Population annealing and large scale simulations in statistical mechanics

Lev Shchur$^{1,2[0000-0002-4191-1324]}$, Lev Barash$^{1,3[0000-0002-2298-785X]}$, Martin Weigel$^{4[0000-0002-0914-1147]}$, and Wolfhard Janke$^{5[0000-0002-5165-9097]}$

[1] Landau Institute for Theoretical Physics, 142432 Chernogolovka, Russia
[2] National Research University Higher School of Economics, 101000 Moscow, Russia
[3] Science Center in Chernogolovka, 142432 Chernogolovka, Russia
[4] Applied Mathematics Research Centre, Coventry University, Coventry, CV1 5FB, United Kingdom
[5] Institut für Theoretische Physik, Universität Leipzig, Postfach 100 920, 04009 Leipzig, Germany

Abstract. Population annealing is a novel Monte Carlo algorithm designed for simulations of systems of statistical mechanics with rugged free-energy landscapes. We discuss a realization of the algorithm for the use on a hybrid computing architecture combining CPUs and GPGPUs. The particular advantage of this approach is that it is fully scalable up to many thousands of threads. We report on applications of the developed realization to several interesting problems, in particular the Ising and Potts models, and review applications of population annealing to further systems.

Keywords: Parallel algorithms · Scalability · Statistical mechanics · Population annealing · Markov Chain Monte Carlo · Sequential Monte Carlo · Hybrid computing architecture CPU+GPGPU.

1 Introduction

Over the last decade or so, the development of computational hardware has followed two main directions. The first is based on the parallelization of devices based on traditional silicon technology. This approach splits in turn into two main groups. The first concerns the further development of CPUs featuring a growing number of cores placed onto a single silicon die. This direction is sometimes also known as the heavy-core approach, expressing the fact that these cores are as complex as traditional full CPUs themselves. Currently, the typical maximum number of such cores is of the order of 30. The second group concerns the development of auxiliary computing devices, which in turn is represented by mainly two subgroups. The first type is based on Intel's Xeon Phi architecture, featuring more than 70 relatively light, but otherwise fully developed x86 cores on a single chip. The second and much larger group comprises graphics processing units (GPUs) featuring several thousands of light cores in one device. The use of GPUs in general purpose and scientific computing is sometimes known as GPGPU – general purpose GPU, used outside of graphics applications. It is mostly due to the progress of development within this stream that Moore's famous law still holds today, even at the level of supercomputers.

The second main direction of recent development of computational hardware concerns the construction of computers based on quantum bits (qubits). This development also splits into (at least) two main streams. Quantum annealers or analog quantum computers calculate the ground state for some class of Hamiltonians. The most famous examples in this class are the D-Wave machines with currently up to about 2000 qubits annealed in parallel. Digital quantum computers, on the other hand, are based on logic realizing quantum operators. Examples of this approach include IBM's 16 qubit universal quantum computing processor as well as the recently announced Google 72 qubit quantum processor Bristlecone.

The big challenge for scientific computing in this landscape is to develop algorithms and computational frameworks that use the available hardware efficiently. Despite the great attention that quantum computers attract worldwide, their use for actual calculations is still quite restricted. In the case of the D-Wave machines, this shortcoming is connected to the fixed Hamiltonian simulated in the annealing that features a rather special topology of connections between qubits. Another open issue relates to the problem of decoherence, where progress is being made by further lowering the operating temperature and minimizing other noise in the system. At present, quantum annealers have not been able to demonstrate a fundamental speed-up against classical machines in the sense of applications to problems that are exponentially hard on classical machines and only polynomial on the quantum computer. Digital or universal quantum computers are at present mostly restricted by the limited size of the available realizations, but more generally there is only a rather limited number of known quantum algorithms showing the profound speed-up against classical methods mentioned above, the most prominent example being Shor's factorization algorithm. Another difficulty concerns error correction, with one of the possible solutions being Kitaev's topological quantum computations based on anyons [1].

Currently available universal supercomputers are still based on the traditional silicon multicore architecture. The motivation for building supercomputers are particularly challenging problems in scientific computing that require such extensive resources. It is very demanding, however, to efficiently use all the power of a supercomputer in a single run. Such a program should be able to run on potentially millions of cores. Hence the computation must be divided into millions of tasks to be scheduled on individual cores. It is thus of crucial importance to develop new, fully scalable algorithms, new programming techniques, and new methods to build programs which can efficiently use the power of supercomputers.

