Evolutionary Markov chain Monte Carlo

Mădălina M. Drugan

Dirk Thierens

institute of information and computing sciences, utrecht university

technical report UU-CS-2003-047

www.cs.uu.nl

Evolutionary Markov chain Monte Carlo

Mădălina M. Drugan and Dirk Thierens

Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{madalina,dirk}@cs.uu.nl

Abstract. *Markov chain Monte Carlo* (MCMC) is a popular class of algorithms to sample from a complex distribution. A key issue in the design of MCMC algorithms is to improve the proposal mechanism and the mixing behaviour. This has led some authors to propose the use of a population of MCMC chains, while others go even further by integrating techniques from evolutionary computation (EC) into the MCMC framework. This merging of MCMC and EC leads to a class of algorithms, we call *Evolutionary Markov Chain Monte Carlo* (EMCMC). In this paper we first survey existing EMCMC algorithms and categorise them in two classes: *family-competitive EMCMC* and *population-driven EMCMC*. Next, we introduce the *Elitist Coupled Acceptance* rule and the *Fitness Ordered Tempering* algorithm.

1 Introduction

Markov Chain Monte Carlo (MCMC) algorithms provide a framework for sampling from complicated target distributions that cannot be sampled with simpler, distribution specific, methods. MCMC algorithms are applied in many fields, and their use in Machine Learning has recently been advocated in [1]. Usually, MCMC uses a single chain which runs for a long time. However, to improve the convergence rate, there are some MCMC variants that work with a population of MCMCs. The use of a population makes these algorithms somewhat similar to Evolutionary Algorithms (EA). Indeed some authors have proposed algorithms that integrate techniques from the EC field into the MCMC framework. Here we survey these EMCMC algorithms and classify them into two basic categories: *family-competitive EMCMC* algorithms that operate through an acceptance rule at the family level, and *population-driven EMCMC* algorithms that operate through a population-driven, adaptive proposal distribution. One property of EMCMC algorithms is that they are not necessarily a set of parallel MCMC chains, but that they are a single MCMC at the population level. Besides surveying existing EMCMC algorithms we also propose two alternative techniques: the *Elitist Coupled Acceptance* (ECA) rule and the *Fitness Ordered Tempering* (FOT) algorithm.

The paper is organised as follows. Section 2 discusses basic concepts and algorithms from MCMC. Section 3 describes parallel MCMC algorithms, while Section 4 surveys EMCMC algorithms. We introduce the ECA and FOT techniques in Section 5, and report some experimental results in Section 6.

2 The Markov Chain Monte Carlo framework

MCMC is a general framework to generate samples X_t from a probability distribution $P(\cdot)$ while exploring its search space $\Omega(X)$ using a *Markov chain*. MCMC does not sample directly from $P(\cdot)$ but only requires that the density $P(X)$ can be evaluated within a multiplicative constant $P(X) = \frac{P'(X)}{Z}$, where Z is a normalisation constant and $P'(\cdot)$ is the unnormalised target distribution. A Markov chain is a discrete-time stochastic process $\{X_0, X_1, \ldots\}$ with the property that the state X_t given all previous values $\{X_0, X_1, \ldots, X_{t-1}\}$ only depends on X_{t-1}: $P(X_t \mid X_0, X_1, \ldots, X_{t-1}) = P(X_t \mid X_{t-1})$. We call $P(\cdot \mid \cdot)$ the transition matrix of the Markov chain. $P(\cdot \mid \cdot)$ is a *stationary* - this is, independent of time t - *transition matrix* with the following properties: (i) all the entries are non-negative, and (ii) the sum of the entries in a row is 1. We assume that $P(\cdot) > 0$. MCMC converges, in infinite time, to the probability distribution $P(\cdot)$, thus it samples with higher probability from more important states of $P(\cdot)$. An finite state MCMC which has an *irreducible* and *aperiodic* stationary transition matrix converges to a unique *stationary distribution* [1]. A MCMC chain is irreducible if, and only if, every state of the MCMC chain can be reached from every other state in several steps. A MCMC is aperiodic if, and only if, there exists no cycles to be trapped into. A sufficient, but not necessary, condition to ensure that $P(\cdot)$ is the stationary distribution is that MCMC satisfies the *detailed balance condition* [1]. A MCMC satisfies the detailed balance condition if, and only if, the probability to move from X_t to X_{NEW} multiplied by the probability to be in X_t is equal to the probability to move from X_{NEW} to X_t multiplied by the probability to be in X_{NEW}: $P(X_{NEW} \mid X_t) \cdot P(X_t) = P(X_t \mid X_{NEW}) \cdot P(X_{NEW})$.

Metropolis-Hastings algorithms. Many MCMC algorithms are Metropolis-Hastings (MH) algorithms [10, 20]. Since we cannot sample directly from $P(\cdot)$, MH algorithms consider a simpler distribution $S(\cdot \mid \cdot)$, called the *proposal distribution* for sampling the next state of a MCMC chain. $S(X_{NEW} \mid X_t)$ generates the candidate state X_{NEW} from the current state X_t, and the new state X_{NEW} is accepted with probability:

$$A(X_{NEW} \mid X_t) = \min\left(1, \frac{P'(X_{NEW}) \cdot S(X_t \mid X_{NEW})}{P'(X_t) \cdot S(X_{NEW} \mid X_t)}\right)$$

If the candidate state is accepted the next state becomes $X_{t+1} = X_{NEW}$. Otherwise, $X_{t+1} = X_t$. The transition probability for arriving in X_{NEW} when the current state is X_t is $T(X_{NEW} \mid X_t) = S(X_{NEW} \mid X_t) \cdot A(X_{NEW} \mid X_t)$, if $X_{NEW} \neq X_t$, and $T(X_t \mid X_t) = 1 - \sum_{Y, Y \neq X_t} S(Y \mid X_t) \cdot A(Y \mid X_t)$, otherwise.

