# Population annealing and large scale simulations in statistical mechanics

Lev Shchur
Landau Institute for Theoretical Physics,
Science Center in Chernogolovka,
and
NRU Higher School of Economics

DIONICOS

RSF | Russian Science Foundation

RuSCDays-2018, Sokolniki, 2018-09-24

# Population annealing and large scale simulations in statistical mechanics

in collaboration with

**Lev Barash**, Landau Institute, Russia
**Wolfhard Janke**, Leipzig University, Germany
**Martin Weigel**, Coventry University, United Kingdom

RSCF 14-21-11058 and DIONICOS

DIONICOS

RSF Russian Science Foundation

**RuSCDays-2018, Sokolniki, 2018-09-24**

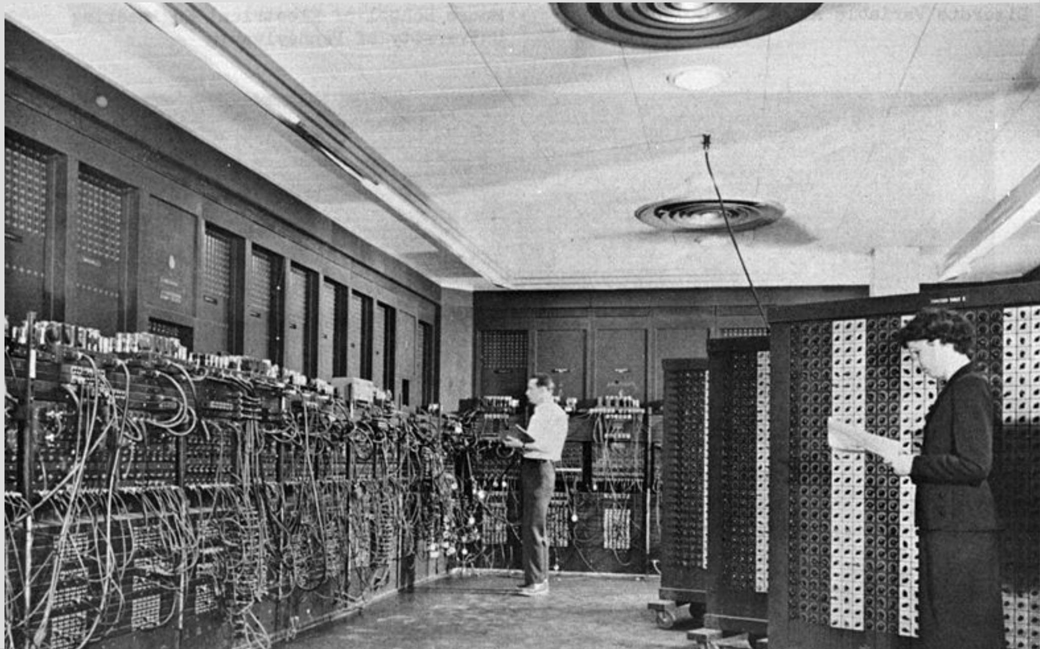
# ЭВМ and Supercomputers

**XX century - fall of 40th – beginning of 50th:**

- nuclear project (hudrodynamics)
- Monte Carlo method - von Neumann, Ulam, Metropolis-Rosenblut$^2$-Teller$^2$
- Fermi-Pasta-Ulam problem
- Random Number Problem – von Neumann, Lehmer

МЭСМ, 1950

ENIAC, 1944



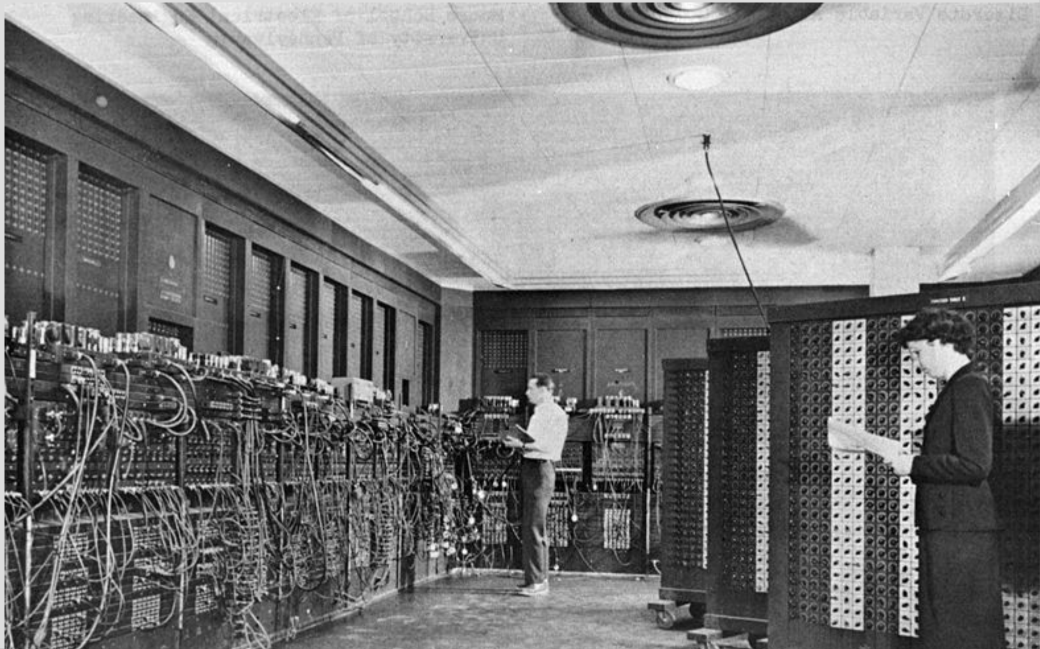
# ЭВМ and Supercomputers

**XX century - fall of 40th – beginning of 50th:**

- nuclear project (hudrodynamics)
- Monte Carlo method - von Neumann, Ulam,
  Metropolis-Rosenblut$^2$-Teller$^2$
- Fermi-Pasta-Ulam problem
- Random Number Problem – von Neumann, Lehmer
- ***Lehmer talk at Symposium*** **on *Large-Scale Digital Calculating***

