot)$
7: Use only the $\lfloor \alpha N \rfloor$ first particles for the conditional density estimate ($\lfloor a \rfloor$ defines the integer part of a)

Sequential techniques can enhance the efficiency of this raw sampler by learning about the target distribution (the posterior distribution), as in Sisson, Fan, and Tanaka (2007), Sisson, Fan, and Tanaka (2009), Beaumont et al. (2009), Toni et al. (2009), Drovandi and Pettitt (2011), Del Moral, Doucet, and Jasra (2012) and Sedki et al. (2012). Those methods rely on importance sampling arguments (changing the sampling distribution together with weighting the particles to take that change into account). But they are much more tricky to parallelize.

Section 2 describes quickly the likelihood-free sequential Monte Carlo sampler (SMC) of Sedki et al. (2012), that is a particular likelihood-free SMC (Del Moral, Doucet, and Jasra 2012) designs to minimize the number of simulated dataset for a self-calibrated level of approximation. Once this scheme will be recall, we discuss its parallel implementation on a shared memory architecture. In Section 3, we compare the different possibilities on a real challenging population genetics example.

2 PARALLEL IMPLEMENTATION OF A LIKELIHOOD FREE SMC SAMPLER

To outperform the likelihood-free rejection sampler, Algorithm 2, it has been proposed to modify the simulation process of the parameters θ, avoiding the use of the prior distribution. Clearly, a sort of optimal proposal distribution would be the posterior distribution, because, in such a case, it would be easier to get a simulated dataset $\mathbf{x}$ near the observation $\mathbf{x}_{obs}$. However, the posterior is precisely our unknown target distribution. So, the idea is to introduce a temporal dimension in order to learn gradually the posterior. That is the idea behind likelihood-free SMC samplers of Del Moral, Doucet, and Jasra (2012). These algorithms corresponds to the implementation of the famous SMC sampler of Del Moral, Doucet, and Jasra (2006) in case where the calculation of the target density is intractable. In Sedki et al. (2012), a self-calibrated version that optimizes the computing time is introduced. For simplicity in the presentation here, we consider a uniform prior distribution over a compact set $\Theta \subset \mathbb{R}$. Algorithm 3 gives a precise

description of iteration t of the Sedki et al. (2012)'s scheme. After an initialization step based on the likelihood-free rejection sampler, at iteration t, we have a particle system whose maximal distance to the observation is ε_{t-1} and we want to decrease this threshold to some ε_t we have to calibrate. Suppose that ε_t is known, to get a particle of the new sample, it is proposed to pick one particle among the ones that are below the new threshold (line 21) and try to move it according to a Markov chain that leaves the current distribution invariant (lines 22 and 23) using a likelihood-fee Monte Carlo Markov Chain (MCMC) move (Marjoram et al. 2003). We shall note that applying one step of a MCMC sampler might leaves you with an unmoved particle. The efficiency of this sequential sampler rely on the acceptance rate of the MCMC moves. The method of Sedki et al. (2012) to calibrated the new threshold ε_t is a trade-off between the decrease of the threshold and the good mixing properties of the MCMC sampler (lines between 1 and 15). Algorithm 3 is run until the acceptance rate of the MCMC sampler at iteration t, ρ_t is less than a given threshold, typically 10%. Note that the variance of the Gaussian random walk MCMC proposal τ_t is calibrated as twice the empirical variance of the $\theta_1^t, \ldots, \theta_N^t$.

Algorithm 3 Iteration t of the self-calibrated likelihood-free SMC sampler of Sedki et al. (2012)