The pseudo-code for the MH algorithm is:

```
$Metropolis-Hastings()$
1  $Initialise\ X_0; t \leftarrow 0$
2  while $true$
3      do $Sample\ X_{NEW}\ from\ S(\cdot \mid X_t)$
4          if $Uniform\_sampling(0,1) \leq A(X_{NEW} \mid X_t)$
5            then $X_{t+1} \leftarrow X_{NEW}$
6            else $X_{t+1} \leftarrow X_t$
7          $t \leftarrow t+1$
```

A MH algorithm is aperiodic, since the chain can remain in the same state with a probability greater than 0, and by construction it satisfies the detailed balance condition. If, in addition, the chain is irreducible, then it converges to the stationary distribution $P(\cdot)$. The rate of convergence depends on the relationship between the proposal distribution and the target distribution: the closer the proposal distribution is to the stationary distribution, the faster the chain converges. Two popular Metropolis-Hastings algorithms are the *Metropolis algorithm* and the *independence sampler*. For the Metropolis algorithm, the proposal distribution is symmetrical $S(X_{NEW} \mid X_t) = S(X_t \mid X_{NEW})$ and the acceptance rule becomes $A(X_{NEW} \mid X_t) = \min(1, \frac{P'(X_{NEW})}{P'(X_t)})$. Because it accepts the candidate states often - and thus the state space is well sampled - the Metropolis rule generally performs well. The proposal distribution of the independence sampler does not depend on the current state $S(X_{NEW} \mid X_t) = S(X_{NEW})$. The independence sample's acceptance probability can be written as $A(X_{NEW} \mid X_t) = \min(1, \frac{w(X_{NEW})}{w(X_t)})$, where $w(\cdot) = \frac{P'(\cdot)}{S(\cdot)}$. Candidate states with low $w(X_{NEW})$ are rarely accepted, while states with high $w(X_{NEW})$ are very often accepted, and the process could get stuck for a long time in states with very high $w(X)$. Obviously, the choice of $w(\cdot)$ greatly influences the convergence rate.

Simulated annealing (SA). SA is a minor modification of a single chain MH algorithm used for optimisation. Instead of sampling from the entire distribution $P(\cdot)$, SA samples at step t from $P'_t(\cdot) = P'(\cdot)^{\frac{1}{Temp[t]}}$, where $Temp[t]$ decreases according to a cooling scheduler to 0. With $Temp[\cdot]$ close to ∞, the chain accepts almost any candidate state according to MH acceptance rule A, whereas, when $Temp[\cdot]$ is close to 0, the chain rejects almost all states that have lower fitness than the current one. Note that, for constant temperature $Temp[t]$, SA is a MCMC which converges to the distribution $P_t(\cdot)$. However, every time the SA chain is cooled, the transition matrix is changed and the detailed balance is not satisfied. Yet, in infinite time, SA converges to the optimum and, more general, if $Temp[i]$ decreases to 1 SA converges to the stationary distribution $P(\cdot)$. SA is a *non-homogeneous MCMC* which converges to a given stationary distribution. The time to convergence depends on the cooling schedule. In practice, a fast cooling schedule is preferred to a slower one, increasing the risk of poor performance.

3 Parallel MCMC algorithms

MCMC algorithms are usually applied in a sequential way. A single chain is run for a long time until it converges to the stationary distribution $P(\cdot)$. The states visited during the initial phase of the run are considered to be unreliable and further ignored. For reasons of computational efficiency this burn-in phase should be as short as possible. MCMCs with a short burn-in phase are said to mix well (note that this is unrelated to the mixing of building blocks in the EC literature). There exist various variations on the standard MCMC algorithm to speed up this mixing process. In this paper we are particularly interested in techniques that use multiple chains in parallel as opposed to a single chain.

Multiple independent chains (MIC). The most straightforward technique to make use of multiple chains is simply to run N independent MCMCs in parallel. The chains are started at different initial states and their output is observed at the same time. It is hoped that this way a more reliable sampling of the target distribution $P(\cdot)$ is obtained. It is important to note that no information exchange between the chains is taking place. Recommendations in the literature are conflicting regarding the efficiency of parallel independent chains. Yet there are at least theoretical advantages of multiple independent chains MCMC for establishing its convergence to $P(\cdot)$ [7].

Parallel tempering (PT). Parallel tempering [6] is a parallel MCMC with N chains each having a different stationary distribution $P'_i(\cdot) = P'(\cdot)^{\frac{1}{Temp[i]}}$, $i = 1, \ldots, N$. The temperatures have an increasing magnitude $Temp[1] < \ldots < Temp[N]$ with $Temp[1] = 1$. The stationary distribution of the lowest temperature chain is therefore equal to the target distribution, or $P'_1(\cdot) = P'(\cdot)$. The temperatures $Temp[i], (2 \leq i \leq N)$ are given a constant value, typically according to a geometrically increasing series. Note that this is similar to the temperature values proposed by the cooling scheduler of simulated annealing, though for SA the different temperatures are generated sequentially in time, while for PT the different temperatures are generated in space - this is, the population - and remain unchanged during the run.