The problem is even more complicated when running code on hybrid supercomputers, with auxiliary accelerated computing devices in addition to the conventional CPUs. In this case, one needs to take into account the specific architecture of the auxiliary devices, too. This includes the specific arithmetic/logical operations, the specific memory organization, and the specific input/output layout.

In this paper we present a mini-review of a recently developed parallel algorithm, dubbed "population annealing", for simulations of spin and particle systems. The idea of this approach goes back to the beginning of the century [2, 3], and the algorithm as we use it now was presented about a decade later in Ref. [4]. The most promising feature of this algorithm is its near ideal scalability both on parallel and hybrid architectures.

2 Population annealing

The population annealing algorithm consists of two alternating steps. The first is a cooling step [2], where equilibrium is maintained by differential reproduction (resampling) of replicas. This part is a realization of a sequential Monte Carlo algorithm. At each temperature step the resampling propagates the population from inverse temperature $\beta_i = 1/k_B T_i$ to the target distribution at inverse temperature $\beta_{i+1} = \beta_i + \Delta\beta_i$. The second step is an equilibration of replicas. It is performed for each replica independently, and any Markov Chain Monte Carlo (MCMC) algorithm can be used. The initial configurations at β_0 should be chosen at equilibrium. Typically $\beta_0 = 0$, where equilibrium can be easily guaranteed.

In total the algorithm comprises the following steps:

1. Set up an equilibrium population of $R_0 = R$ independent copies (replicas) at inverse temperature β_0.
2. Propagate the population to the inverse temperature β_i ($i = 1, 2, \ldots$): Resample replicas $j = 1, \ldots, R_{i-1}$ with their normalized Boltzmann weights $\hat{\tau}_i(E_j) = (R/R_{i-1}) \exp\left[-(\beta_i - \beta_{i-1})E_j\right]/Q_i$, where
$$Q_i = \sum_{j=1}^{R_{i-1}} \frac{e^{-(\beta_i-\beta_{i-1})E_j}}{R_{i-1}}. \qquad (1)$$
3. Update each replica by θ sweeps of the chosen MCMC algorithm at inverse temperature β_i.
4. Calculate estimates for observables $\mathcal{O}$ as population averages $\sum_j \mathcal{O}_j/R_i$.
5. Goto step 2 unless the target temperature β_f has been reached.

In addition to the basic algorithm, it was pointed out in Ref. [4] that both statistical errors as well as bias can be reduced by combining the results of several independent population annealing simulations. Bias is minimized if the combination is performed as a weighted average of the results from M runs, where the weight of the m-th simulation should be chosen as

$$\omega_m(\beta_i) = \frac{e^{-\beta_i \hat{F}_m(\beta_i)}}{\sum_{m=1}^{M} e^{-\beta_i \hat{F}_m(\beta_i)}}. \qquad (2)$$

Here, $\hat{F}_m$ corresponds to an estimate of the free energy from the m-th run that can be deduced from the normalization factors Q_i in Eq. (1), namely [4]

$$-\beta_i \hat{F}(\beta_i) = \ln Z_{\beta_0} + \sum_{k=1}^{i} \ln Q_k, \tag{3}$$

where Z_{β_0} is the partition function at the initial inverse temperature β_0. At least for $\beta_0 = 0$ this can often be determined analytically. It is clear that statistical errors decrease with the size R of the population as $1/\sqrt{R}$, and it was proposed in Ref. [4] that, asymptotically, the bias decays as $1/R$.

It follows that statistical as well as systematic errors can be arbitrarily reduced by adding additional parallel resources used to simulate a larger population. Consequently, population annealing (PA) by construction is an ideally scalable algorithm. With a well-designed implementation, the approach should allow one to efficiently utilize nearly arbitrarily large parallel resources (supercomputers). In particular, the method is well suited for GPUs which are known to be most effective if the number of parallel threads significantly exceeds the number of available cores and latency hiding works well [5]. For GPUs with a few thousand cores, efficient operation is guaranteed with a total of several ten to hundred thousand threads, where each thread simulates a single replica [6]. As a consequence, it is possible to efficiently simulate hundreds of millions of replicas on compute clusters with hybrid CPU+GPU architecture.

