Population annealing methods using hybrid parallel computing architecture

Lev Shchur
Landau Institute for Theoretical Physics RAS
and
Laboratory for Computational Physics
NRU Higher School of Economics

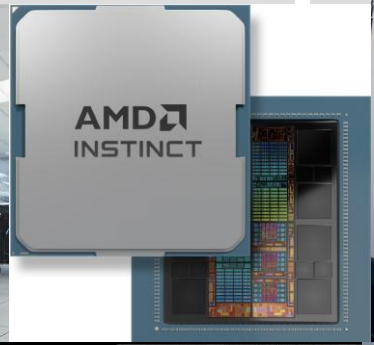
Grant RSCF 25-11-00158

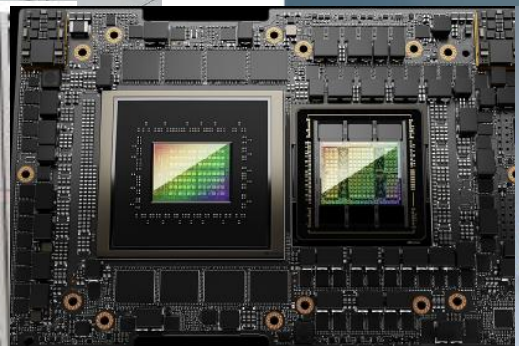
GRID2025, JINR, Dubna, 09.07.2025

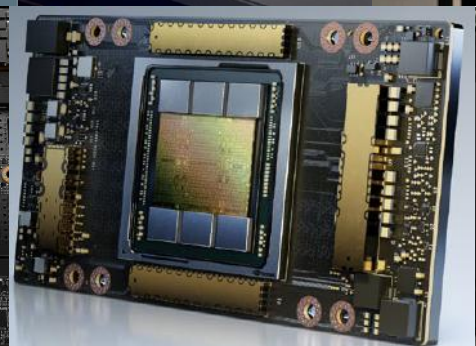

СС СР
АКАДЕМИЯ НАУК
ИНСТИТУТ
ТЕОРЕТИЧЕСКОЙ ФИЗИКИ
ИМЕНИ Л. Д. ЛАНДАУ

Population annealing methods using hybrid parallel computing architecture

- Modern computing systems are heterogeneous in terms of processor types
- They consist of a large number of nodes containing various types of computing elements







Population annealing methods using hybrid parallel computing architecture

- Most processor devices contain nested computing elements
- This is the path to further following Moore's Law

- Developing software capable of loading all computing elements with useful tasks is a rather non-trivial task

There are two options:

- Develop a universal, potentially fully scalable approach

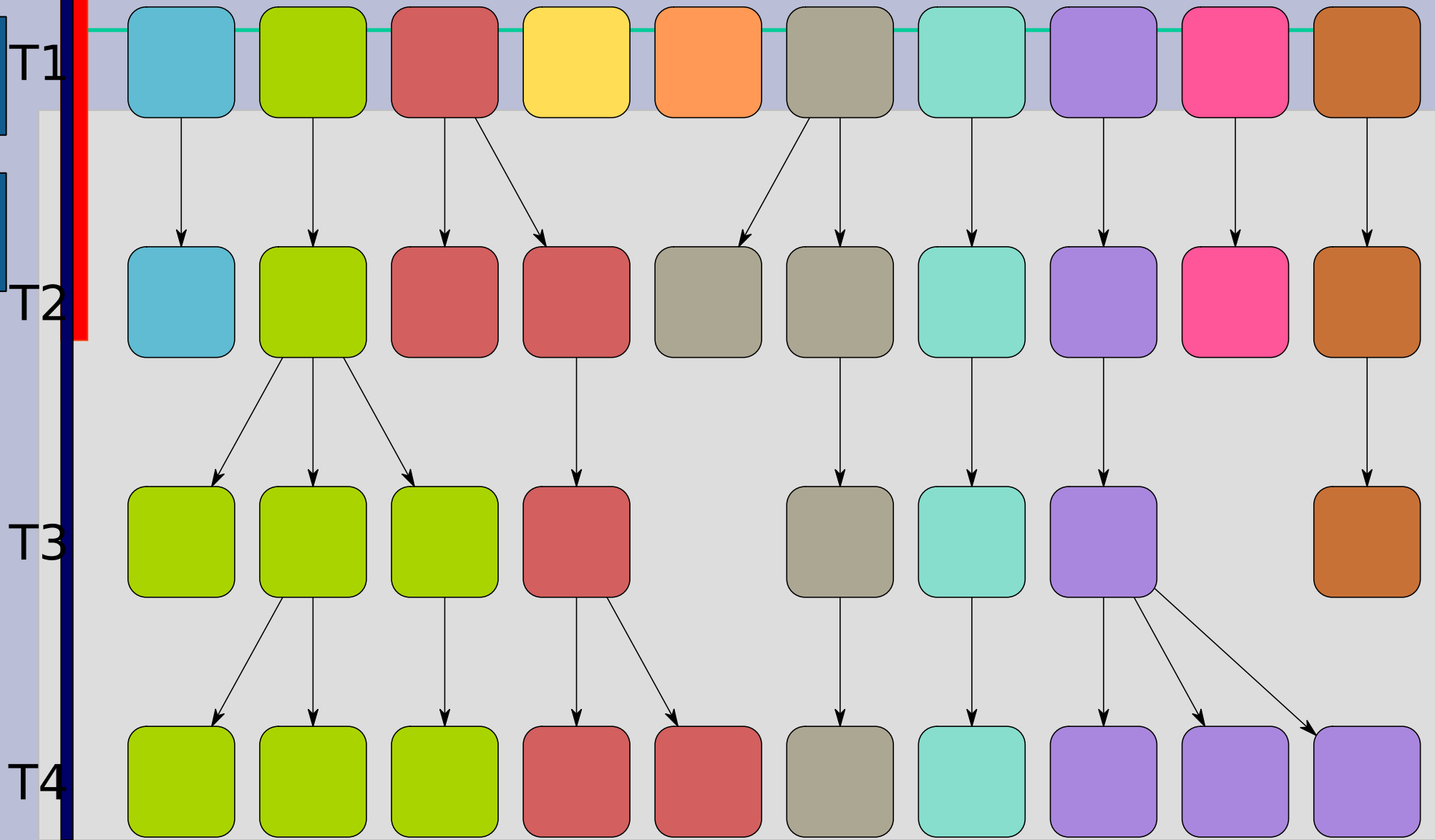
- Invent an algorithm that will be effective for a specific, albeit broad, set of tasks

Population annealing algorithm

1. The population annealing (PA) algorithm may be suitable for study systems with rough free energy landscapes (spin glasses, molecular dynamics, polymer folding, optimization problems, …).
2. PA combines the power of well-known efficient algorithms - simulated annealing, Boltzmann weighted differential reproduction, and sequential Monte Carlo processes.
3. Capable of bringing the replica population to equilibrium even in low temperature regions.
4. It provides an ideal assessment of free energy and entropy.
5. Enables efficient parallel implementation – multiple replicas.