The candidate states are generated using mutation and accepted with the standard Metropolis-Hasting acceptance rule. Chains in PT exchange information by swapping states. Two chains i and j interact by trying to exchange their current states $X_t[i]$ and $X_t[j]$ using the *swapping acceptance rule*:

$$A_S(X_t[i], X_t[j]) = \min\left(1, \frac{P'_i(X_t[j]) \cdot P'_j(X_t[i])}{P'_i(X_t[i]) \cdot P'_j(X_t[j])} \cdot \frac{S(X'_t \mid X''_t)}{S(X''_t \mid X'_t)}\right)$$

where $X'_t = (\ldots, X_t[i], \ldots, X_t[j], \ldots)$ and $X''_t = (\ldots, X_t[j], \ldots, X_t[i], \ldots)$. Note that A_S is a MH acceptance rule, satisfying the detailed balance condition. A_S accepts with probability 1 an exchange of states if the more important state is inserted into the chain with the lower temperature. To increase the acceptance rate the two chains usually have adjacent temperatures ($|i - j| = 1$). Heuristically, PT improves mixing: better states of a warmer chain can be inserted in a colder chain that is mixing slower.

Parallel sintering (PS). Parallel sintering [12] can be viewed as a generalisation of parallel tempering where the proposal distributions of each chain in the population is a member of some family of distributions $\{P_i(\cdot) \mid i = 1, \ldots, N\}$ defined over spaces of different dimensions (resolutions), that are related to the target distribution by the highest dimensional distribution (or largest resolution) $P_1(\cdot) = P(\cdot)$. As in PT parallel sintering exchanges information between adjacent chains by exchanging states through the swapping acceptance rule. Here too the idea is to increase the mixing of the slowly mixing highest dimensional chain towards the target distribution by inserting states of the lower dimensional - and faster mixing - chains in the population. Parallel sintering is a multiple chain MCMC version of the single chain MCMC called simulated sintering [16].

4 Evolutionary MCMC algorithms

In the previous section, we have outlined parallel MCMCs. In the following we survey existing EMCMC algorithms, and distinguish between two categories: *family-competitive EMCMC* and *population-driven EMCMC*.

Table 1. Comparison of the presented EMCMC algorithms

algorithm	perturbation op.	communication	type EMCMC	optim. / sampl.(distrib.)
EMC	mut, recomb	A_C, A_S	fam-comp	$\prod_{i=1}^{N} P'_i(\cdot)$
MRGA	mut, recomb	A_C, A_S	fam-comp	$\prod_{i=1}^{N} P'_i(\cdot)$
popSA		Boltzmann trials	fam-comp	maxim
PRSA	mut, recomb	A_C	fam-comp	maxim
mparSA	mut, neigh. recomb	neighbours	fam comp	maxim
popMCMC	mut	$S(\cdot \mid \cdot)$	pop-driven	$\prod_{i=1}^{N} P'(\cdot)$
eMCMC	mut, recomb	recomb, $S(\cdot \mid \cdot)$	pop-driven	maxim
ECA	mut, recomb	ECA	fam-comp	$\prod_{i=1}^{N} R'(\cdot)$

Family-competitive EMCMC algorithms let two chains from the population exchange information to sample two new states. This interaction can be implemented by the proposal mechanism and/or by the acceptance rule. Recombining the states of the two chains is an example of the former approach, while in the latter two states are selected from the two proposed states and their 'parent' states. We call any MH acceptance rule which selects two states from the family of four states a *coupled acceptance rule*. Note that in this view parallel tempering (and parallel sintering) belong to the class of family-competitive EMCMC algorithms. Family-competitive EMCMC algorithms correspond to family-competitive EAs where two individuals create offspring by mutation and recombination, and selection takes places at this family level - examples are the elitist recombination GA [25], the deterministic crowding GA [18], the genetic invariance GA [5].

Population-driven EMCMC algorithms adapt their proposal distribution according to the entire, current population of states. The adaptation is such that the algorithm maintains a single MCMC at the population level - this is, a MCMC whose states are populations. Population-driven EMCMC algorithms correspond to the probabilistic model-building evolutionary algorithms in the sense that new solutions are proposed on the basis of information taken from the entire, current population. In the probabilistic model-building evolutionary algorithms a distribution is learned from the current population, and offspring is generated by sampling from this distribution [22].

Table 1 compares the EMCMC algorithms surveyed here according to their perturbation operators, their methods of the communication between chains, their EMCMC category, and their use as a sampler or optimiser. Call $X_t = (X_t[1], \ldots, X_t[N])$ the population at time t of N states (or individuals) $X_t[\cdot]$. To each individual $X[\cdot]$ we associate a *fitness function* $f : \Omega(X[\cdot]) \rightarrow I\!R$, where $X[\cdot]$ samples from $P'(X[\cdot]) = g(f(X[\cdot]))$ (g is a monotonic function, $g : I\!R \rightarrow I\!R^+ \setminus \{0\}$). Here, we consider an individual $X[\cdot]$ as a string of l characters $X[\cdot][1]X[\cdot][2]\ldots X[\cdot][l]$. We use $X[\cdot][\cdot]^a$ to denote the $a-$th value $\Omega(X[\cdot][\cdot])$, the set of all possible values of $X[\cdot][\cdot]$. We call $X[\cdot][j]^a$ an *allele* of $X[\cdot][j]$. The position of $X[\cdot][j]$ in $X[\cdot]$ is called the *locus* of $X[\cdot][j]$. X'_t, X''_t are two intermediate populations.