3 Algorithmic improvements

A number of algorithmic improvements to the algorithm as discussed in the previous section have been attempted [6–8]. From the description of the algorithm one identifies a number of tunable parameters: the temperature step $\Delta\beta_i = \beta_i - \beta_{i-1}$, the number of MCMC rounds θ, and the size R of the population. For the approach using weighted averages, one can additionally vary the number M of independent runs. Optimal values of these parameters will depend on the model simulated and on the prescribed accuracy. In Section 5 we present some examples of simulations of different models, but first we discuss some considerations regarding the parameter choice.

We first turn to the choice of temperature step. For too large a step the resampling factors will be dominated by rare events, leading to very low diversity in the population after resampling. Very small temperature steps, on the other hand, lead to resampling factors that are all very close to unity, and so the resampling just leaves the population invariant. How can a close to optimal value of the temperature step be found? In Ref. [6] we have proposed an adaptive scheme for choosing temperature steps to achieve this. It calculates the overlap of energy histograms between the neighbouring temperatures, and either increases or decreases the inverse temperature step in a way to keep the overlap of neighboring energy histograms fixed. We find that values of histogram overlap between 50% and 90% usually provide a good compromise between sufficient histogram overlap without an unnecessary proliferation of temperature steps. If the simulation procedure needs to use averaging over independent runs, obtaining the sequence of temperatures via calculations of histogram overlap is carried out only once, during the first run, and the rest of the runs use the same annealing schedule.

Regarding θ and R, it turns out that θ needs to have a (model dependent) minimal value to ensure that the anneal remains in equilibrium, especially if the energy is a slowly relaxing variable. Beyond that, increasing R reduces statistical as well as systematic errors. A detailed discussion of the dependence of performance on these parameters can be found in Ref. [7].

Another algorithmic improvement is given by the multi-histogram reweighting method, also referred to as Weighted Histogram Analysis Method (WHAM) [9–11]. Assume we have performed a PA simulation with inverse temperature points $\beta_1, \beta_2, ..., \beta_K$ and population sizes R_i, where $i = 1, ..., K$. The multi-histogram reweighting technique is based on reweigthing the measurements from all temperatures to a chosen reference point and combining them by calculating error weighted averages. If we have histograms $P_{\beta_i}(E)$ at the inverse temperatures β_i (normalized such that $\sum_E P_{\beta_i}(E) = R_i$), we obtain from the error-weighted combined histogram at the common reference point [6]

$$\Omega(E) = \frac{\sum\limits_{i=1}^{K} P_{\beta_i}(E)}{\sum\limits_{i=1}^{K} R_i \exp[\beta_i(F_i - E)]}, \tag{4}$$

which is an estimate of the density of states (DOS). As we have the estimates (3) from a PA run at hand, we can immediately evaluate (4) and in many cases find good results without further iterations (which, nevertheless, can always be used to further improve the accuracy). Note that if one refrains from using iterations, one needs to store and update at each temperature step only the single partially summed histogram $\sum_{i=1}^{K} P_{\beta_i}(E)$ for each energy. Therefore, memory requirements are quite moderate and, for example, for the 2D Ising model only $\sim L^2$ values have to be stored in total.

4 GPU accelerated PA algorithm

As explained above, the PA algorithm appears to be ideally suited for implementations on massively parallel hardware. We presented an implementation of PA for the Ising model on Nvidia GPUs in Ref. [6]. The individual GPU kernels are [6, 12]:

(1) initialization of the population of replicas (kernel `ReplicaInit`)
(2) equilibrating MCMC process (kernel `checkKerALL`)
(3) calculation of energy and magnetization for each replica (kernel `energyKer`)
(4) calculation of $Q(\beta, \beta')$ (kernel `QKer`)
(5) calculation of the number of copies n_i of each replica i (kernel `CalcTauKer`)

(6) calculation of the partial sums $\sum_{i=1}^{j} n_i$, which identify the positions of replicas in the new population (kernel `CalcParSum`)
(7) copying of replicas (kernel `resampleKer`)
(8) calculation of observables via averaging over the population (kernel `CalcAverages`)
(9) calculation of histogram overlap (kernel `HistogramOverlap`)
(10) updating the sum of energy histograms $\sum_{i=1}^{K} P_{\beta_i}(E)$ for the multi-histogram reweighting (kernel `UpdateShistE`)

Some details for the most important GPU kernels (1)–(4) are as follows:

(1) *Equilibration process.* To create sufficient parallel work for the GPU devices, it turns out to be useful to combine the replica-level parallelism with an additional domain decomposition, thus exploiting also spin-level parallelism. The basic step consists of a checkerboard decomposition of the lattice which allows for independent updates of all spins of one sub-lattice [13]. The code works with thread blocks of size `EQthreads`, which should be a power of two. The optimal value of `EQthreads` is 128 for $L < 128$, and it is sometimes beneficial to increase it to 256, 512 or 1024 in the case of large system size $L \geq 128$. Each block of threads works on a single replica of the population, using its threads to update tiles of size $2 \times$ `EQthreads` spins. To this end it flips spins on one checkerboard sub-lattice first, moving the tiles over the lattice until it is covered, synchronizes and then updates the other sub-lattice. An important (if well known) trick for optimization is that the value of the exponential function for the Metropolis criterion should not be calculated at each spin flip, since the exponential is not a fast single-cycle operation. However, there are only a few possible values of ΔE. For example, for the 2D Ising model, $\Delta E = 2Js_i \sum_{\mathrm{neighb}} s_j + 2Hs_i$. A lookup table containing 10 possible values of $\min[1, \exp(-\beta\Delta E)]$ is placed in the fast *texture* memory of the GPU for an optimal performance.
(2) *Calculation of energies and other observables.* The GPU parallel reduction algorithm is employed. First, each of the $N = 2^n$ threads calculates one particular summand. All calculated N summands $s_1, \ldots, s_N$ are placed in shared memory. Then, the first $N/2^i$ threads perform $s_i := s_i + s_{i+N/2^i}$ for $i = 1, 2, 3, \ldots$. This scheme can be visualized as a binary tree [14]. For the case of modern devices with CUDA compute capability ≥ 3.0, we use the "shuffle" operations that allow threads to access registers from different threads in the same warp. The latter approach is faster, but it is not supported by older devices, where we use the former method and store partial results in shared memory.
(3) *Calculation of the normalization factor* $Q_i = \frac{1}{R_{i-1}} \sum_{j=1}^{R_{i-1}} \exp[-(\beta_i - \beta_{i-1})E_j]$. Since Q_i is a large sum, each of the summands can be independently calculated with one GPU thread. Then, the same reduction method as in (2) can be used.
(4) *Calculation of numbers of copies* n_i. For each replica, the values τ_i and n_i are calculated by a separate GPU thread.

In the present program we use the random number generator `PHILOX` available also in the CURAND library [15]. Alternatively, one might employ the generators previously developed by some of us in Refs. [16–19]. These libraries allow to use up to 10^{19} uncorrelated parallel streams of random numbers.

5 Applications

In this Section we discuss a number of applications of the method, including the Ising model that undergoes a continuous phase transition, the Potts model with an additional first-order regime, models with disorder and frustration as well as simulations of off-lattice systems.

5.1 Ising model and second-order phase transitions

The Ising model plays the same role in statistical mechanics as the fruit fly in biology. New algorithms and other new ideas are usually tested on the Ising model first. It is the first model of statistical mechanics which was proven to have singularities near the spontaneous transition from the disordered to the ordered phase. The Ising model is the simplest model of a ferromagnet, for example for a piece of iron that develops a spontaneous magnetic moment below the Curie temperature T_c, while it is paramagnetic above T_c. The ferromagnetic Ising model in zero external magnetic field and on a square lattice of linear size L is given by the Hamiltonian

$$H = -J \sum_{\langle ij \rangle} s_i s_j, \tag{5}$$

where $J > 0$. The summation is restricted to nearest neighbors and the variables s_i can take one of two values, $s_i = +1$ or $s_i = -1$. Periodic boundary conditions are assumed. The ground state is achieved when all variables are equal, i.e., either all of them take the value $+1$ or all of them are -1. Hence the ground state is two-fold degenerate. The interpretation is that the variables s_i can be viewed as magnetic moments, pointing only in two directions, up for the $+1$ value of s_i, or down for -1. In other words, all spins are aligned in the ground state, and the total magnetic moment $|M|$ of the system is equal to the maximal possible value L^2 and the energy E is equal to $-2L^2J$. This ground state is reached at zero temperature. For high temperatures the spins will take random orientations and, consequently, the average magnetic moment will be zero. A phase transition occurs at inverse temperature $\beta_c = J/k_B T_c = \ln(1+\sqrt{2})/2$ [20]. The model admits an exact solution, which was first derived by Onsager [21]. The resulting detailed knowledge about its non-trivial behavior turns this model into an ideal testing ground for new ideas.