**ENIAC, 1944**

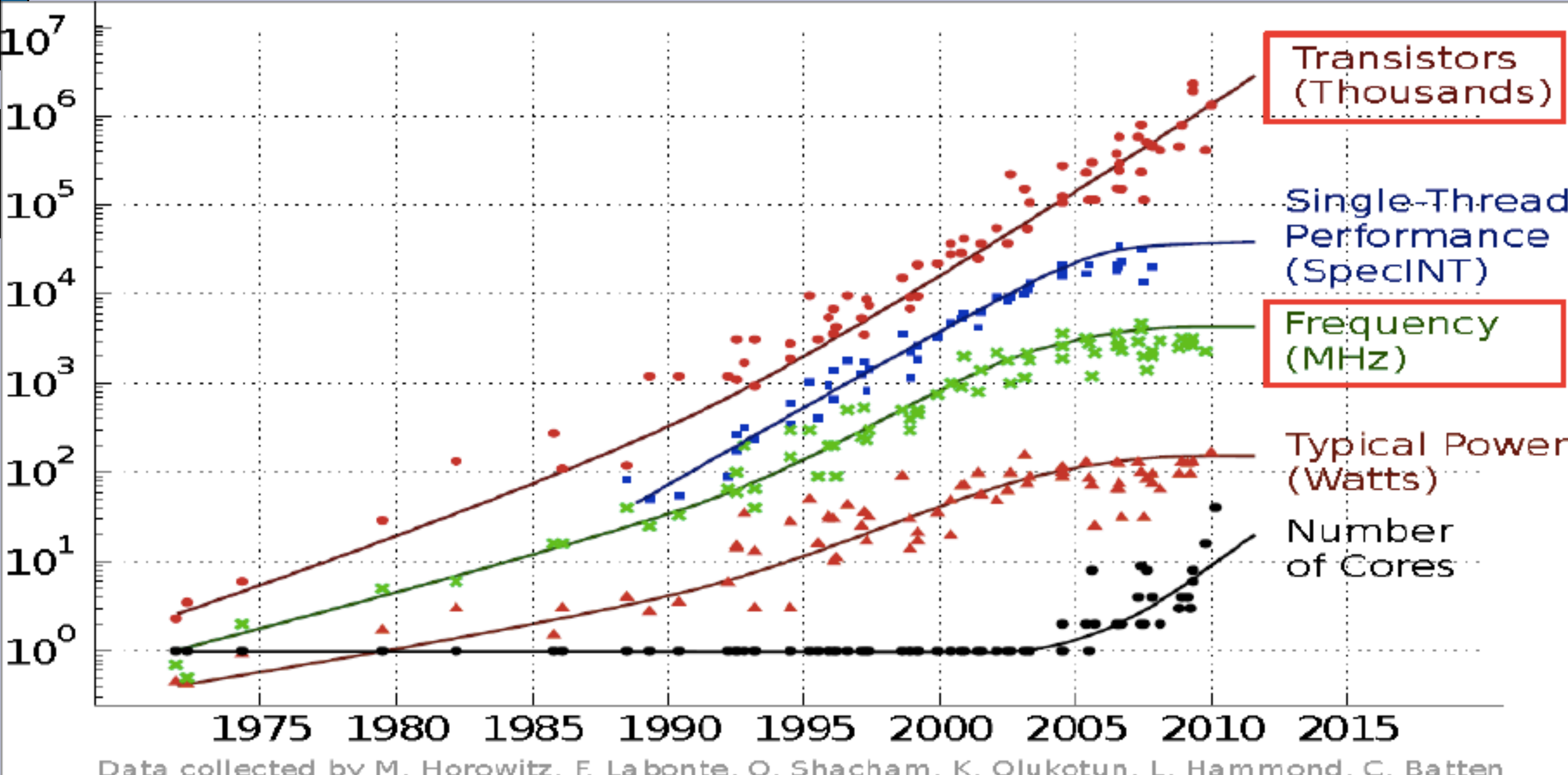
**МЭСМ, 1950**



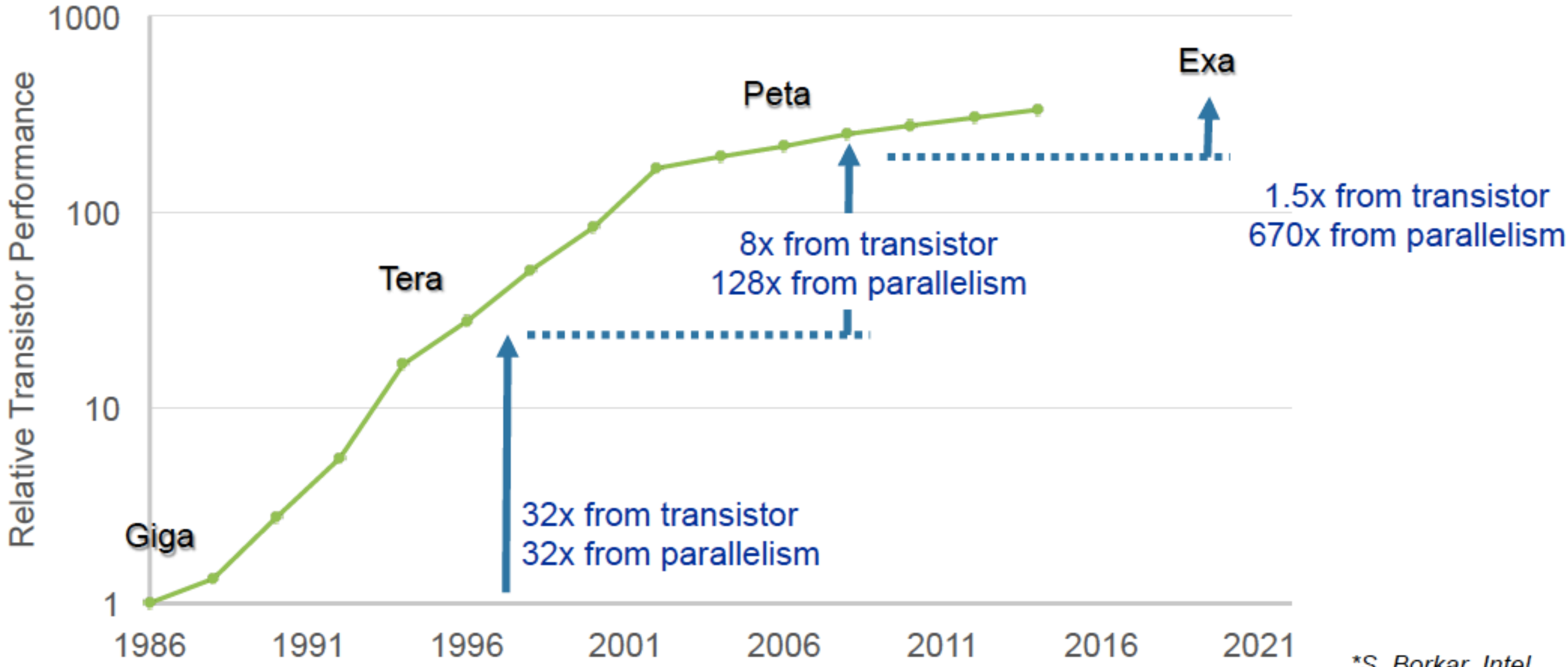
# Population annealing and large scale simulations in statistical mechanics

- *Motivation*
- *Population annealing*
- *Results*
- *Examples*
- *Implementation*
- *Arguments*
- *Conclusion*

# Moore's law – saturation?
# Or something new happens?



# From Giga to Exa, via Tera & Peta*




*S. Borkar, Intel

**Performance from parallelism**

ECP EXASCALE COMPUTING PROJECT



From J. Messina, 31 July, 2016 (ECP director, ANL)

# Main goal

The big challenge for scientific computing is to develop algorithms and computational frameworks that use the parallel hardware efficiently.

There are two approaches:

- Develop a universal framework which is potentially is fully scalable;
- Invent an algorithm which is efficient for the particular set of problems.

# Examples

*There are two approaches:*

- *Develop a universal framework which is potentially is fully scalable;*

For example, Parallel Discrete Event Simulation

(talk by Lilia Ziganurova on Wednesday at CSP2018, 14.45-15.10, Strogino)

- *Invent an algorithm which is efficient for the particular set of problems*

For example, Population Annealing

(this talk).

# Population Annealing algorithm

1. The population annealing algorithm is designed for study systems with rough free-energy landscapes (spin glasses, optimization problems…).