Hukushima & Iba, AIP Conf. Proc. (2003) - idea
Machta, Phys. Rev. E. (2010) – CPU
Barash, Weigel, Borovsky, Janke & LS, Comp. Phys. Comm. (2017) – GPU
Russkov, Chulkevich & LS, Comp. Phys. Comm. (2021) – GPU/MPI
Weigel, Barash, Janke & LS, Phys. Rev. E (2021) – detailed analysis of PopAnn

Population annealing algorithm



GPU accelerated PA algorithm

(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`)

Barash, Weigel, Borovsky, Janke & LS, Comp Phys Comm 220 (2017) 341

Population Annealing algorithm

Partition function ratio $Q(\beta_k, \beta_{k-1})$ is given by

$$Q(\beta_k, \beta_{k-1}) = \frac{\sum_{j=1}^{\tilde{R}_{\beta_k}} \exp\left[-(\beta_{k-1} - \beta_k)E_j\right]}{\tilde{R}_{\beta_k}} \qquad (1)$$

and normalized weights $\tau_j(\beta_k, \beta_{k-1})$ calculated accordingly by

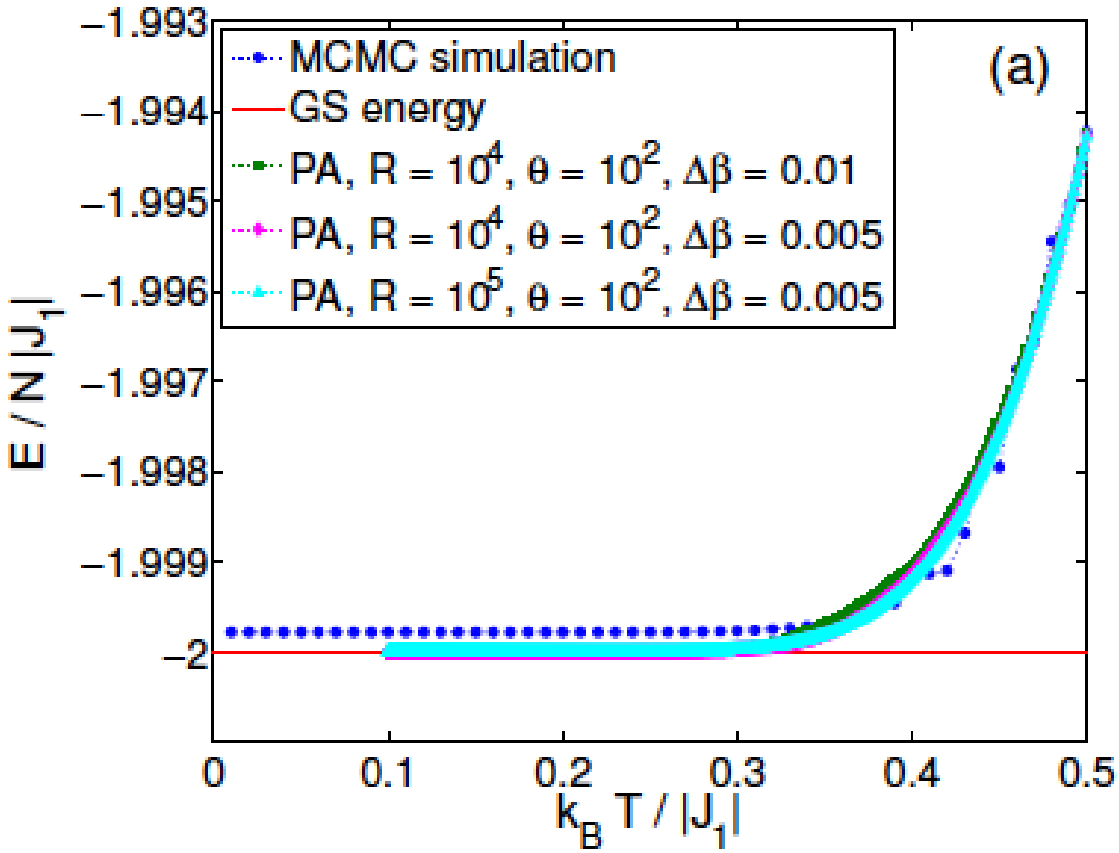
$$\tau_j(\beta_k, \beta_{k-1}) = \frac{\exp\left[-(\beta_{k-1} - \beta_k)E_j\right]}{Q(\beta_k, \beta_{k-1})} \qquad (2)$$

and

$$\tilde{R}_{\beta_k} = \sum_{j=1}^{\tilde{R}_{\beta_k}} \tau_j(\beta_k, \beta_{k-1}). \qquad (3)$$

$$-\beta_k \tilde{F}(\beta_k) = \sum_{i=K}^{k+1} \ln Q(\beta_k, \beta_{k-1}) + \ln \Omega$$

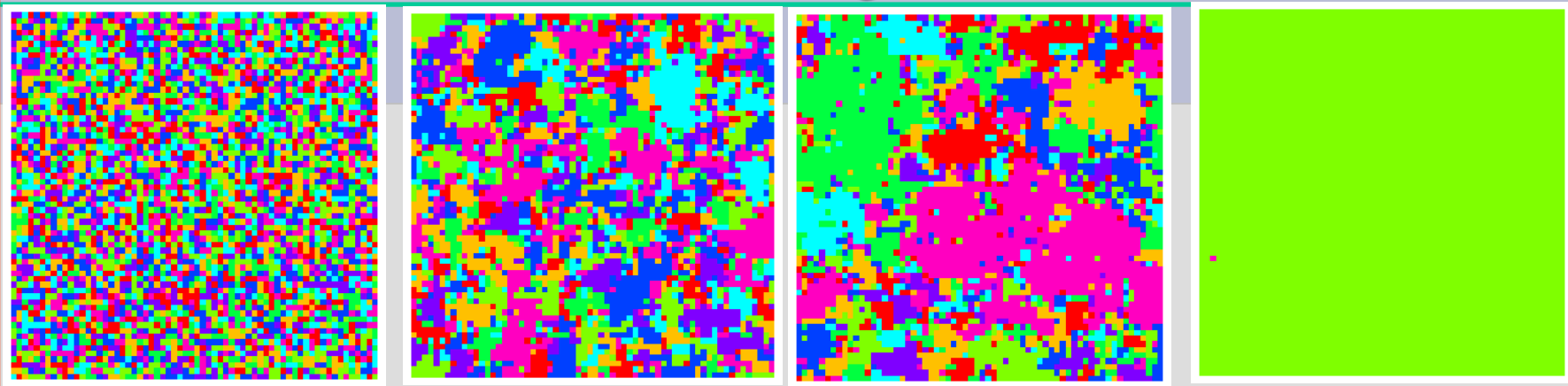
PopAnn, CUDA: frustrated Ising ferromagnet on the stacked triangular lattice