Simulating the square-lattice Ising model with population annealing as implemented on GPU and a reference CPU implementation, we show in Table 5.1 the performance of the two codes in terms of time per spin flip in nanoseconds, for $\theta = 500$ and a population size $R = 50\,000$ for different system sizes. The GPU code is found to be about 620 times faster than the sequential CPU program. The

Table 1. Peak performance of the CPU and GPU PA implementations in units of the total run time divided by the total number of spin flips performed, for different system sizes. The best GPU performance is achieved for large θ, and here $\theta = 500$ was chosen for a population of $R = 50\,000$ replicas. GPU performance data are for the Tesla K80 (Kepler card) and GeForce GTX 1080 (Pascal card). The sequential CPU code was benchmarked on a single core of Intel Xeon E5-2683 v4 CPU running at 2.1 GHz.

	time per spin flip (ns)		
	CPU	GPU (Kepler card)	GPU (Pascal card)
$L = 16$	23.1	0.094	0.038
$L = 32$	22.9	0.092	0.034
$L = 64$	22.6	0.092	0.036
$L = 128$	22.6	0.097	0.039

additional application of a multi-spin coding technique yields a further speedup of up to 10 times for the GPU implementation, for details see Ref. [6].

Figure 1 shows the temperature steps generated by our adaptive stepping algorithm applied to the two-dimensional Ising model with $J = 1$, while imposing a histogram overlap of approximately 85% (see also Sec. 3). One can see that the closest inverse temperature spacing is found close to the transition point, as expected. The adaptive algorithm automatically suggests the optimal annealing schedule and often saves a significant amount of computing time.

5.2 Potts model and first-order phase transitions

A natural generalization of the Ising model is the q-states Potts model with Hamiltonian

$$H = -J \sum_{\langle ij \rangle} \delta_{s_i s_j}, \tag{6}$$

where the spins s_i can take q values, $s_i = 1, 2, ..., q$, and $J > 0$ is the ferromagnetic coupling constant. We study the model on the square lattice with periodic boundary conditions, where the sum in Eq. (6) goes over all nearest-neighbour pairs $\langle ij \rangle$. Below we set $J = 1$ to fix units. The transition temperature of the model follows from self-duality and is given [22, 23] by the relation $\beta_t = J/k_B T_t = \ln(1+\sqrt{q})$. The phase transition is continuous for a number of states $q \leq 4$, with additional logarithmic corrections at $q = 4$. For $q > 4$ the phase transition is discontinuous with a jump in the internal energy, i.e., a first-order phase transition. The distinctive features of first-order phase transitions are phase coexistence and metastability, which can be observed experimentally in the process of heating and cooling the system [24]. These effects are also pronounced in simulations with canonical Monte Carlo methods, leading to hysteresis of magnetization and energy profiles in a heating/cooling cycle [25]. The strength of the transition increases with q. The correlation length ξ does not diverge at the transition temperature T_t, i.e., it is finite, although the value can

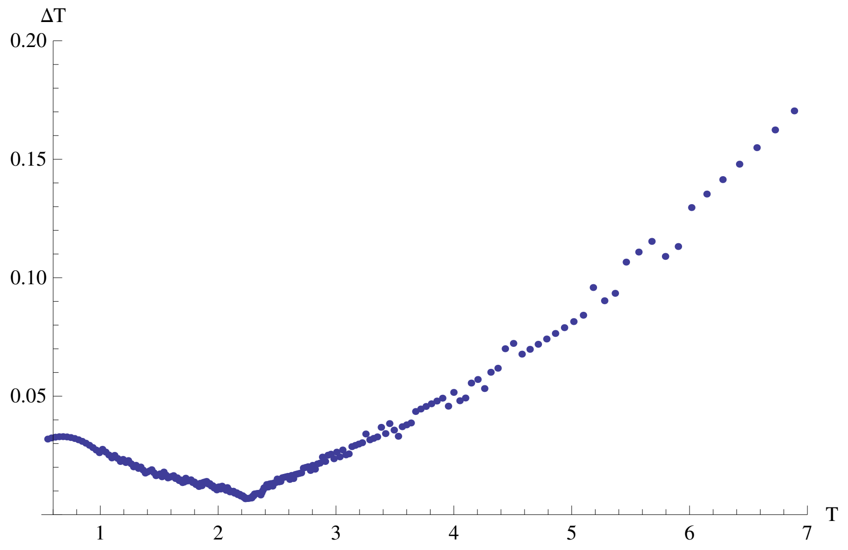

Fig. 1. Schedule of temperature steps for the adaptive temperature-step extension of the PA algorithm applied to the Ising model on an $L = 64$ square lattice with parameters $\theta = 100$, $R = 5000$, and overlap ≈ 0.85.

be extremely large depending on the number of states q. For $q = 20$ it is as small as 2.7 lattice spacings and for $q = 5$ states it is extremely large, $\xi \approx 2552$ [26].