2. Combines the power of the known efficient algorithms - simulated annealing, Boltzmann weighted differential reproduction, and sequential Monte Carlo process.
3. Bring the population of replicas to the equilibrium even in the low-temperature region.
4. Provides a very good estimate of the free energy.
5. PopAnn is performed over a large number of replicas with many spin updates, and it is a good candidate for massive parallelism.
6. Efficient parallel implementation – many replicas.

K. Hukushima, Y. Iba, AIP Conf. Proc. **690** 200 (2003)
J. Machta, Phys. Rev. E **82** 026704 (2010)

# Population Annealing algorithm by Machta (PAM)

1. Set up an equilibrium ensemble of $R = R_0$ independent copies (replicas) of the system at inverse temperature $\beta_0$. Often $\beta_0 = 0$, where this can be easily achieved.
2. To create an approximately equilibrated sample at $\beta_i > \beta_{i-1}$, resample configurations with their relative Boltzmann weight $\tau_i(E_j) = \exp[-(\beta_i - \beta_{i-1})E_j]/Q_i$, where $Q_i = \sum_j \exp[-(\beta_i - \beta_{i-1})E_j]/R_{i-1}$.
3. Update each replica by $\theta$ rounds of an MCMC algorithm at inverse temperature $\beta_i$.
4. Calculate estimates for observable quantities $\mathcal{O}$ as population averages $\sum_j \mathcal{O}_j/R_i$.
5. Goto step 2 unless the target temperature $\beta_{K-1}$ has been reached.