We present below the pseudo-code for a general framework EMCMC algorithm which contains both EMCMC categories.

```
$EMCMC()$
1  $Initialise\ X_0[i],\ i = 1, \ldots, N; t \leftarrow 0$
2  $Calculate\ S(\cdot \mid \cdot)$
3  **while** $true$
4      **do** $Sample\ X'_t\ from\ S(\cdot \mid X_t)$
5         $Recombine\ states\ from\ X'_t\ obtaining\ X''_t\ according\ to\ S(\cdot \mid \cdot)$
6         $Accept\ states\ from\ X''_t\ given\ X_t\ according\ to\ an\ acceptance\ rule\ obtaining\ X_{t+1}$
7         $Recalculate\ S(\cdot \mid \cdot)$
8         $t \leftarrow t + 1$
```

Evolutionary Monte Carlo (EMC). Liang and Wong [14], [15] propose the evolutionary Monte Carlo algorithm, which incorporates recombination into the parallel tempering (PT) algorithm to speed up the search and preserve good building blocks of the current states of the chains. Like PT, EMC has a population of MCMC chains with constant and (geometrically) increasing temperatures. Chains interact through the swapping acceptance rule A_S that attempts to exchange states between chains with adjacent temperature. This way, the good individuals sampled in the warmer chain are transfered to a colder chain where they may be preserved longer. The lowest temperature chain $Temp[1] = 1$ converges to the target distribution $P(\cdot)$, where $P'_i(\cdot) = \exp\left(\frac{-f(X[i])}{1}\right)$.

The candidate states are generated in two different ways and accepted by two different rules. With probability q_m each chain in the population generates a new state by mutation and accepts it with the standard Metropolis-Hastings acceptance rule. With probability $1 - q_m$ new states are generated by recombination of two states from different chains, and the offspring states are accepted with the *coupled acceptance rule*:

$$A_C((X_{NEW}[i], X_{NEW}[j]) \mid (X_t[i], X_t[j])) =$$

$$\min\left(1, \frac{P'_i(X_{NEW}[i]) \cdot P'_j(X_{NEW}[j])}{P'_i(X_t[i]) \cdot P'_j(X_t[j])} \cdot \frac{S(X'_t \mid X'_{NEW})}{S(X'_{NEW} \mid X'_t)}\right),$$

with $X'_t = (\ldots, X_t[i], \ldots, X_t[j], \ldots)$, $X'_{NEW} = (\ldots, X_{NEW}[i], \ldots, X_{NEW}[j], \ldots)$.

Note that when $q_m = 1$ EMC reduces to PT. Liang and Wong discussed experimental results for a model selection problem, a time series change-point identification problem, and for a protein folding problem. These experiments showed the effectiveness of EMC as compared to PT.

Multi-resolution Genetic Algorithm-style MCMC (MRGA). Holloman, Lee and Higdon [12] also proposed to integrate recombination in their multi-resolution parallel sintering algorithm. MRGA runs several chains for each resolution, sampling from the distribution at that resolution. Each individual has: (i) a variable size part, which represents information specific to its resolution, updated with a MCMC and (ii) a fixed dimensional (e.g. lowest dimension) vector with common interpretability used to move between chains (e.g. for image processing this could be the mean of neighbourhood cells). MRGA samples at the individual level using mutation, while it samples at the population level using recombination between the fixed dimensionality vector of two chains of the same or different resolutions. Both new individuals are accepted/rejected using the swapping acceptance rule A_S, which, in this case is equivalent with the coupled acceptance rule A_C. Like in parallel sintering, to improve mixing, chains of different resolutions periodically fully exchange the fixed dimensionality vectors using the swapping acceptance rule A_S. MRGA proposes with probability p_{swap} a full exchange, or with probability $1 - p_{swap}$ a crossover exchange.

Population MCMC (popMCMC). Laskey and Myers [13] introduced popMCMC where a population of MCMC chains exchange information through a population-based, adaptive proposal distribution. As before the Boltzmann distribution is used: $P'(X_t[\cdot]) = \exp^{-f(X_t[\cdot])}$,where f is a fitness function. Each generation a locus $X_t[i][j]$ is randomly picked to be mutated. An allele $X_t[i][j]^g$ on the j-th locus of the candidate individual $X_{NEW}[i]$ is generated using a distribution $\hat{\alpha}_{ija} = \frac{N(X_t[i][j]^a)+1}{N+|\Omega(X[\cdot][\cdot])|}$, where $N(X_t[i][j]^a)$ is the number of individuals that have the allele $X_t[i][j]^a$ on the j-th locus and $|\Omega(X[\cdot][\cdot])|$ is the total number of alleles. $X_{NEW}[i]$ is accepted using the MH acceptance rule $A(X_{NEW}[i] \mid X_t[i])$.

Since the proposal distribution depends on the current population, the transition matrix of a single individual is not stationary, yet the transition matrix of the whole population is stationary. Whenever good individuals from the population have a specific allele on a certain locus, new individuals will have with high probability the same allele on the same locus. Since popMCMC is a population of independently sampling MCMC chains, the allele remains in the population for a while. It is interesting to observe the similarity between popMCMC and univariate EDAs [22] in generating the candidate individuals from the structure of the current population. PopMCMC converges to the stationary distribution of N independently sampled points from $P(\cdot)$. The authors show experimentally that popMCMC finds highly likely Bayesian network structures faster than multiple independent MCMC chains.