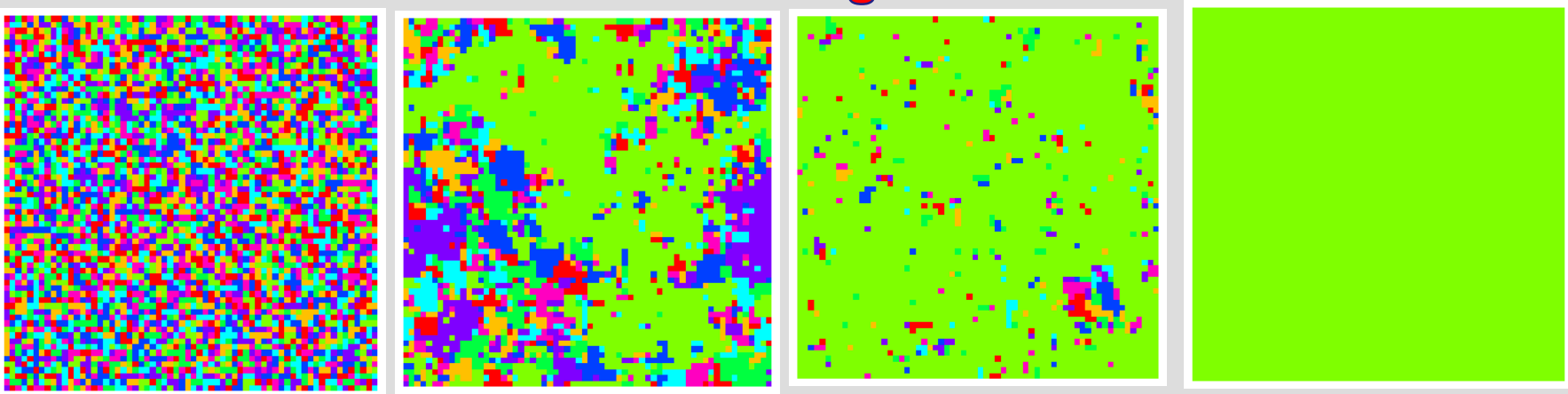
EPJ Web of Conferences **108**, 02016 (2016)