J. Machta, Phys. Rev. E **82** 026704 (2010)

# Population Annealing algorithm by Machta (PAM)

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

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

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

# Potts model with Population Annealing

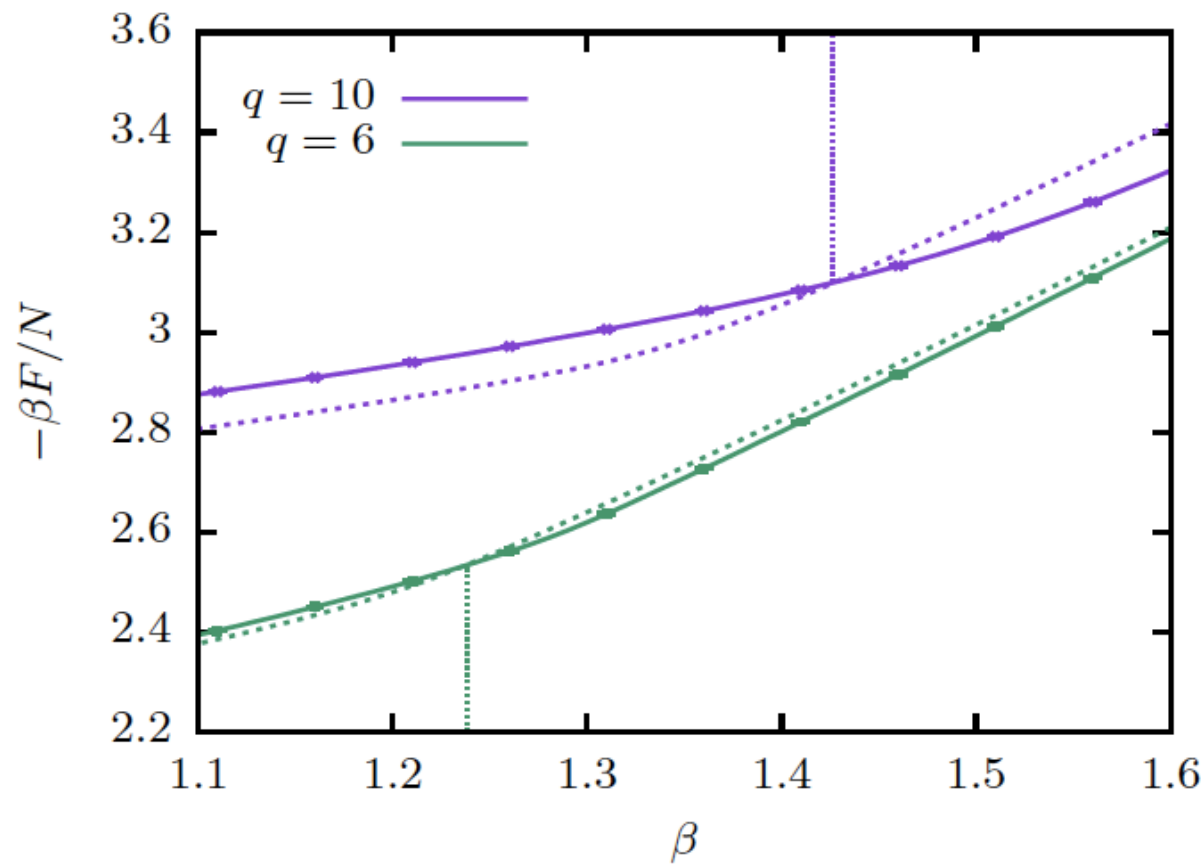
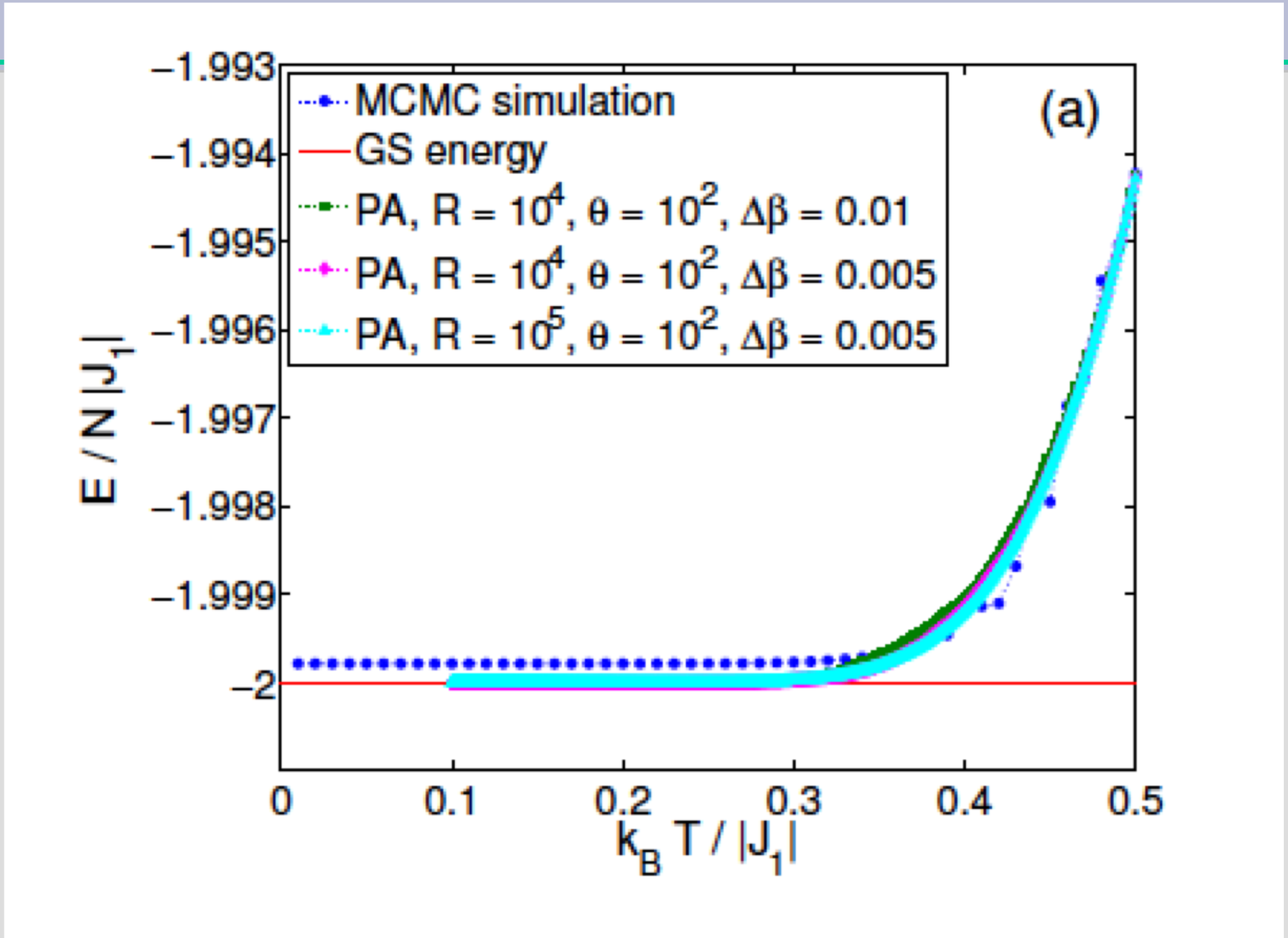