Evolutionary MCMC (eMCMC). Zhang and Cho [27] proposed the eMCMC algorithm which is designed to find the optimum of a distribution by generating L samples from a population of N MCMC chains and then selecting the best N states of them $(L > N)$. The initial population is sampled according to a prior Gaussian distribution. Each generation, each chain from the current population generates several new candidate individuals using mutation and recombination which are accepted according to a Metropolis acceptance rule $A(\cdot \mid \cdot)$. The best N individuals from the L individuals are then selected for the next generation. Their posterior distribution - estimated with a Gaussian distribution model - represents the proposal distribution $S(\cdot \mid \cdot)$ for the next generation. We observe that eMCMC is also related with EDA [22] since the candidate individuals are generated from $S(\cdot \mid \cdot)$ which is adapted each generation. The eMCMC algorithm is an EMCMC algorithm which samples using mutation and recombination and where $S(\cdot \mid \cdot)$ is adapted each generation to sample from promising regions of the target distribution. Experimentally, eMCMC outperformed single chain MCMC and the standard GA for a system identification task.

Population-based simulated annealing (popSA). Goldberg [8] proposed a population-based simulated annealing algorithm which creates a Boltzmann distribution over the population. Instead of MH acceptance rule, it uses the *logistic acceptance rule* which has the probability to accept a candidate individual: $1/(1 + \exp\frac{f(X_t[\cdot])-f(X_{NEW}[\cdot])}{Temp[t]})$, where $Temp[t]$ is the temperature for generation t. When $Temp[t]$ is constant, the unnormalised stationary distribution of this algorithm is the Boltzmann distribution $P'_t(X_t[\cdot]) = \exp\frac{f(X_t[\cdot])}{Temp[t]}$, where f is the fitness function. Each individual $X_t[i]$ from the current population chooses two other individuals from the population for coupling: one $X_t[j]$ has a fitness value different from $X_t[i]$ by a threshold θ and one $X_t[k]$ has a fitness value different from $X_t[i]$ or from both $X_t[i]$ and $X_t[j]$ by a threshold θ. An anti-acceptance competition is held between $X_t[j]$ and $X_t[k]$, where $X_t[j]$ is accepted with probability $1/(1 + \exp\dfrac{f(X_t[j]) - f(X_t[k])}{Temp[t]})$. The primary acceptance competition is held between the winner of the first competition $X_t[jk]$ and $X_t[i]$, where $X_t[i]$ is accepted with probability $1/(1 + \exp\dfrac{f(X_t[jk]) - f(X_t[i])}{Temp[t]})$. The competitions are chosen to avoid the danger that copies of an individual are taking over the population. The anti-acceptance rule prefers poorer individuals, whereas the primary acceptance rule prefers the better individuals. This way, the population equivalent is created to generate a neighbour of the population at random.

Parallel recombinative simulated annealing (PRSA). Mahfoud and Goldberg [19] proposed a population-based simulated annealing algorithm which also made use of recombination. All individuals from the population have the same temperature which decreases every generation according to a cooling schedule. New individuals are generated using mutation and one point recombination between two individuals $X_t[i]$ and $X_t[j]$. PRSA uses the logistic acceptance rule to accept a candidate individual. Two possible competitions are considered: single acceptance/rejection holds two competitions between a parent vs. the child formed from its own right-end and the other parent left-end, or double acceptance/rejection holds one competition between both parents vs. both children using the coupled acceptance rule:

$$A_C((X_{NEW}[i], X_{NEW}[j]) \mid (X_t[i], X_t[j])) =$$

$$\min(1, 1/(1 + \exp\frac{f(X_t[i]) + f(X_t[j]) - f(X_{NEW}[i]) - f(X_{NEW}[j])}{Temp[t]}))$$

Massively parallel simulated annealing (mparSA). Rudolph [23] introduced the massively parallel simulated annealing algorithm. He experimentally achieved fast convergence to the optimum with a population of independent SA chains. Like SA, mparSA uses the Boltzmann function $P'_t(X_t[\cdot]) = \exp(-\frac{f(X_t[\cdot])}{Temp[t]})$, where f is a fitness function and $Temp[t]$ is the temperature for all individuals from the t-th generation. The algorithm associates the population with a connected graph. Each node has an individual which communicates with the individuals in the neighbouring nodes in the graph. For each chain, each step, the current state $X_t[i]$ is recombined with a neighbour and the result is mutated several times obtaining new neighbours. The best candidate individual $X_{NEW}[i]$ is then selected and is accepted with the MH acceptance rule $A(X_{NEW}[i] \mid X_t[i])$. The temperature is decreased each step according to a SA cooling scheduler. Rudolph pointed out that the conflicting goals of fast convergence and global convergence to the optimum can be satisfied with an adaptive proposal distribution, whereas it cannot with a fixed proposal distribution. As in Evolutionary Strategies, mparSA uses a non-fixed proposal distribution $S_t(\cdot \mid \cdot)$ which is adapted each generation with a lognormally distributed random variable α_t: $S_t(\cdot \mid X_t[i]) = \alpha_t S_{t-1}(\cdot \mid X_t[i])$.