Potts model with Population Annealing

Cooling

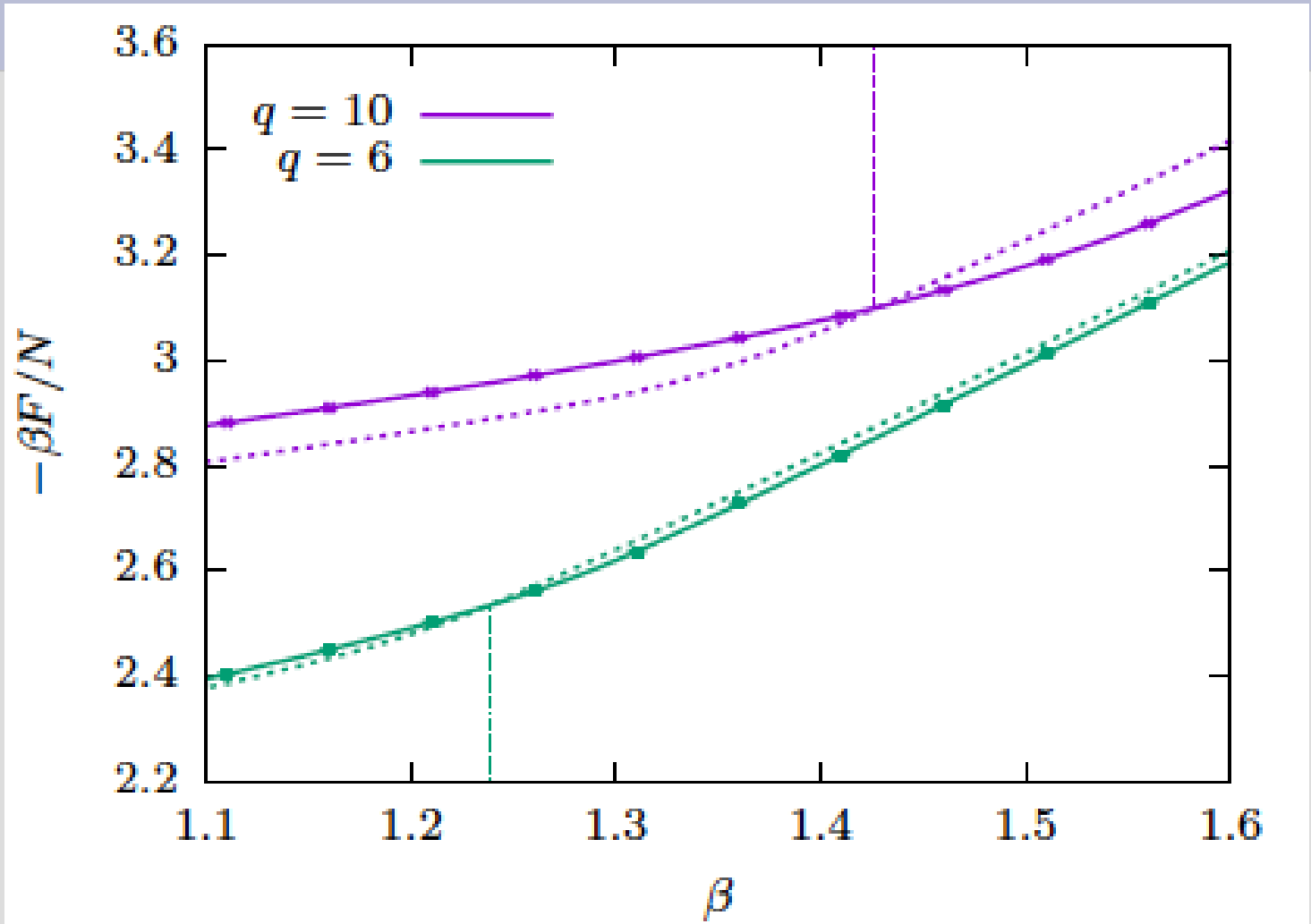


Heating

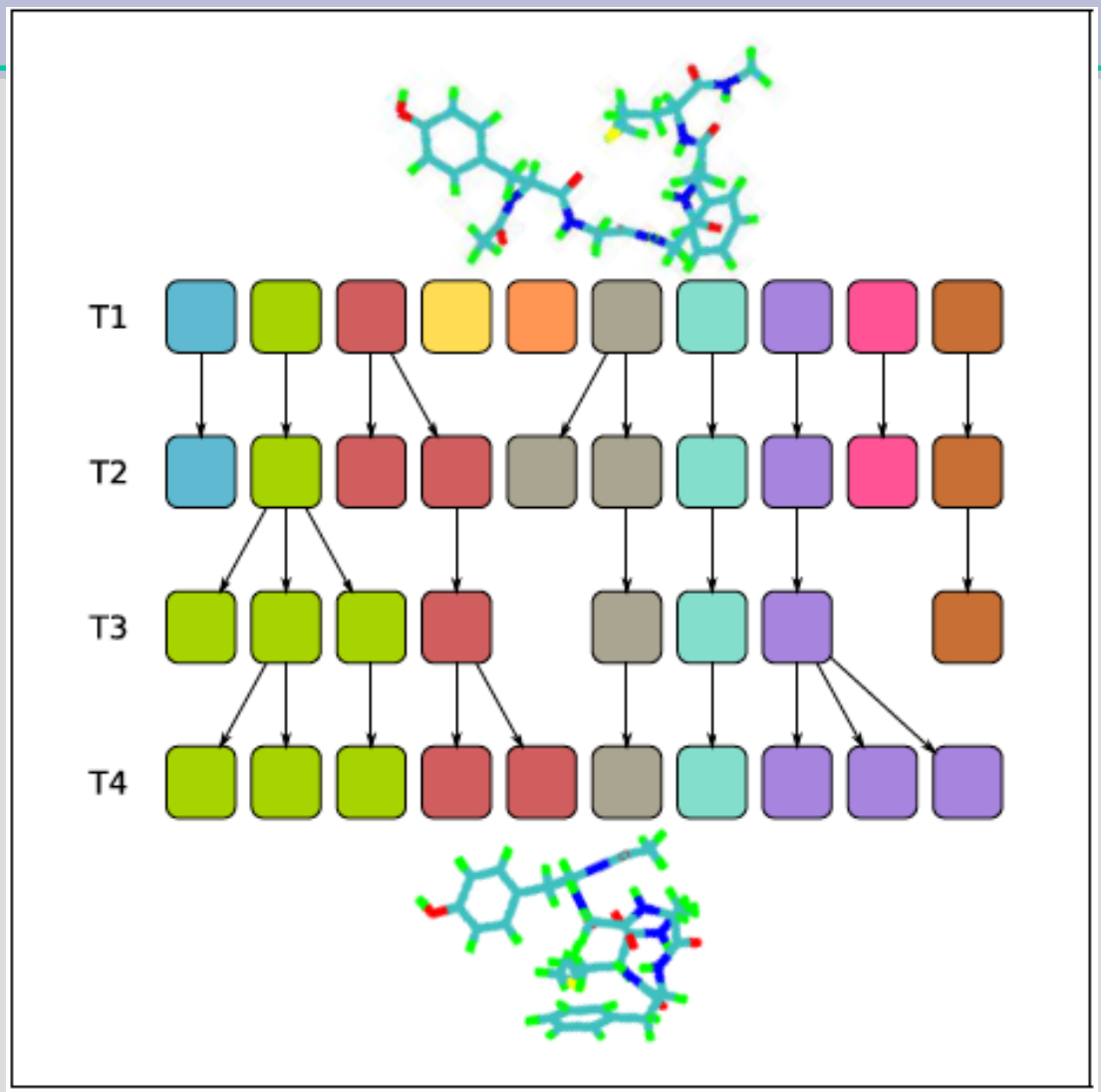


Barash, Weigel, Janke & LS, EPJ (2017)