1: ♯ `Calibration of` α'
2: $\alpha' \leftarrow 0.0$
3: **repeat**
4: $\quad \alpha' \leftarrow \alpha' + 0.01$
5: $\quad$ Sort the particles $(\theta_i^{t-1}, \mathbf{x}_i^{t-1})$ with respect to their distances to the observation $d_i = d(S(\mathbf{x}_i^t), S(\mathbf{x}_{\text{obs}}))$
6: $\quad \varepsilon'(\alpha') \leftarrow d_{\lfloor \alpha' N \rfloor}$
7: $\quad$ **for** $i = 1$ **to** $\lfloor \alpha' N \rfloor$ **do**
8: $\quad\quad$ **if** particle $(\tilde{\theta}_i, \tilde{\mathbf{x}}_i)$ does not exist **then**
9: $\quad\quad\quad$ draw $\tilde{\theta}_i$ from normal distribution $N(\theta_i^{t-1}, \tau_{t-1})$ and $\tilde{\mathbf{x}}_i$ from $\ell(\mathbf{x}|\tilde{\theta}_i)$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad$ Set $N'(\alpha')$ as the number of particles $(\tilde{\theta}_i, \tilde{\mathbf{x}}_i)$ among the $\lfloor \alpha' N \rfloor$ first ones satisfy $\tilde{\theta}_i \in \Theta$ and $d(S(\tilde{\mathbf{x}}_i), S(\mathbf{x}_{obs})) \leq \varepsilon'(\alpha')$
13: $\quad \rho'(\alpha') \leftarrow N'(\alpha') / \lfloor \alpha' N \rfloor$
14: **until** $\alpha' + \rho'(\alpha') \geq 0.9$
15: $\alpha_t \leftarrow \alpha'$, $\varepsilon_t \leftarrow \varepsilon'(\alpha')$ and $\rho_t \leftarrow \rho'(\alpha')$
16: ♯ `Computation of the new particle system`
17: **for** $j = 0$ **to** $\lfloor \alpha_t N \rfloor$ **do**
18: $\quad$ **if** $\tilde{\theta}_i \in \Theta$ and $d(S(\tilde{\mathbf{x}}_i), S(\mathbf{x}_{obs})) \leq \varepsilon_t$ **then** $(\theta_j^t, \mathbf{x}_j^t) \leftarrow (\tilde{\theta}_j, \tilde{\mathbf{x}}_j)$ **else** $(\theta_j^t, \mathbf{x}_j^t) \leftarrow (\theta_j^{t-1}, \mathbf{x}_j^{t-1})$ **end if**
19: **end for**
20: **for** $j = \lfloor \alpha_t N \rfloor + 1$ **to** N **do**
21: $\quad$ Draw some integer number I uniformly between 1 and $\lfloor \alpha_t N \rfloor$
22: $\quad$ Draw $\tilde{\theta}$ from normal distribution $N(\theta_I^{t-1}, \tau_{t-1})$ and $\tilde{\mathbf{x}}$ from $\ell(\mathbf{x}|\tilde{\theta})$
23: $\quad$ **if** $\tilde{\theta} \in \Theta$ and $d(S(\tilde{\mathbf{x}}), S(\mathbf{x}_{obs})) \leq \varepsilon_t$ **then** $(\theta_j^t, \mathbf{x}_j^t) \leftarrow (\tilde{\theta}, \tilde{\mathbf{x}})$ **else** $(\theta_j^t, \mathbf{x}_j^t) \leftarrow (\theta_I^{t-1}, \mathbf{x}_I^{t-1})$ **end if**
24: **end for**

In Sedki et al. (2012), there is no consideration about the parallelization issues. The comparison with others likelihood-free SMC schemes is done via the number of simulated dataset needed to obtain the same level of approximation.

Monte Carlo computation is often considered as an easy setting for parallel computing: averaging large numbers of simulations drawn from a given distribution breaks up into independent pieces. The only *parallel overhead* comes from averaging between the different chunks. But the first and important issue for parallel Monte Carlo is to design independent Random Number Generators (RNG) that might run in parallel while minimizing the communications between processors. RNG is a research topic in itself, and some parallel RNGs have been proposed in the literature. We found that the Dynamic Creator of Matsumoto and Nishimura (2000) was simple to use and gave good results. It produces a set of independent Mersenne-Twister generators with different internal parameters and different seeds. There is no limitation on the number of RNGs it produces, once initialized, the different RNGs do not require any communication between them and each of them runs as quickly as a single Mersenne-Twister generator. With those RNGs, standard independent and identically distributed Monte Carlo algorithms run in parallel with very little difficulty. Hence Algorithm 2 is *embarrassingly parallel*. The different outputs can be easily merged for conditional density estimation which we do not seek to run in parallel since this is not the time consuming part of the ABC methodology.

In contrast, sequential algorithms such as SMC samplers (Del Moral, Doucet, and Jasra 2006) are *not embarrassingly parallel*. However, these samplers, whose iterations are composed of independent drawings from a distribution based on results of previous iterations, are easiest to parallelize than MCMC algorithms. We focused here on the self-calibrated likelihood-free SMC sampler of Sedki et al. (2012) presented above. This scheme does not breaks up into independent tasks without introducing a lot of communications between the processors charged with them. Still, if the processors share the same memory, the many simulations according to the probabilistic model in lines 9 and 22 can be performed in parallel. For instance, we used the OpenMP API for shared-memory parallel implementation of Algorithm 3. But it might be clear that, if internal parallelization of SMC samplers is difficult, we shall bypass the obstacle by running several independent SMC samplers in parallel. Even on modern computing clusters composed of multiple-core machines (or nodes), an hybrid solution, running several samplers which are internally parallel. We compare these different solutions in the next Section.