Related work. Cercueil and Francois [3] give an overview of literature where EAs are viewed as Monte Carlo methods which generates sample from a probability distribution defined on the trajectories of their population. This helps to unify the convergence theories for EAs. For doing that, Cerf [4] has a GA with the mutation rate related to a temperature, no crossover, and Boltzmann roulette selection. Each generation, an intermediate population is generated by mutation. The next population is generated from a Boltzmann distribution constructed from the intermediate population. Lozano and co-authors discuss a hybrid between the genetic algorithm and simulated annealing based on a probabilistic Boltzmann reduction operator [17]. In a very recent publication Strens proposes an EMCMC algorithm with an 'exclusive-or' proposal operator [24]. This operator takes one parent and two reference states, and generates an offspring state that disagrees from its parent in a similar way as the two reference states.

Sequential Monte Carlo (SMC) is a new branch of Monte Carlo simulation, not necessarily a Markov chain, which uses a population of samples to carry out on-line approximations of a target distribution. SMC has sampling procedures which are similar to proportional selection in EAs. Bienvenue et al. introduce niching in some SMC algorithms to maintain diversity in the population [2]. Del Moral [21] study the convergence of GAs with SMC. Higuchi [11] and Tito, Vellasco and Pacheco [26] proposes GA filter and Genetic Particle Filter, respectively. They integrate recombination into SMCs [11] to speed up convergence.

5 Elitist coupled acceptance rule and fitness ordered tempering

In the previous section we have surveyed a number of EMCMC algorithms, and discussed some techniques that showed how multiple MCMC chains could exchange information in order to speed up the convergence to some target distribution. In the following we introduce the elitist coupled acceptance rule (ECA) and fitness ordered tempering (FOT), whose main purpose is to converge efficiently to a target distribution that is biased towards the most likely states (or good solutions).

Fitness ordered tempering (FOT). FOT maintains a population of N chains each having its own temperature $Temp[i]$. As in parallel tempering, FOT has a - typically geometrical - increasing series of temperatures $Temp[1] = 1 < \ldots < Temp[N] = Temp^{max}$. The population of N solutions (or states) is sorted according to their fitness (or probability), and the solution at rank i gets the temperature $Temp[i]$, where the most fit solution gets $Temp[1] = 1$, and the worst solution $Temp[N] = Temp^{max}$. Therefore a solution has lower temperature than any solution worse than itself, unless they are copies of each other. In case there are multiple copies of the same solution ties within the sorted population are broken randomly, so each copy gets an adjacent temperature. Copies receive a different temperature to avoid that multiple copies of good solutions will remain in the population almost indefinitely.

In case there are multiple solutions with the same fitness but who are not copies of each other, the temperature ladder has to be recomputed so that each unique solution with the same fitness gets the same temperature. The number of different temperature values $Temp[i]$ at each generation is therefore equal to the number of solutions with different fitness value, unless they are copies, in which case they also get a different temperature value. This scheme is necessary to avoid that some solution might get another temperature in an identically composed population, and ensures that FOT has a homogeneous Markov chain transition matrix at the population level. Contrary to parallel tempering, here the temperature assignment depends on the fitness of the solutions relative to the fitness of the other solutions in the current population. Since we want to remain in the vicinity of good solutions, we prefer to assign the lower temperatures to the better solutions.

Elitist coupled acceptance rule (ECA). The ECA algorithm applies a coupled Metropolis Hastings acceptance rule to two solutions and their two children. ECA accepts the best two solutions from the family of four if at least one of them is a child. However, when both children have a lower fitness than both their parents, the children can still replace the parents with a probability determined by the coupled acceptance rule. This increases the explorative character of the algorithm. Call $X_t' = (\ldots, X_t[i], \ldots, X_t[j], \ldots)$, $X'_{NEW} = (\ldots, X_{NEW}[i], \ldots, X_{NEW}[j], \ldots)$, $P'_{X_t}(X[\cdot]) = \exp\left(\frac{f(X_t[\cdot])}{Temp[\cdot]}\right)$, and $\max_2$ the function returning the two most fit solutions. The ECA acceptance rule is now given as:

```
ECA$(((X_{NEW}[i], X_{NEW}[j]) \mid (X_t[i], X_t[j])))$
1  if $\{X_t[i], X_t[j]\} = \{X_{NEW}[i], X_{NEW}[j]\}$
2    then return $1$
3  $(X_{MAX}[i], X_{MAX}[j]) \leftarrow \max_2 (X_{NEW}[i], X_{NEW}[j], X_t[i], X_t[j])$
```

