Electron Glasses: slow dynamics of a long-range interacting system

François Landes

ICTP, Trieste

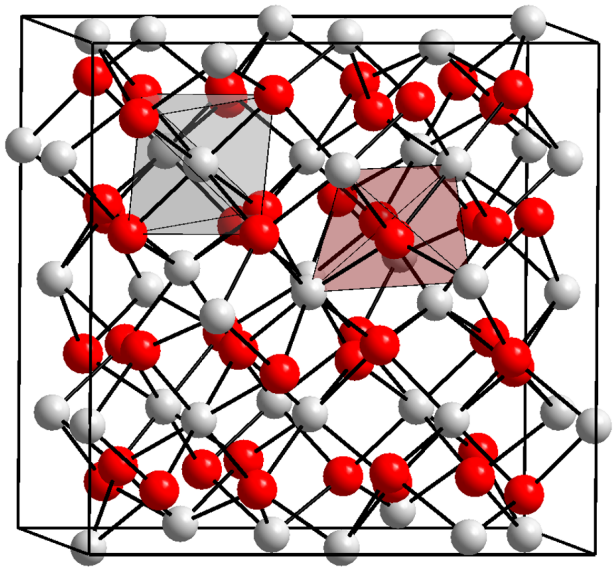
Workshop on Accelerated High-Performance Computing
in Computational Sciences

May 29, 2015

In collaboration with:
Markus Mueller (ICTP, Italy),
Matteo Palassini (UC Barcelona, Spain)
Ezequiel Ferrero (LiPhy Grenoble, France).

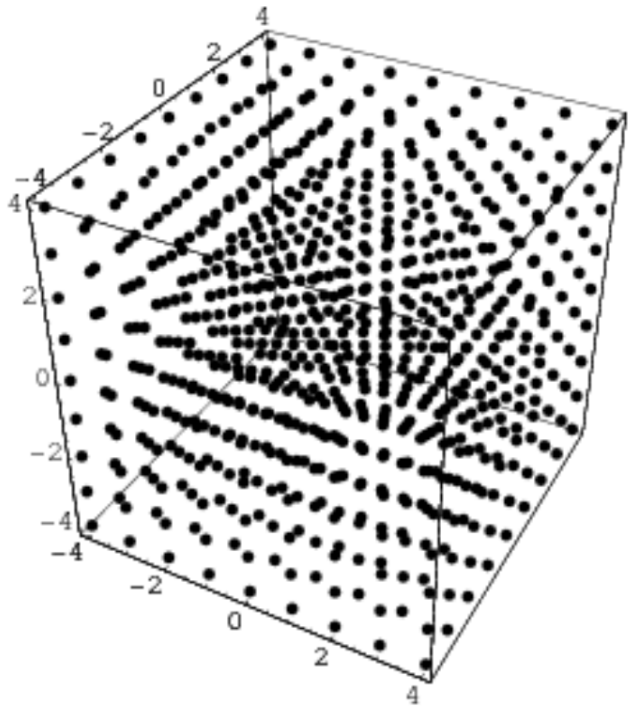
Outline

Physical System



- Semi-conductors with particular structure, (e.g. Inidium Oxide, In_xO)
- Low Temperature ($T \approx 4K$) $\Rightarrow$ slow, glassy dynamics
- Disorder (impurities, etc.)
  $\Rightarrow$ electrons are strongly localized
  $\Rightarrow$ electrons seen as "little balls" (high-school picture)

Model: geometry



- Square lattice, $N = L^d$ sites, $d = 2$ or 3 dims, $L \sim 32 - 256$ in 3D.
- Site $i \in [1, N]$ characterized by position $\mathbf{r}_i$
- Each site is either empty or occupied: $n_i = 0$ or 1
- Background charge: $K = +1/2$ on each site, balances the electrons' charges (there are KN e^-)

Model: interactions

- On-site disorder: each site gets a bonus energy from disorder,
  $\Rightarrow \phi_i(n_i - K)$, with random $\phi_i \in [-1, 1]$
- Aside from uniform background charge, no explicit screening !
  $\Rightarrow$ Coulomb interaction: $\sim \frac{(n_i - K)(n_j - K)}{|\mathbf{r}_i - \mathbf{r}_j|}$ (long-ranged !!)

Hamiltonian :

$$H = \sum_i \phi_i n_i + \frac{1}{2} \sum_i \sum_{j \neq i} \frac{(n_i - K)(n_j - K)}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{1}$$

Local energy level:

$$\varepsilon_i \equiv \phi_i + \sum_{j \neq i} \frac{(n_j - K)}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{2}$$

Dynamics

- jump from a to b : total energy changes by:
  $$\Delta H = \varepsilon_b - \varepsilon_a - 1/\mathbf{r}_{ab}$$
- At $T = 0K$, only transitions that lower the energy allowed:
  $$\Delta H < 0$$
- Electrons actually tunnel between sites (rate $\sim e^{-2r_{ab}/\xi}$), but let's use uniform rate 1.
- At $T > 0$, tunnelling accepted with rate $\min(1, e^{-\Delta H/k_B T})$. No Ground State at $T > 0$.