3 COMPARISON OF DIFFERENT PARALLELIZATION STRATEGIES

We here consider the same population genetics example as in Sedki et al. (2012). That is a honeybee population dataset including eight microsatellite loci. The evolutionary scenario concerns four populations observed at the present time linked through three divergence and two admixture events involving two unobserved populations. The same dataset has also been analyzed in Cornuet et al. (2006) with a slightly different population scenario. The dimension of Θ is 15 and the one of the summary statistics vector $S(\cdot)$ is 30. The prior distributions on the divergence and admixture times, the parameters involved in the mutation rates and the admixture rates are uniform and described in Sedki et al. (2012).

We have used the likelihood-free SMC sampler to get an approximation of the target posterior distribution. The number of simulated datasets during the initialization step was fixed to 1,200,000 and we use $N = 200,000$ particles per iteration. After, 10 iterations the SMC sampler stopped. The total number of simulated datasets is 3,200,000 and the resulting Effective Sample Size is around 72,000. As already argued in Sedki et al. (2012), to get the same level of approximation, the likelihood-free rejection sampler needs more than twice the number of simulated datasets, typically around 7,200,000.

We have compared the following four different strategies:

- independent runs of the likelihood-free rejection sampler (for 40 cores and 7,200,000 simulated datasets, we generate 180,000 datasets on each core and the 40 independent results are merged at the end);
- independent runs of the likelihood-free SMC sampler (for 40 cores and 200,000 particles per iteration, we run 40 independent SMC samplers with 5,000 particles per iteration and the 40 results are merged at the end);

- independent runs of the likelihood-free SMC sampler parallelized on 4 cores (for 40 cores and 200,000 particles per iteration, we run 10 independent SMC samplers with 20,000 particles per iteration, the SMC calculations are parallelized on 4 cores and the 10 independent results are merged at the end);
- independent runs of the likelihood-free SMC sampler parallelized on 10 cores (for 40 cores and 200,000 particles per iteration, we run 4 independent SMC samplers with 50,000 particles per iteration, the SMC calculations are parallelized on 10 cores and the 4 independent results are merged at the end).

Table 1 presents the results. It contains the computing time in seconds of the different strategies for the same level of approximation, that is the same threshold ε and the same Effective Sample Size. Note that these experiments have been repeated enough to ensure that the outcomes do not vary between versions of the ABC samplers. When the number of cores are important, the results obtained using independent runs of the likelihood-free SMC sampler are much more variable than the other ones, notably . On our population genetics example, a total number of 100 cores is the maximum for such a strategy. Indeed, in that case, there is only 2,000 particles per iteration and, using less particles per iteration induces lot of variability in the learning process of likelihood-free SMC sampler. It turns out that:

- the parallelization of the likelihood-free SMC sampler with OpenMP was useless when using more than 4 cores per instance;
- running independent instances of the likelihood-free SMC sampler is time saving. Here the total number of particles is allocated uniformly over the instances, but each instance requires a minimal number of particles, say a few thousands in our case, for calibration purpose.
- the standard likelihood-free rejection sampler is certainly the most easy algorithm to parallelize, and we advocate its use when facing more than a certain number of cores, say 80 in our case.

Table 1: Computing time in seconds of different algorithms and parallelization strategies.

Total number of cores	**Likelihood-free rejection sampler**	**Likelihood-free SMC sampler**	**SMC sampler parallelized on 4 cores**	**SMC sampler parallelized on 10 cores**
20	2010	866	1240	2427
40	1046	453	662	1213
80	522	237	350	685
100	419	197	295	560

4 DISCUSSION

On a challenging population genetics example, using OpenMP, we have shown that the parallelization of the likelihood-free SMC sampler was useless when using more than 4 cores per instance and we demonstrated that the use of the standard likelihood-free rejection sampler should be advocated when facing more than 80 cores. These precise conclusions in term of number of cores are specific to the considered example and the use of OpenMP. In situations where the prior already gives a good information on the parameters (like in our population genetics example), the sophisticated SMC sampler do not save a large amount of time. In this case, we strongly believe that the overhead to coordinate parallel simulations, i.e., task start up and termination time as well as data communications, might dominates the computing time even in shared memory architectures. Even if general guidelines are difficult to give, a good strategy can be to use independent instances of the likelihood-free SMC sampler and analyze carefully the variability of the obtained results to deduce the best parallelization strategy.