Potts model with Population Annealing Hysteresis



Population Annealing - Simulations of Biopolymers



Population Annealing - Simulations of Biopolymers

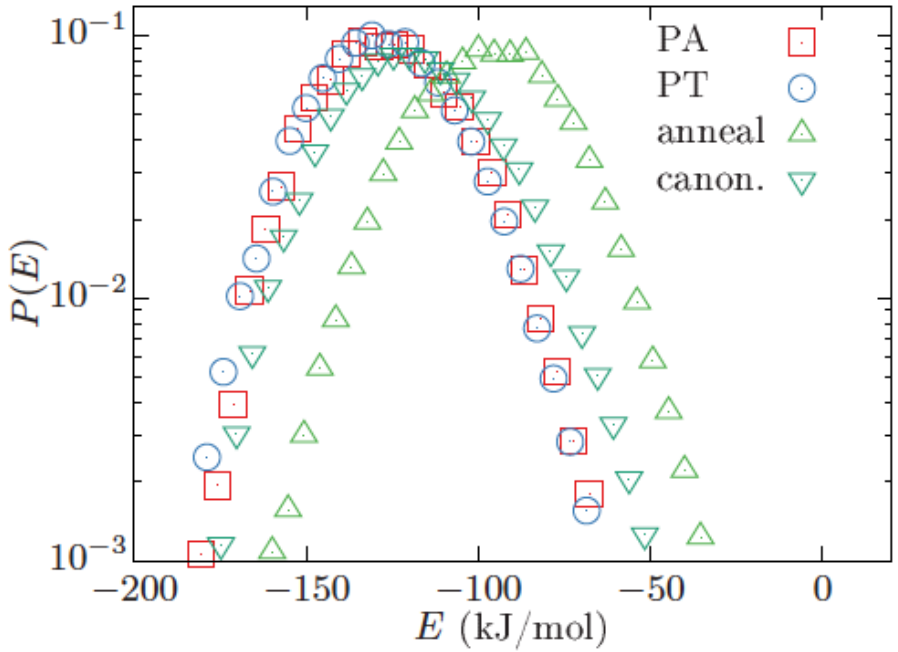
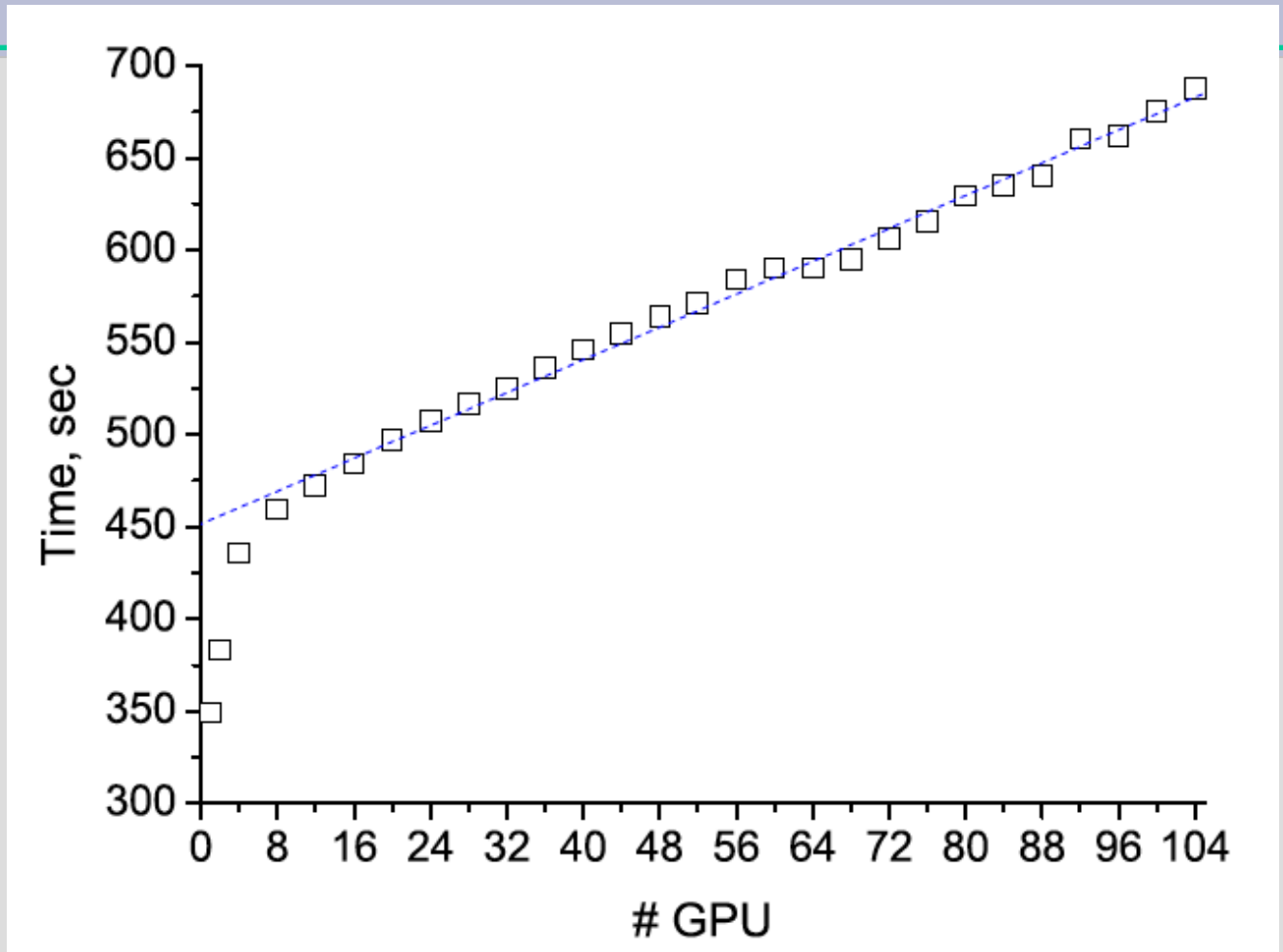


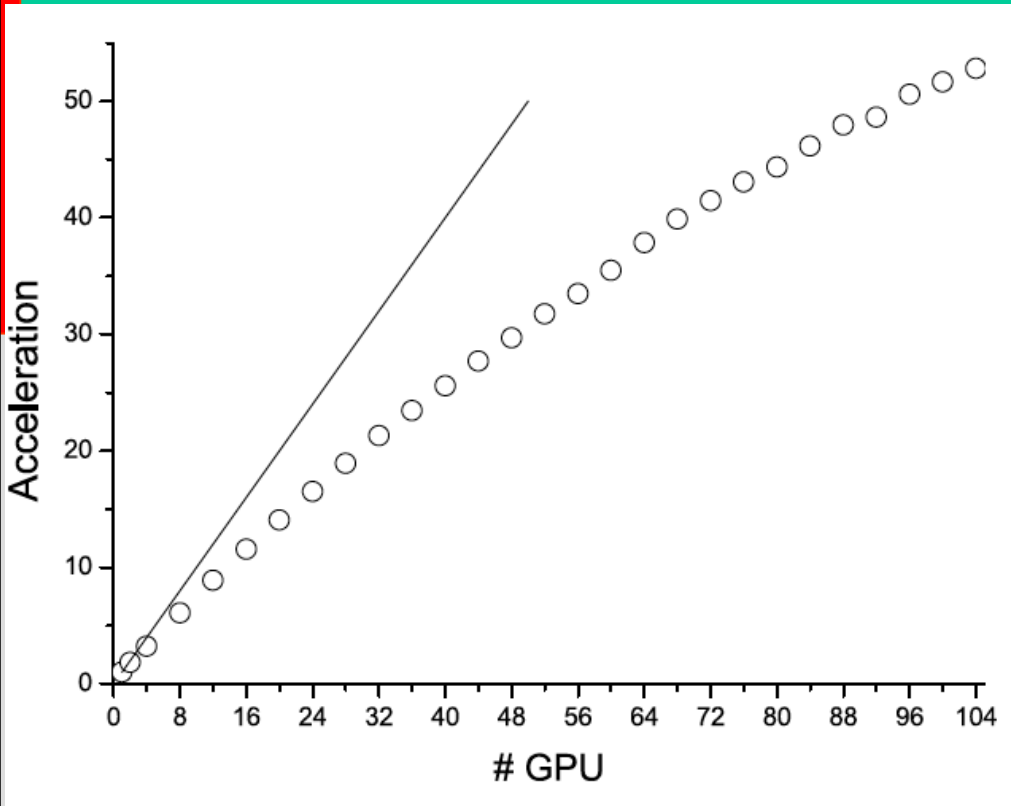
Figure 3: Energy histograms at the lowest temperature, $T = 200$ K, as obtained from the population annealing (PA), parallel tempering (PT), population annealing without resampling ("anneal"), and canonical ("canon.") simulations, respectively.