Outline

Sequential Algorithm

Monte-Carlo Dynamics:

- Pick a pair (a, b) at random
- Compute $\Delta H = \varepsilon_b - \varepsilon_a - 1/\mathbf{r}_{ab}$
- If $\Delta H < 0$:

  If $n_a = 1, n_b = 0$:

  Do: $n_a = 0$, $n_b = 1$. Update all the ε_i.

Energy Update:
If jump $a \to b$ was accepted: $\forall i \in [1, N]$, do $\varepsilon_i += \frac{1}{r_{ib}} - \frac{1}{r_{ia}}$.
(except for $i = a$ and $i = b$, it is a bit different).

Problems:

- I) Updating all the ε_i is $\sim$ **O(N) operations**. Typically takes at least 50% of computing time, up to 80%.
- II) In real life, as time increases, the **dynamics slows down** (hence the name, electron **glasses**).

  $\Rightarrow$ In the simulation, the failure rate increases a lot !

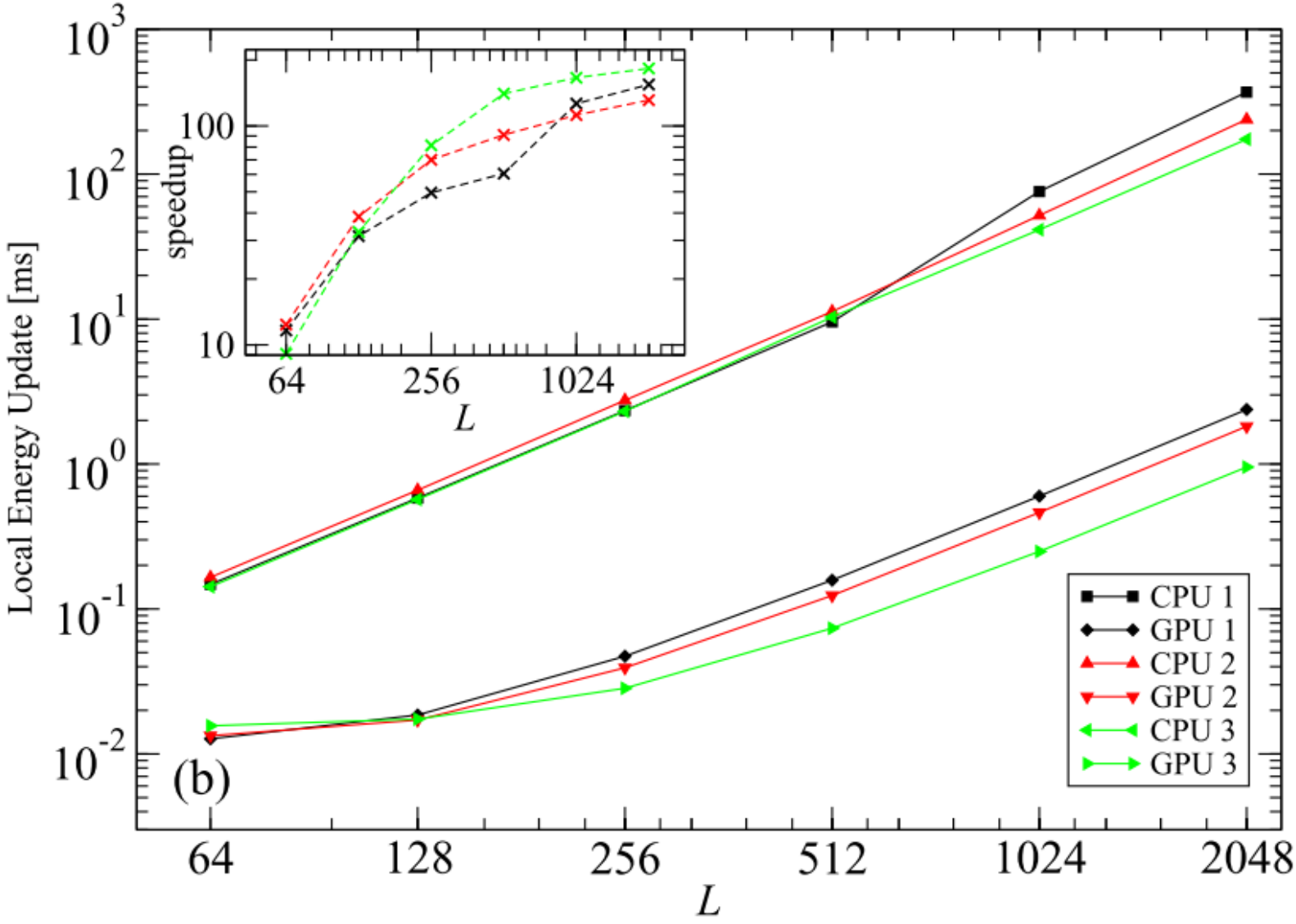
I) Energy updates: The solution

In: 2 integers, a and b.
Out: N changes of float values.