**Fig. 2.** Metastable free energy for the $q$-state Potts model with $q = 6$ (green) and $q = 10$ (magenta) for a cooling (solid lines) and a heating (dashed lines) cycle. The vertical dotted lines indicate the locations of the transition points $\beta_t$. The lattice size is $L = 32$, and the PA parameters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

Eur. Phys. J. Special Topics **226** 595–604 (2017)

# PopAnn: frustrated Ising ferromagnet on the stacked triangular lattice




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

# Ising model with Population Annealing

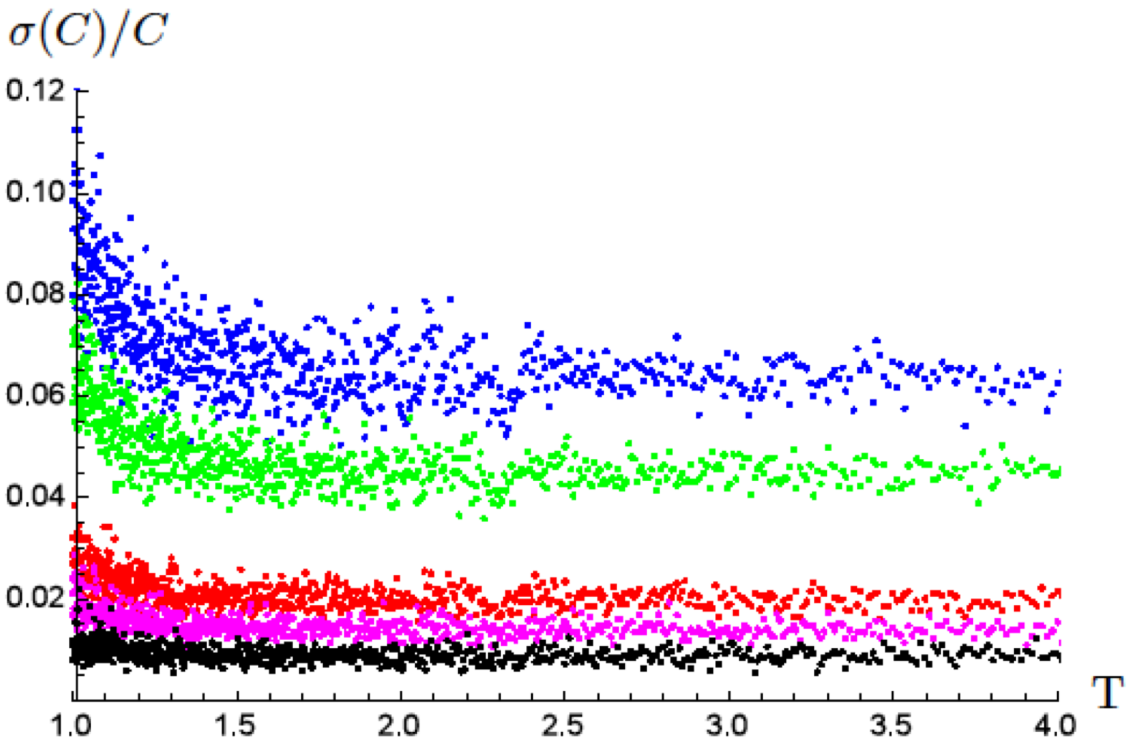
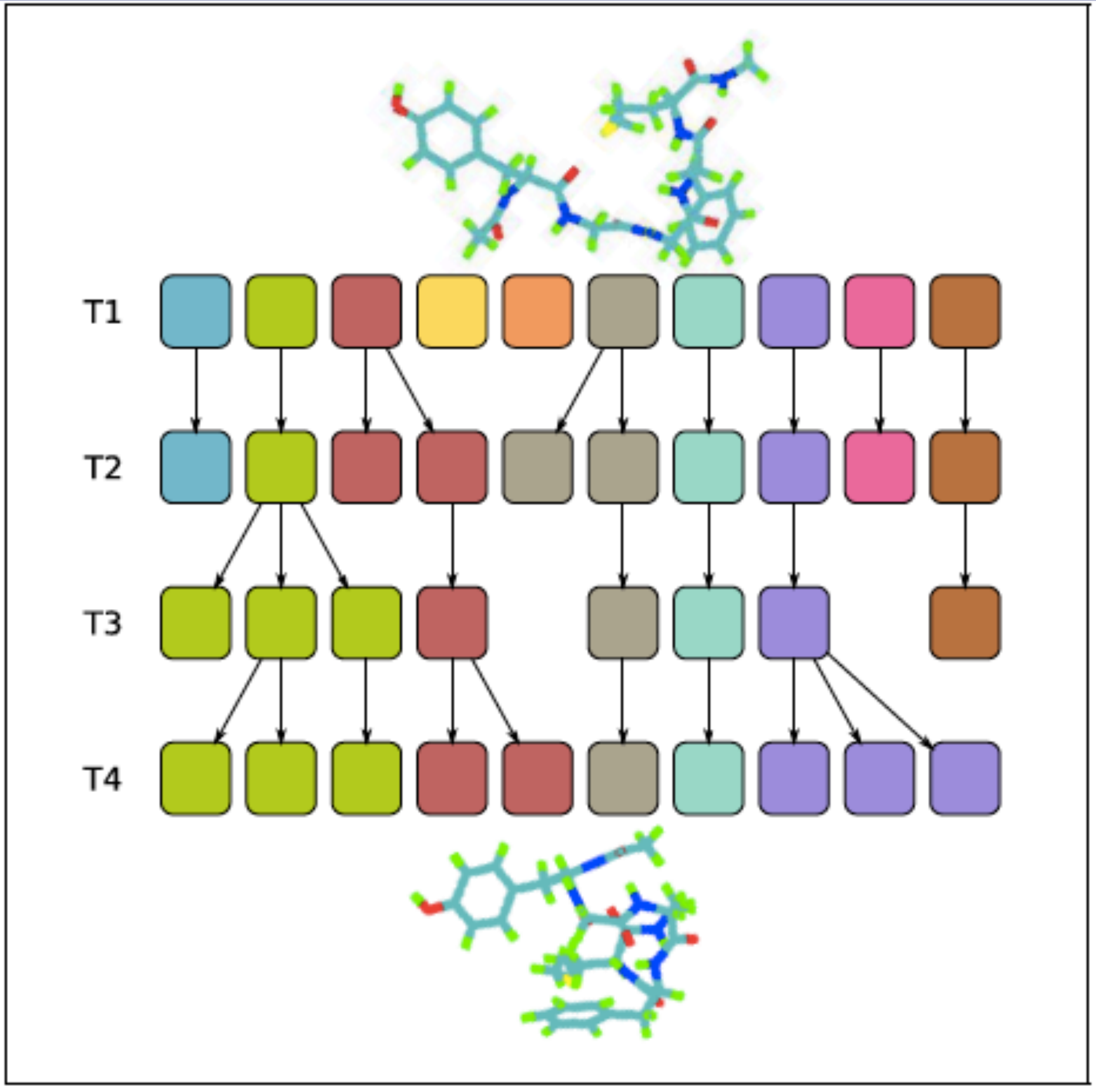