Simulations on the full-scale of cHARISMa (HSE)



26 nodes: 2x Intel Xeon Gold 6152 + 4x V100

Simulations on the full-scale of cHARISMa (HSE)



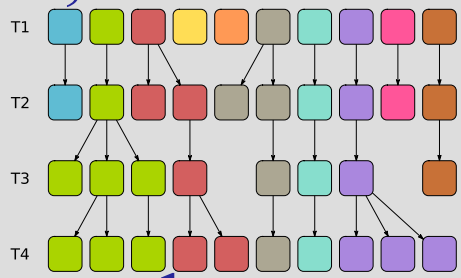
more than 2 million replicas in parallel

26 nodes: 2x Intel Xeon Gold 6152 + 4x V100

MPI/CUDA PA implementation

Technical problems solved:

- Load balancing of all GPU nodes with an approximately same number of replicas;
- Blocks of replicas = 2000;
- 10 blocks of replicas per GPU;



- Minimization of the extensive memory exchange between nodes;
- Approx. 2.5 million replicas in one run
  - technically impossible with the smaller clusters!

Microcanonical population annealing algorithm

- Initial configuration of a large number of replicas
- Setting the energy ceiling (floor)
- Transition from one energy level to another
- Monte Carlo at infinite temperature
- Calculating the proportion of replicas
  at the upper (lower) energy level
- Entropy calculation → Density of States → Functions of interest

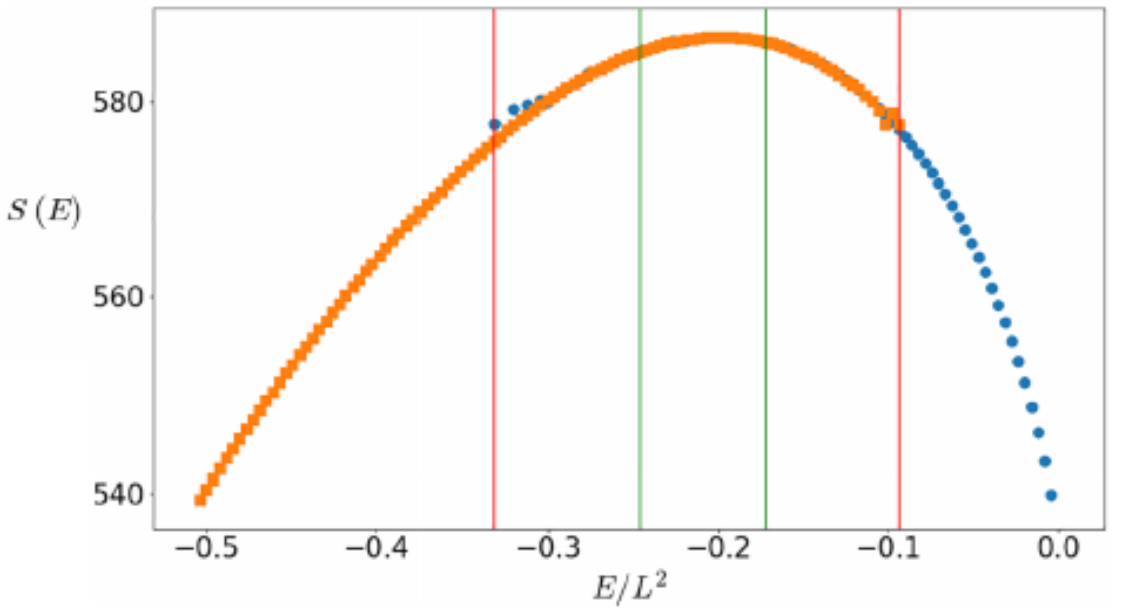
---

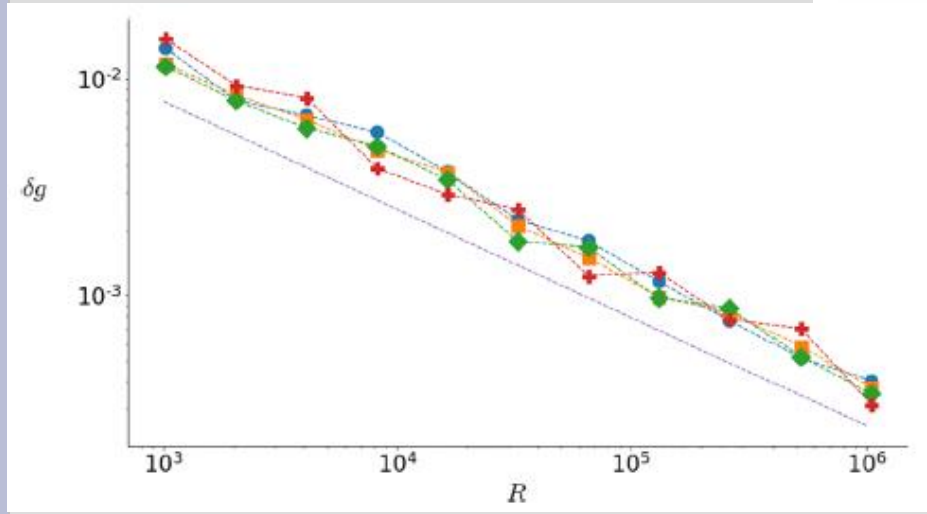
Rose & Machta, Phys. Rev. E. (2019) – GPU on the base of PA-2017
Mozolenko & LS, Phys. Rev. E. (2024) – modified GPU on the base of Rose-Machta
Mozolenko, Fadeeva & LS, Phys. Rev. E. (2024) – comparison with Wang-Landau
Sukhoverkhova, Mozolenko & LS, Phys. Rev. E. (2025) – combined with ML

Microcanonical population annealing algorithm

$$S^c(E) = \ln(\epsilon(E)) + \sum_{E'>E} \ln(1-\epsilon(E')),$$

$$S^f(E) = \ln(\epsilon(E)) + \sum_{E'<E} \ln(1-\epsilon(E')).$$





Mozolenko & LS, Phys. Rev. E. 109 (2024)

Microcanonical population annealing algorithm

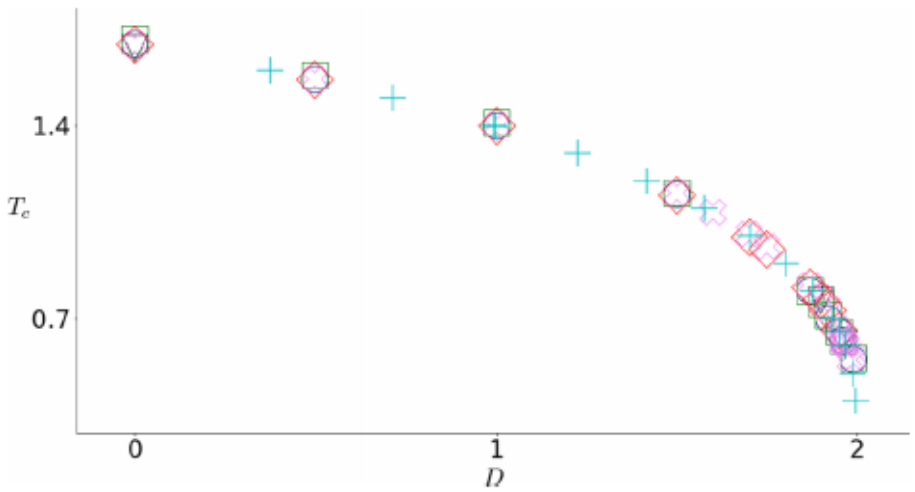


FIG. 1. Phase diagram obtained by different methods: transfer matrix [30] (blue circles), Monte Carlo [31] (black triangles), Wang-Landau [32] (green squares), high-energy and low-energy expansions [5,33] (red diamonds), microcanonical algorithm [34] (cyan pluses), and microcanonical population annealing (current work, violet crosses). The error bars are much smaller than the symbols.