Like standard canonical simulations, PA as described above also leads to hysteresis effects for the Potts model in the first-order regime and as such it is not immediately well suited for simulating systems undergoing first-order transitions. Using the inherent free-energy estimate provided by Eq. (3) leads to a possible way of determining the transition temperature, however. To this end, one extends the free-energy branches of the ordered and disordered phases and locates the transition at the point where the two curves cross [25]. For the case of the cooling temperature schedule, we start with the initial inverse temperature $\beta_0 = 0$ at which the partition function value is given by $Z(\beta_0) = q^N$, where N is the number of lattice sites. Additionally, we also use a heating temperature schedule, where the initial free energy at zero temperature $\beta \to \infty$ can be calculated as [27]

$$-\beta F_{\beta\to\infty} = \ln q - \beta E_0, \tag{7}$$

with the ground-state energy $E_0 = -2N$.

Figure 2 shows the resulting metastable free-energy branches calculated during cooling from infinite temperature and heating from zero temperature for the Potts model with a number of spin components of $q = 6$ and $q = 10$, respectively. The cooling and heating curves intersect close to the transition temperature, our results being compatible with the presence of small deviations of the order of $1/L^2$ as predicted in Ref. [25].

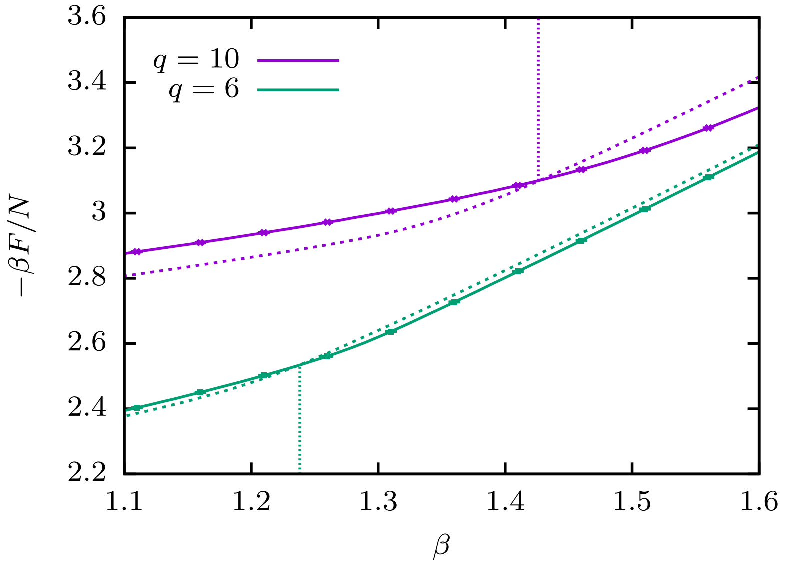

Fig. 2. Metastable free energy for the q-state Potts model with $q = 6$ (green) and $q = 10$ (magenta) for a cooling (solid lines) and a heating (dashed lines) cycle. The vertical dotted lines indicate the locations of the transition points β_t. The lattice size is $L = 32$, and the PA parameters are $\theta = 10$, $R = 10000$, and $\Delta\beta = 0.01$. Taken from Ref. [27].

5.3 Frustrated models and spin glasses

Similar to the parallel tempering heuristic, population annealing is an approach particularly suited for the simulation of systems with complex free-energy landscapes. A number of applications to frustrated and disordered systems have been reported in the literature. Population annealing has been successfully used for the relaxation to the ground state of a frustrated Ising antiferromagnet on the stacked triangular lattice with a ferromagnetic interlayer coupling [28]. Previous simulations using conventional, canonical Monte Carlo algorithms were not able to reach the ground state. More generally, the properties of PA for ground-state searches of spin systems were investigated in Ref. [29]. Apart from that, PA has been used in a series of simulational studies of spin glasses [30]. Among the questions addressed there is that of the number of thermodynamic states in the low-temperature phase. The results appear to be fully compatible with a single pair of pure states such as in the droplet/scaling picture. However, the results for whether or not domain walls induced by changing boundary conditions are space filling are also compatible with scenarios having many thermodynamic states, such as the chaotic pairs picture and the replica-symmetry breaking scheme. Substantially larger system sizes would be required to clearly determine whether domain walls induced by changing boundary conditions are space filling or not using average spin overlaps in windows or link overlaps.