FIG. 5: Relative specific heat errors, $\sigma(C(T))/C(T)$, as a function of the temperature for a different number of replicas $R$, $R = 500$ - blue, $R = 1000$ - green, $R = 5000$ - red, $R = 10000$ - pink, and $R = 25000$ - black.

Computer Physics Communications (2017)

# Population Annealing for Molecular Dynamics Simulations of Biopolymers



# Population Annealing for Molecular Dynamics Simulations of Biopolymers

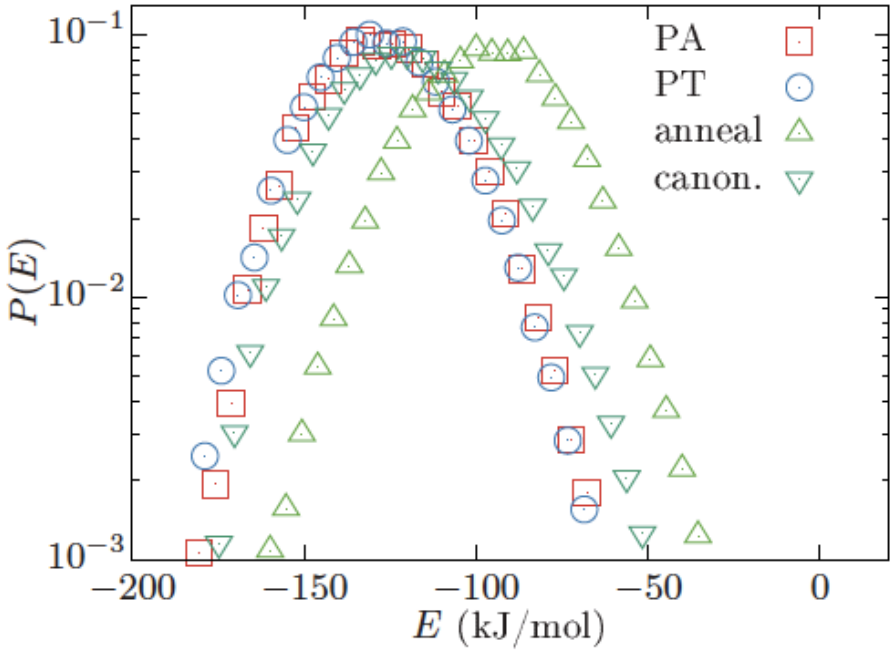



Figure 3: Energy histograms at the lowest temperature, $T = 200$ K, as obtained from the population annealing (PA), parallel tempering (PT), population annealing without resampling ("anneal"), and canonical ("canon.") simulations, respectively.