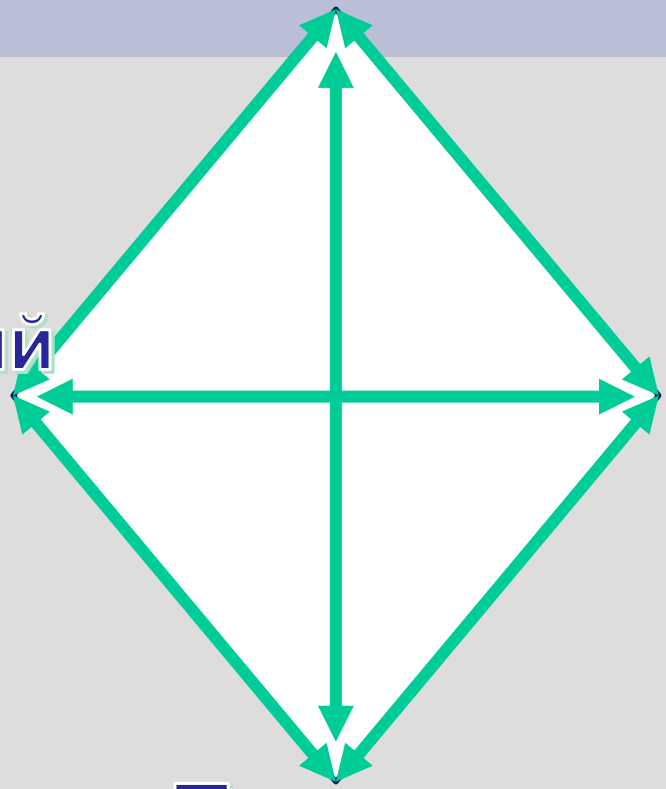
The Blume-Capel model [1,2] in the absence of a magnetic field is described by a Hamiltonian,

$$H = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j + \Delta \sum_i \sigma_i^2, \qquad (1)$$

Mozolenko & LS, Phys. Rev. E. (2024)

Эксперимент

Вычислительный эксперимент

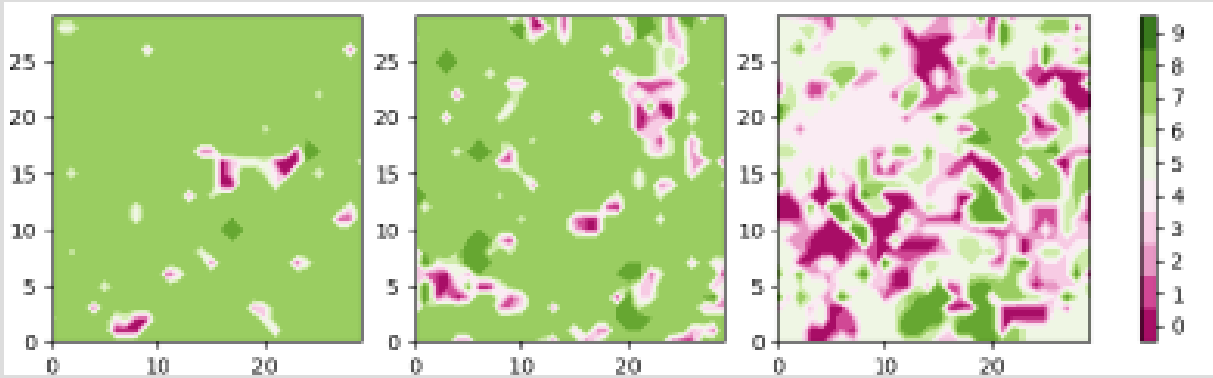


Машинное обучение

Теория

Microcanonical population annealing algorithm and ternary classification

- 2^{17} replicas by microcanonical PA
- 2^{13} uncorrelated replicas at each energy level
- Ternary classification with the value of disordered energy and ordered energy by supervised learning
- Probability of ordered phase, mixed phase, and disordered phase



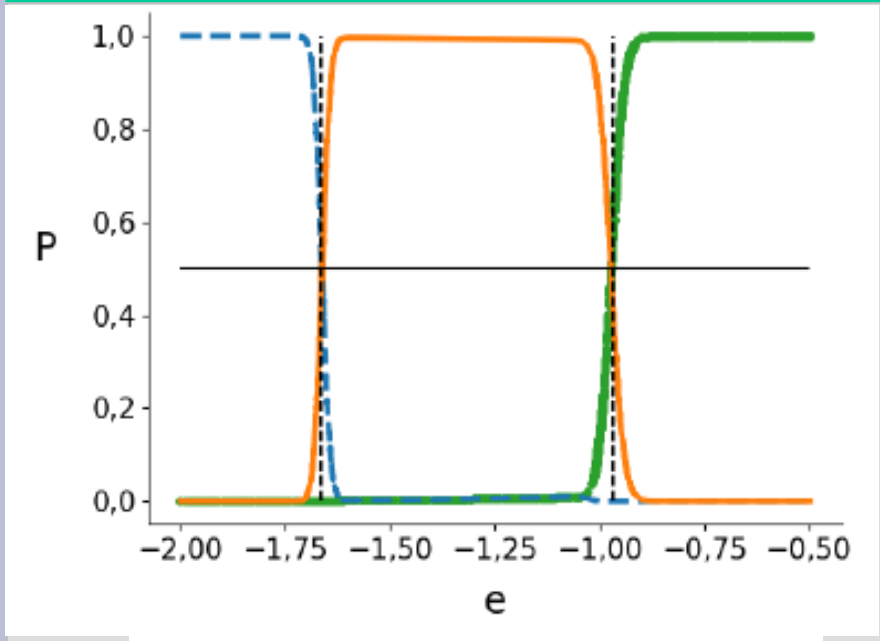
OS if $e < e_o$;