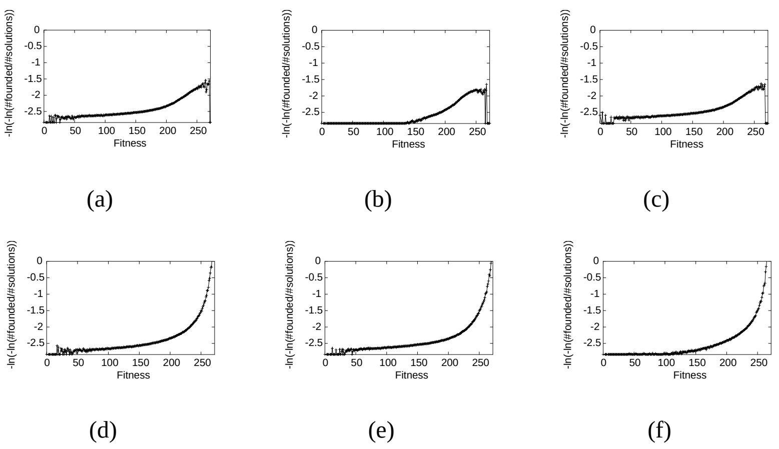

Fig. 1. The number of different solutions found over the total number of possible solutions for a fitness value on a logarithmic scale for (a) MIC, (b) popMCMC, (c) PT, (d) PT with recombination, (e) PRSA and (f) ECA/FOT

```
4  if $X_{MAX}[i] \in \{X_{NEW}[i], X_{NEW}[j]\} \vee X_{MAX}[j] \in \{X_{NEW}[i], X_{NEW}[j]\}$
5      then return $1$
6      else return $\frac{P'_{X_t}(X_{NEW}[i])\cdot P'_{X_t}(X_{NEW}[j])}{P'_{X_t}(X_t[i])\cdot P'_{X_t}(X_t[j])} \cdot \frac{S(X'_t|X'_{NEW})}{S(X'_{NEW}|X'_t)}$
```

Note that the temperatures of the current and accepted solutions remain the same. ECA can be viewed as a combination between the family-competitive, elitist replacement rule from regular GAs [25] and the coupled acceptance rule A_C. To see the difference between ECA and A_C let us consider 2 parent states and their offspring such that $P'_{X_t}(X_{NEW}[i]) > P'_{X_t}(X_t[j]) > P'_{X_t}(X_t[i]) > P'_{X_t}(X_{NEW}[j])$. Call $F = \frac{P'_{X_t}(X_{NEW}[i])}{P'_{X_t}(X_t[i])} \cdot \frac{P'_{X_t}(X_{NEW}[j])}{P'_{X_t}(X_t[j])}$. If $F < 1$, A_C may loose $X_{NEW}[i]$ which is the best solution of this family, while if $F > 1$, A_C accepts $X_{NEW}[j]$ which is the worst solution.

Combining the FOT temperature assignment mechanism and the ECA acceptance rule leads to the ECA/FOT MCMC algorithm: each generation the temperatures of each individual in the population are recomputed with FOT, new individuals are proposed by mutation and recombination, and are accepted - or rejected - with the ECA acceptance rule. It is important to note that at the individual level the transition matrix of ECA/FOT is not stationary, since, in time, the same fitness values may be associated with different temperatures. At the population level however, the transition matrix is stationary because for the same population of individuals, the temperature is always assigned in the same way. The ECA/FOT MCMC is aperiodic, since, for each state, there is a non-zero probability to remain in the current state, and irreducible, because it has a non-zero probability to arrive from each state to each other state. If the state space is finite, ECA/FOT converges to a distribution $R(\cdot) = \prod_{N}^{i=1} R'(\cdot)$, where $R'(\cdot)$ is the unnormalised distribution for a single chain which depends on $Temp_{MAX}$ and $Temp_{MIN}$, the temperature assignment (e.g. geometrically) and the size of the population.

6 Experimental results

To illustrate the use of the ECA/FOT algorithm we have applied 6 (E)MCMC algorithms to the well known deceptive trap function [9], and counted the number of different solutions visited as a function of their fitness. The 6 algorithms are multiple independent MCMC chains (MIC), population MCMC (popMCMC), parallel tempering (PT), parallel tempering with recombination, parallel recombinative simulated annealing (PRSA), and elitist coupled acceptance with fitness ordered tempering (ECA/FOT). The first three algorithms do not apply recombination, while the other three do. For comparison purposes, we sample for each algorithm from the Boltzmann distribution $\exp\left(\frac{f(\cdot)}{Temp[\cdot]}\right)$. In the MIC algorithm, each chain has been given a constant temperature calculated with the PT ladder. The PRSA algorithm uses the MH acceptance rule A (instead of the logistic acceptance rule), and two single acceptance/rejection competition between a parent and one of its children.

The trap function has 10 building blocks of length $k = 3$ and a linearly scaled fitness function: $f = \sum_{i=1}^{10} i \cdot f_i$. f_i is the fitness value of the i-th trap function and depends only on the number of one bits at each building block: $f_i(3) = 5, f_i(2) = 0, f_i(1) = 3, f_i(0) = 4$. The algorithmic parameters are chosen as follows. We set $Temp_{MAX} = 160$ for the MH acceptance rule and $Temp_{MAX} = 320$ for the ECA acceptance rule, resulting in an acceptance probability of 0.73 for two individuals with fitness difference equal to 50. $Temp_{MIN}$ is equal to 1. We choose a geometrically increasing temperature ladder that increases the temperature each step with $\beta = \exp\left(\frac{1}{s} \cdot \log \frac{Temp_{MIN}}{Temp_{MAX}}\right)$, where s is the total number of steps for the scheduler. At step j, $Temp_j = Temp_{MAX} \cdot \beta^j$. We choose the population size N equal with the number of generations to obtain the same β for all algorithms. If $\beta = 0.97$, we obtain $N = 250$. The mutation rate is $\frac{3}{l}$, which is the optimal rate for a deceptive function with building block length $k = 3$. The recombination operator is two point crossover . For each algorithm, we have made 10 independent runs and sampled the whole population every 10 generations, resulting in 25 samples and 6250 solutions in total.

Figure 1 shows - as expected - the advantage of using recombination as proposal operator for functions (or distributions) whose structure can be exploited by the crossover operator. One can also notice that ECA/FOT is more biased towards highly likely individuals than PT or PRSA.