REFERENCES

Beaumont, M., W. Zhang, and D. Balding. 2002. "Approximate Bayesian Computation in population genetics". *Genetics* 162 (4): 2025–2035.

Beaumont, M. A. 2010. "Approximate Bayesian Computation in Evolution and Ecology". *Annu. Rev. Ecol. Evol. Syst.* 41:379–406.

Beaumont, M. A., J.-M. Cornuet, J.-M. Marin, and C. P. Robert. 2009. "Adaptive approximate Bayesian computation". *Biometrika* 96 (4): 983–990.

Cornuet, J.-M., L. Excoffier, P. Franck, and A. Estoup. 2006. "Inférence bayésienne dans des scénarios évolutifs complexes avec populations mélangées: application à l'abeille domestique". In *Les actes du BRG, 6*, 163–180. Bureau des ressources gntiques, 2006.

Csillèry, K., M. Blum, O. Gaggiotti, and O. François. 2010. "Approximate Bayesian Computation (ABC) in practice". *Trends in Ecology and Evolution* 25 (7): 410–418.

Del Moral, P., A. Doucet, and A. Jasra. 2006. "Sequential Monte Carlo samplers". *J. R. Stat. Soc. Ser. B Stat. Methodol.* 68 (3): 411–436.

Del Moral, P., A. Doucet, and A. Jasra. 2012. "An adaptive sequential Monte Carlo method for approximate Bayesian computation". *Stat. Comput.*. (to appear).

Drovandi, C. C., and A. N. Pettitt. 2011. "Estimation of parameters for macroparasite population evolution using approximate Bayesian computation". *Biometrics* 67 (1): 225–233.

Marin, J.-M., P. Pudlo, , C. P. Robert, and R. Ryder. 2012. "Approximate Bayesian computational methods". *Stat. Comput.*. (to appear).

Marjoram, P., J. Molitor, V. Plagnol, and S. Tavaré. 2003. "Markov chain Monte Carlo without likelihoods". *Proc. Natl. Acad. Sci. USA* 100 (26): 15324–15328.

Matsumoto, M., and T. Nishimura. 2000. "Dynamic creation of pseudorandom number generators". In *Monte Carlo and quasi-Monte Carlo methods 1998 (Claremont, CA)*, 56–69. Berlin: Springer.

Sedki, M., J.-M. Cornuet, J.-M. Marin, P. Pudlo, and C. Robert. 2012. "Efficient learning in ABC algorithms". Technical report, ArXiv.org.

Sisson, S., Y. Fan, and M. Tanaka. 2009. "Sequential Monte Carlo without likelihoods: Errata". *Proc. Natl. Acad. Sci. USA* 106 (39): 16889.

Sisson, S. A., Y. Fan, and M. M. Tanaka. 2007. "Sequential Monte Carlo without likelihoods". *Proc. Natl. Acad. Sci. USA* 104 (6): 1760–1765 (electronic).

Toni, T., D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf. 2009. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". *J. R. Soc. Interface* 6 (31): 187–202.

AUTHOR BIOGRAPHIES

JEAN-MICHEL MARIN is a Professor of Statistics and Numerical Probabilities in the Institut de Mathématiques et Modélisation de Montpellier (I3M, the Mathematics department) of the University of Montpellier 2. He is co-head of the Mathematics department (I3M) since 2010. His research interests/expertise include Monte Carlo methods, MCMC algorithms, adaptive importance sampling schemes; Bayesian statistics and model choice; with applications in population genetics and molecular biology. His email address is jean-michel.marin@math.univ-montp2.fr and his web page is http://www.math.univ-montp2.fr/~marin.

PIERRE PUDLO is an assistant Professor in the Institut de Mathématiques et Modélisation de Montpellier (I3M) of the University of Montpellier 2. He is also hosted in the laboratory "Centre de Biologie pour la Gestion des Populations" of INRA who carries out research in the fields of systematics, genetics and ecology relevant to the management of populations and communities of organisms for the purposes of agriculture, public health and biodiversity. His research interests/expertise include applied probabilities with a specific focus on population genetic models; Bayesian analysis; Monte Carlo methods; and clustering (data analysis). His

email address is pierre.pudlo@univ-montp2.fr and his web page is http://www.math.univ-montp2.fr/~pudlo.

MOHAMMED SEDKI is now a post doctoral scholar in the "Select" team of the mathematics Department of University Paris-Sud and Inria. He received a doctorate in Statistics from the University of Montpellier 2 in october 2012. His research interests are in adaptive importance sampling schemes; likelihood free Bayesian analysis (ABC), with applications in population genetics. His email address is msedki@math.univ-montp2.fr.