5.4 Off-lattice systems

PA is not restricted to lattice spin systems, but it can be used for a range of off-lattice applications too. This was first demonstrated in Ref. [31] for simulations of a binary mixture of hard disks. While this system is often studied using molecular dynamics, it was here treated with PA Monte Carlo simulations, using a range of steps in density instead of in temperature. The authors determined the equation of state in the glassy region of a 50/50 mixture of hard spheres with the diameter ratio 1.4:1 and obtained precise results that are in reasonable agreement with previous simulations using parallel tempering [32].

Very recently, it was also demonstrated that PA can be combined with molecular dynamics simulations in place of the equilibrating sub-routine [33]. As the authors show for the benchmark example of the penta-peptide met-enkephalin, PA combined with molecular dynamics with a stochastic thermostat shows a performance similar to that of the more established parallel tempering method for this problem while providing far superior parallel scaling properties.

6 Conclusion

We have given a brief review of recent developments relating to the population annealing algorithm. The method is attractive especially since it is a highly parallel approach and efficient realizations can be quite easily developed for any known parallel architecture. In particular, we discussed here an implementation of the algorithm with CUDA, and demonstrated an acceleration of the simulations as compared with a conventional CPU realization by several orders of magnitude. The corresponding simulations work well for systems with continuous transitions such as the Ising model. For first-order transitions as present in the Potts model with a sufficiently large number of states, the approach suffers from metastability, but the natural free-energy estimate at least allows for a determination of the transition point by thermodynamic integration. Population annealing in the micro- or multicanonical ensembles promises to provide potential improvements in this respect. There is further scope for improvement in the implementation of the algorithm: A realization within the MPI-CUDA paradigm is currently being developed.

This work was partially supported by the grant 14-21-00158 from the Russian Science Foundation and by the Landau Institute for Theoretical Physics in the framework of the tasks from the Federal Agency of Scientific Organizations. The authors acknowledge support from the European Commission through the IRSES network DIONICOS under Contract No. PIRSES-GA-2013-612707.

References

1. A. Yu. Kitaev, Fault-tolerant quantum computation by anyons, Annals Phys. **303** 2–30 (2003)
2. Y. Iba, Population Monte Carlo algorithms, Trans. Jpn. Soc. Artif. Intell. **16** 279–286 (2001)
3. K. Hukushima and Y. Iba, Population annealing and its application to a spin glass, AIP Conf. Proc. **690** 200–206 (2003)
4. J. Machta, Population annealing with weighted averages: A Monte Carlo method for rough free-energy landscapes, Phys. Rev. E **82** 026704 (2010)
5. M. Weigel, Monte Carlo methods for massively parallel computers, in: Order, Disorder and Criticality, Vol. 5, ed. Yu. Holovatch (World Scientific, Singapore, 2018), pp. 271–340
6. L. Yu. Barash, M. Weigel, M. Borovský, W. Janke, and L. N. Shchur, GPU accelerated population annealing algorithm, Comp. Phys. Comm. **220** 341–350 (2017)
7. M. Weigel, L. Yu. Barash, L. N. Shchur, and W. Janke, in preparation
8. C. Amey and J. Machta, Analysis and optimization of population annealing, Phys. Rev. E **97** 033301 (2018)
9. A. M. Ferrenberg and R. H. Swendsen, Optimized Monte Carlo data analysis, Phys. Rev. Lett. **63** 1195–1198 (1989)
10. S. Kumar, D. Bouzida, R. H. Swendsen, P. A. Kollman, and J. M. Rosenberg, The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method, J. Comp. Chem. **13** 1011–1021 (1992)
11. S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, and P. A. Kollman, Mu1tidimensional free-energy calculations using the weighted histogram analysis method, J. Comp. Chem. **16** 1339–1350 (1995)
12. Code repository for the GPU accelerated PA algorithm is located at: https://github.com/LevBarash/PAising
13. M. Weigel, Performance potential for simulating spin models on GPU, J. Comput. Phys. **231** 3064–3082 (2012). T. Yavors'kii and M. Weigel, Optimized GPU simulation of continuous-spin glass models, Eur. Phys. J. Special Topics **210** 159–173 (2012)
14. M. McCool, J. Reinders, and A. Robison, Structured Parallel Programming: Patterns for Efficient Computation (Morgan Kaufman, Waltham, MA, 2012)
15. J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw, Parallel random numbers: As easy as 1, 2, 3, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11 (ACM, New York, NY, USA, 2011), article no. 16. M. Manssen, M. Weigel, and A. K. Hartmann, Random number generators for massively parallel simulations on GPU, Eur. Phys. J. Special Topics **210** 53–71 (2012)
16. L. Yu. Barash and L. N. Shchur, RNGSSELIB: Program library for random number generation, SSE2 realization, Comp. Phys. Comm. **182** 1518–1526 (2011)
17. L. Yu. Barash and L. N. Shchur, RNGSSELIB: Program library for random number generation. More generators, parallel streams of random numbers and Fortran compatibility, Comp. Phys. Comm. **184** 2367–2369 (2013)
18. M. S. Guskova, L. Yu. Barash, and L. N. Shchur, RNGAVXLIB: Program library for random number generation, AVX realization, Comp. Phys. Comm. **200** 402–405 (2016)
19. L. Yu. Barash and L. N. Shchur, PRAND: GPU accelerated parallel random number generation library: Using most reliable algorithms and applying parallelism of modern GPUs and CPUs, Comp. Phys. Comm. **185** 1343–1353 (2014)