7 Conclusion

Evolutionary Markov Chain Monte Carlo combine techniques form Evolutionary Computation and parallel Markov Chain Monte Carlo algorithms to design new algorithms for sampling or optimising complex distributions resp. functions. EMCMC algorithms offer new proposal operators and new acceptance rules. Individual states in the population can be single MCMC chains that interact with each other, though it is not necessary that they are indeed single MCMC chains. At the population level however - this is, states are entire populations of given size - EMCMC algorithms need to be MCMC chains.

We have surveyed a number of existing EMCMC algorithms in the literature, and categorised them in two classes: *family-competitive EMCMC* and *population-driven EMCMC* algorithms. We have also introduced an EMCMC algorithm, ECA/FOT, which applies a Fitness Ordered Temperature assignment mechanism and an Elitist Coupled Acceptance rule. EAC/FOT is, at population level, an MCMC which converges to a stationary distribution biased towards the most likely states.

Clearly, the merger of the EC and MCMC paradigms represents a rich source for future development of powerful sampling and optimisation algorithms.

References

1. C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan. An introduction to MCMC for Machine Learning. *Machine Learning*, pages 5–43, 2003.
2. A. Bienvenüe, M. Joannides, J. Bérard, E. Fontenas, and O. François. Niching in Monte Carlo filtering algorithms. In E. Lutton M. Schoenauer P. Collet, JK Hao, editor, *Artificial Evolution Selected Papers, 5th International Conference, EA 2001*, pages 19–30. Springer, 2002.
3. A. Cercueil and O. François. Monte Carlo simulation and population-based optimization. In *Congress on Evolutionary Computation 2001*, pages 191–198, 2001.
4. R. Cerf. A new genetic algorithm. *Annals of Applied Probabilities*, 6:778–817, 1996.
5. J. Culberson. Genetic invariance: A new paradigm for genetic algorithm design. Technical Report 92-02, Edmonton, Canada, 1992.
6. C. J. Geyer. Markov Chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 156–163, 1991.
7. W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. *Markov Chain Monte Carlo in practice*. Chapman & Hall, 1996.
8. D. E. Goldberg. A note on Boltzmann tournament selection for Genetic Algorithms and Population-oriented Simulated Annealing. *Complex Systems*, 4:445–460, 1990.
9. D. E. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and Genetic Algorithms. In *Proceedings of Parallel Problem Solving from Nature*, pages 37–46, 1992.
10. W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.
11. T. Higuchi. *Self-Organizing Time Series Model*, volume Sequential Monte Carlo Methods in Practice, pages 429 – 444. Springer, 2001.

12. C. H. Holloman, H. K. H. Lee, and D. M. Higdon. Multi-resolution Genetic Algorithms and Markov Chain Monte Carlo. Technical report, Duke University, 2002.
13. K. B. Laskey and J. W. Myers. Population Markov Chain Monte Carlo. *Machine Learning*, pages 175–196, 2003.
14. F. Liang and W. H. Wong. Evolutionary Monte Carlo: Applications to C_p model sampling and change point problem. In *Statistica Sinica*, pages 317–342, 2000.
15. F. Liang and W.H. Wong. Evolutionary Monte Carlo for protein folding simulations. In *Journal of Chemical Physics*, number 115, pages 3374–3380, 2001.
16. J.S. Liu and C. Sabatti. Simulated Sintering: Markov Chain Monte Carlo with spaces of varying dimensions. In *Bayesian Statistics 6*, pages 389–413. Oxford University Press, 1999.
17. J. A. Lozano, P. Larrañaga, M. Graña, and F. X. Albizuri. Genetic algorithms: bridging the convergence gap. *Theoretical Computer Science*, 229(1–2):11–22, 1999.
18. S. W. Mahfoud. Crowding and preselection revisited. In Reinhard Männer and Bernard Manderick, editors, *Parallel problem solving from nature 2*, pages 27–36, Amsterdam, 1992. North-Holland.
19. S. W. Mahfoud and D. E. Goldberg. Parallel Recombinative Simulated Annealing: a Genetic Algorithm. *Parallel Computing*, pages 1–28, 1995.
20. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Jornal of Chemical Phisics*, 21:1087–1092, 1953.
21. P. Moral, L. Kallel, and J. Rowe. Modelling Genetic algorithms with Interacting Particle systems. In *Theoretical Aspects of Evolutionary Computing*, pages 10–67. L. Kallel, B. Naudts, A. Rogers, 2001.
22. M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21:5–20, 2002.
23. G. Rudolph. Massively Parallel Simulated Annealing and its Relation to Evolutionary Algorithms. *Evolutionary Computation*, pages 361–382, 1994.
24. M. J. A. Strens. Evolutionary MCMC sampling and optimization in discrete spaces. In *Proceedings of the Twentieth International Conference on Machine Learning ICML-2003*, 2003.
25. D. Thierens and D.E. Goldberg. Elitist recombination: an integrated selection-recombination GA. In *Proceedings of the First IEEE World Congress on Computational Intelligence*, pages 508 – 512, 1994.
26. E.A.H. Tito, M.M.B.R. Vellasco, and M.A.C. Pacheco. Genetic Particle Filter: An evolutionary perspective of SMC methods. Technical report, Catolic University of Rio de Janeiro, 2002.
27. B. T. Zhang and D. Y. Cho. System identification using evolutionary Markov Chain Monte Carlo. *System Architecture*, pages 287–599, 2001.