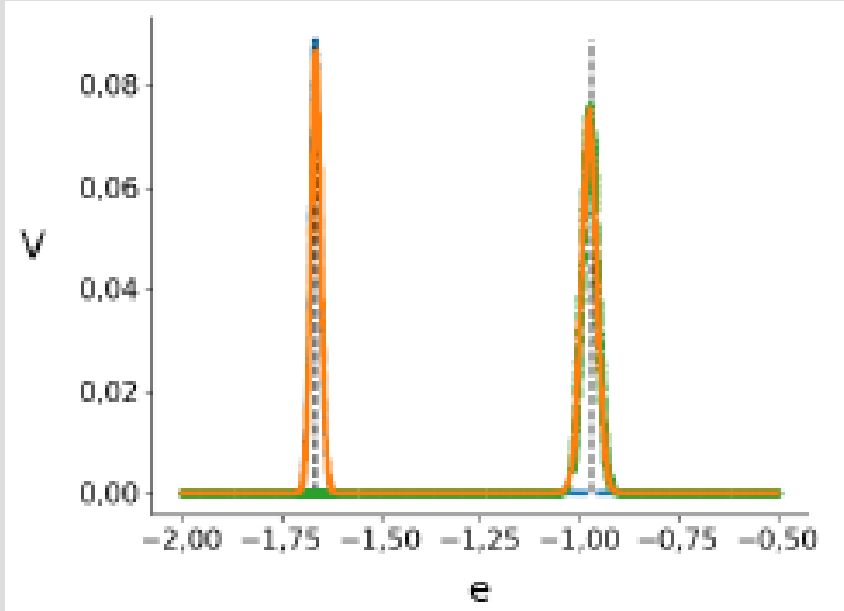
CS if $e_o < e < e_d$

DS if $e > e_d$.

Sukhoverkhova, Mozolenko & LS, Phase probabilities in first-order phase transitions using machine learning and multicanonical population annealing, Phys. Rev. E. (2025)

Microcanonical population annealing algorithm and ternary classification



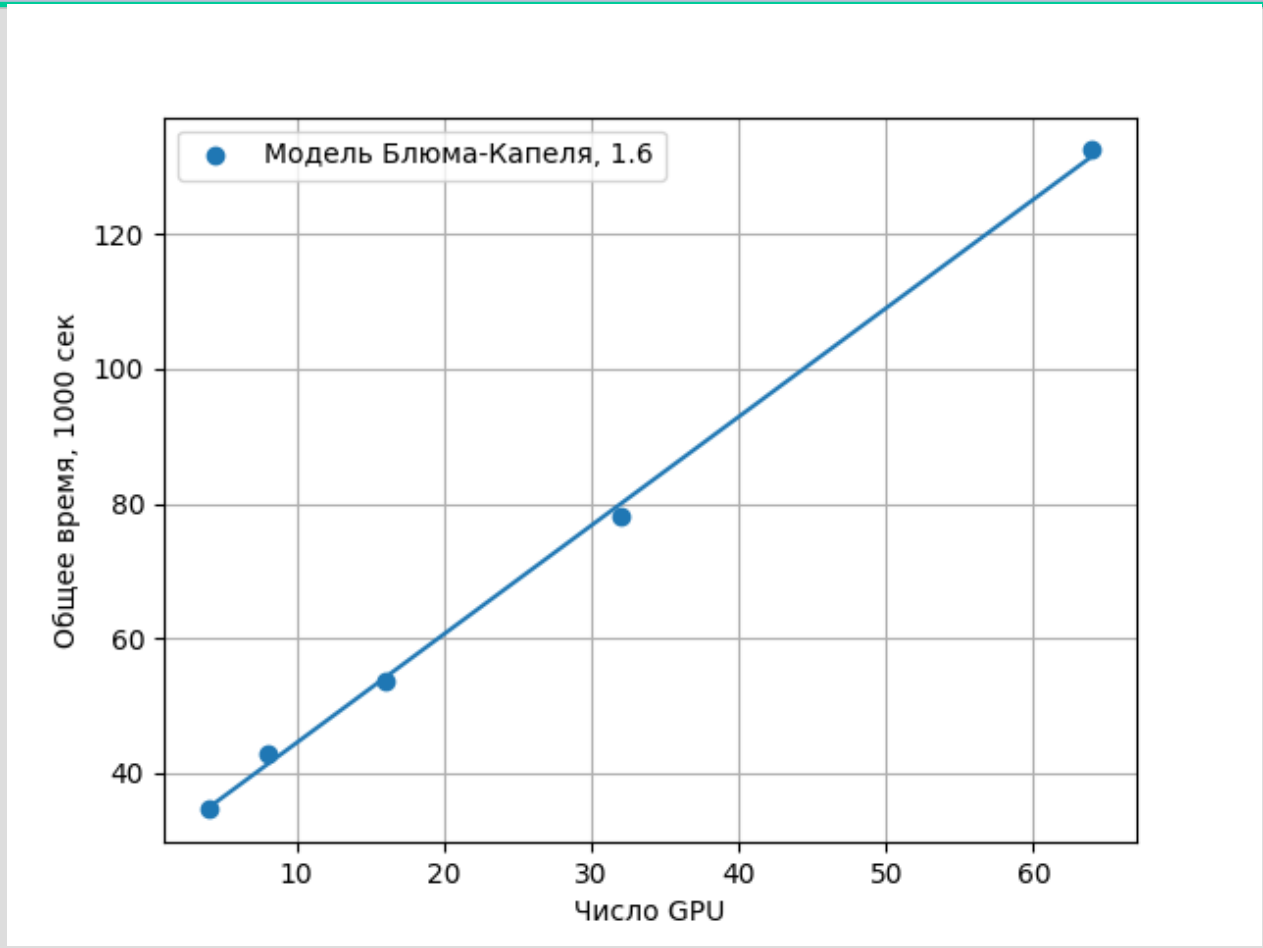


$$P_{xS}(e) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} p^i_{xS}(e)$$

$$V_{xS} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left[p^i_{xS}(e)\right]^2 - \left[\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} p^i_{xS}(e)\right]^2$$

Sukhoverkhova, Mozolenko & LS, Phase probabilities in first-order phase transitions using machine learning and multicanonical population annealing, Phys. Rev. E. (2025)

Microcanonical population annealing algorithm CUDA+MPI



Russkov, XIX Int. Conf. Supercomputing and Mathematical Modeling, Sarov (2024)

Conclusion

- The *population annealing* approach is a promising tool for solving a number of problems:

• "complex" ground state,
• rough energy landscape,
• optimization problems.

- The *Population annealing* algorithm and the *Microcanonical Population annealing* algorithm are natural candidates for large-scale and *fully scalable simulations* with *heterogeneous parallelism*.

Grant RSCF 25-11-00158