20. H. A. Kramers and G. H. Wannier, Statistics of the two-dimensional ferromagnet. Part I, Phys. Rev. **60** 252–262 (1941)
21. L. Onsager, Crystal statistics. I. A two-dimensional model with an order-disorder transition, Phys. Rev. **65** 117–149 (1944)
22. R. J. Baxter, Potts model at the critical temperature, J. Phys. C: Solid State Phys. **6** L445–L448 (1973)
23. F. Y. Wu, The Potts model, Rev. Mod. Phys. **54** 235–268 (1982); *ibid.* **55** 315 (1983) (*Erratum*)
24. K. Binder and D. Heermann, Monte Carlo Simulation in Statistical Physics (Springer, Berlin, 2010)
25. W. Janke, First-order phase transitions, in: Computer Simulations of Surfaces and Interfaces, NATO Science Series, II. Mathematics, Physics and Chemistry – Vol. 114, eds. B. Dünweg, D. P. Landau, and A.I. Milchev (Kluwer, Dordrecht, 2003), pp. 111–135
26. C. Borgs and W. Janke, An explicit formula for the interface tension of the 2D Potts model, J. Physique I **2** 2011–2018 (1992)
27. L. Yu. Barash, M. Weigel, L. N. Shchur, and W. Janke, Exploring first-order phase transitions with population annealing, Eur. Phys. J. Special Topics **226** 595–604 (2017)
28. M. Borovský, M.Weigel, L. Yu. Barash, and M. Žukovič, GPU-accelerated population annealing algorithm: Frustrated Ising antiferromagnet on the stacked triangular lattice, EPJ Web of Conferences **108** 02016 (2016)
29. W. Wang, J. Machta, and H. G. Katzgraber, Comparing Monte Carlo methods for finding ground states of Ising spin glasses: Population annealing, simulated annealing, and parallel tempering, Phys. Rev. E **92** 013303 (2015)
30. W. Wang, J. Machta, and H. G. Katzgraber, Evidence against a mean-field description of short-range spin glasses revealed through thermal boundary conditions, Phys. Rev. B **90** 184412 (2014); Chaos in spin glasses revealed through thermal boundary conditions, Phys. Rev. B **92** 094410 (2015). W. Wang, J. Machta, H. Munoz-Bauza, and H. G. Katzgraber, Number of thermodynamic states in the three-dimensional Edwards-Anderson spin glass, Phys. Rev. B **96** 184417 (2017)
31. J. Callaham and J. Machta, Population annealing simulations of a binary hard-sphere mixture, Phys. Rev. E **95** 063315 (2017)
32. G. Odriozola and L. Berthier, Equilibrium equation of state of a hard sphere binary mixture at very large densities using replica exchange Monte Carlo simulations, J. Chem. Phys. **134** 054504 (2011)
33. H. Christiansen, M. Weigel, and W. Janke, Population annealing for molecular dynamics simulations of biopolymers, Preprint arXiv:1806.06016