Computer Physics Communications 220 (2017) 341–350

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

# 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`)

Computer Physics Communications 220 (2017) 341–350

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

# GPU accelerated PA algorithm

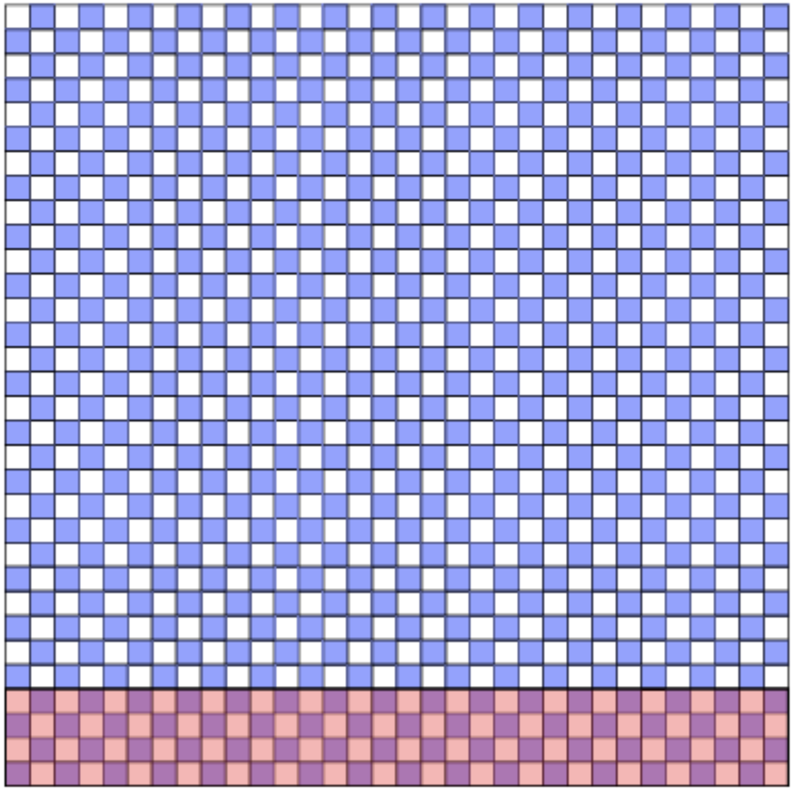


**Fig. 1.** Diagrammatic representation of the mapping of thread blocks to spins in the updating kernel. The code works with thread blocks of size EQthreads. Each block works on a single replica of the population, using its threads to update tiles of size $2 \times$ EQthreads spins. To this end, it flips 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.

**Table 3**
Times $t_{SF}$ per spin flip (in ns) for SSC and MSC GPU codes run on the Tesla K80 GPU for a $L = 128$ system.

| | | *R* | | | |
|---|---|---|---|---|---|
| | | 2 000 | 10 000 | 50 000 | 100 000 |
| | | Single-spin coding (SSC) | | | |
| | 1 | 0.229 | 0.213 | 0.213 | 0.213 |
| | 5 | 0.123 | 0.119 | 0.119 | 0.120 |
| | 10 | 0.111 | 0.108 | 0.107 | 0.109 |
| $\theta$ | 50 | 0.101 | 0.0994 | 0.0985 | 0.0987 |
| | 100 | 0.0998 | 0.0976 | 0.0977 | 0.0975 |
| | 200 | 0.0997 | 0.0972 | 0.0970 | 0.0970 |
| | 500 | 0.0991 | 0.0971 | 0.0969 | 0.0969 |
| | | Multi-spin coding (MSC) | | | |
| | 1 | 0.2504 | 0.1439 | 0.1440 | 0.1543 |
| | 5 | 0.0596 | 0.0372 | 0.0341 | 0.0341 |
| | 10 | 0.0359 | 0.0240 | 0.0232 | 0.0234 |
| $\theta$ | 50 | 0.0168 | 0.0136 | 0.0123 | 0.0121 |
| | 100 | 0.0144 | 0.0123 | 0.0110 | 0.0108 |
| | 200 | 0.0132 | 0.0119 | 0.0103 | 0.0101 |
| | 500 | 0.0125 | 0.0118 | 0.0098 | 0.0097 |

# GPU accelerated PA algorithm

**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 $\theta$, 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 |

# Random Number Generation at "large scale"

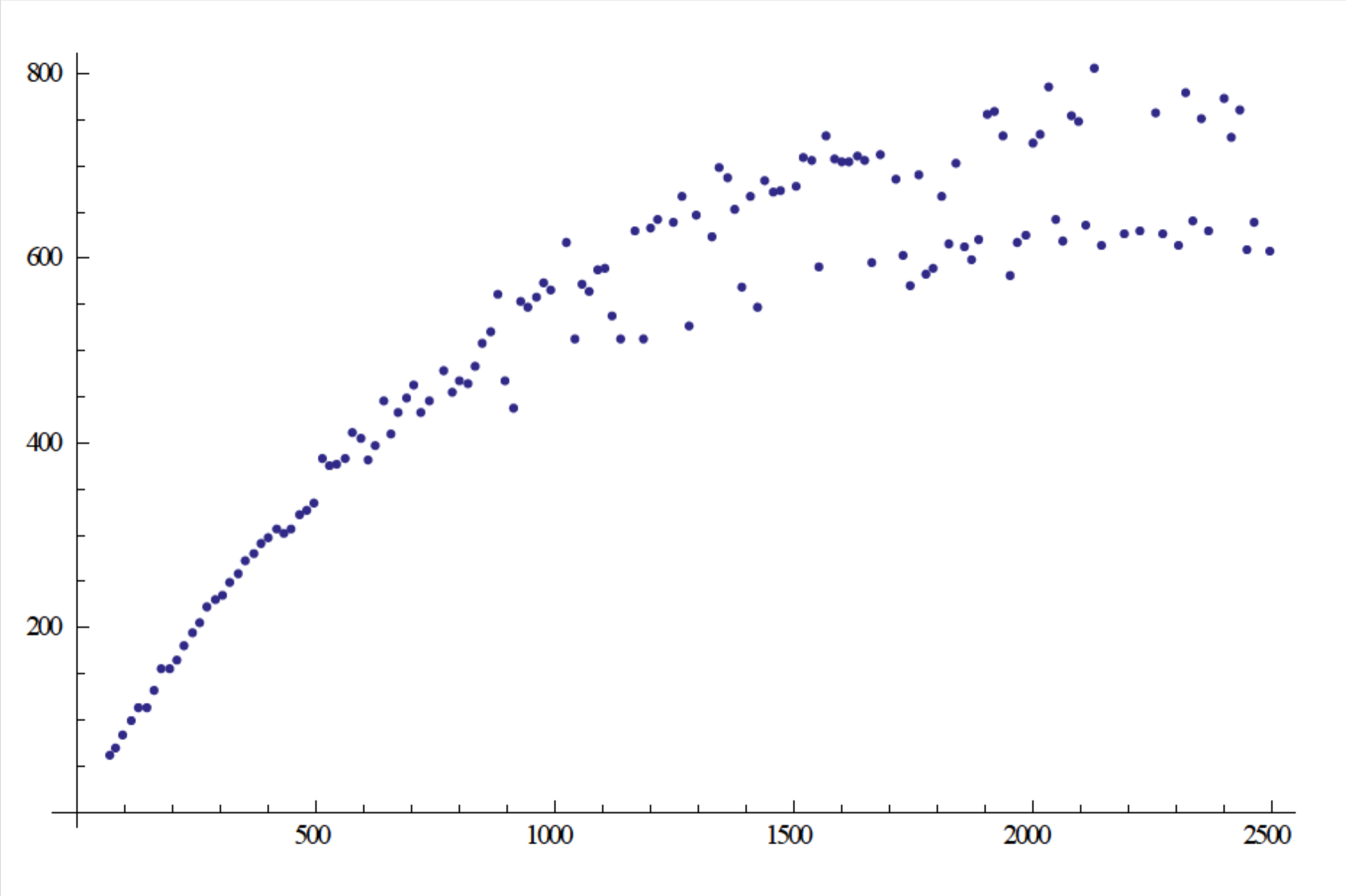
**PRAND**: GPU accelerated parallel random number generation library:
Using most reliable algorithms and applying parallelism of modern GPUs and CPUs
(L. Barash, LS, **CPC 2014**)

Highlights of PRAND - SSE-accelerated, GPU-accelerated, C, Fortran and CUDA, parallel streams

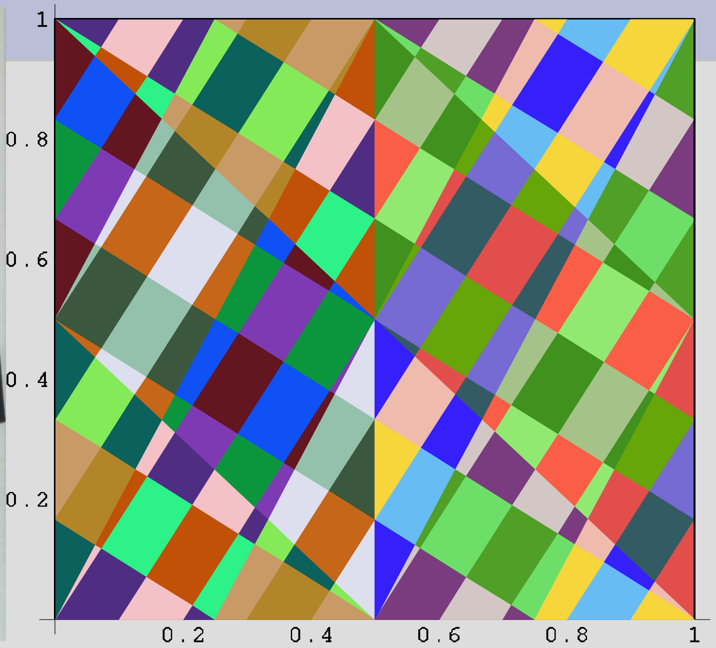
*Program summary URL:* http://cpc.cs.qub.ac.uk/summaries/AESB_v1_0.html
*Program obtainable from:* CPC Program Library, Queen's University, Belfast, N. Ireland

# MC integral

- MC integral
- Example with 3000 GPGPU (Lomonosov)



# PRAND: GM generators (on torus)




Kandisky, Malevich, Barash-Shchur

# MPI/GPU PA implementation

Technical problems to solve:

- Load balancing of all GPU nodes with an approximate same number of replicas;
- Minimize large memory exchange between nodes;
- …

# Conclusion

\- *Population annealing* approach is the *promising* tool for the *number of problems*:

- "complex" ground state
- the rough energy landscape
- optimization problems

\- *Population annealing* algorithm is the natural candidate for large-scale and *fully scalable simulations* with the *heterogeneous parallelism*.

RSF Russian Science Foundation

DIONICOS

*International Conference on Computer Simulation in Physics and beyond*

**September 24-27, 2018, Moscow, Russia**

**Organized by**

National Research University Higher School of Economics
Science Center in Chernogolovka
Landau Institute for Theoretical Physics

CSP2018 will be held at the building of Moscow Institute for Electronics and Mathematics,
National Research University Higher School of Economics
Tallinskaya ul. 34, Moscow

http://csp2018.ac.ru/