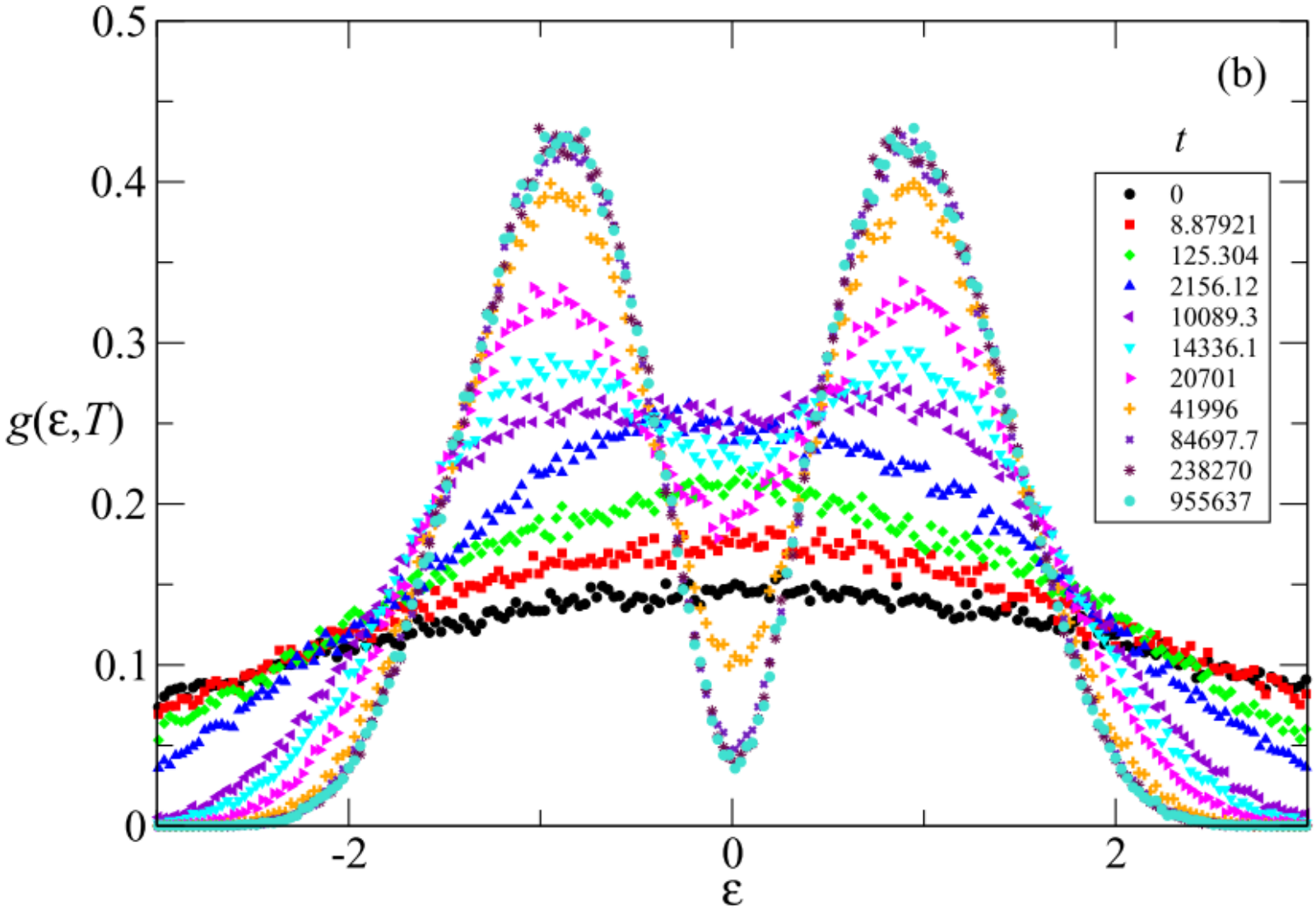
Solution: *"embarrassing parallelism"*:
1 thread updates the energy of 1 site. As many threads as sites.
So simple $\Rightarrow$ *Shame*

Performance gain: at least $\times 2$, probably more like $\times 5$ (time spent in the updates becomes negligible, a few % of total time at most).

I) Energy updates: Performance



II) Fail rate of Monte Carlo steps: slowdown



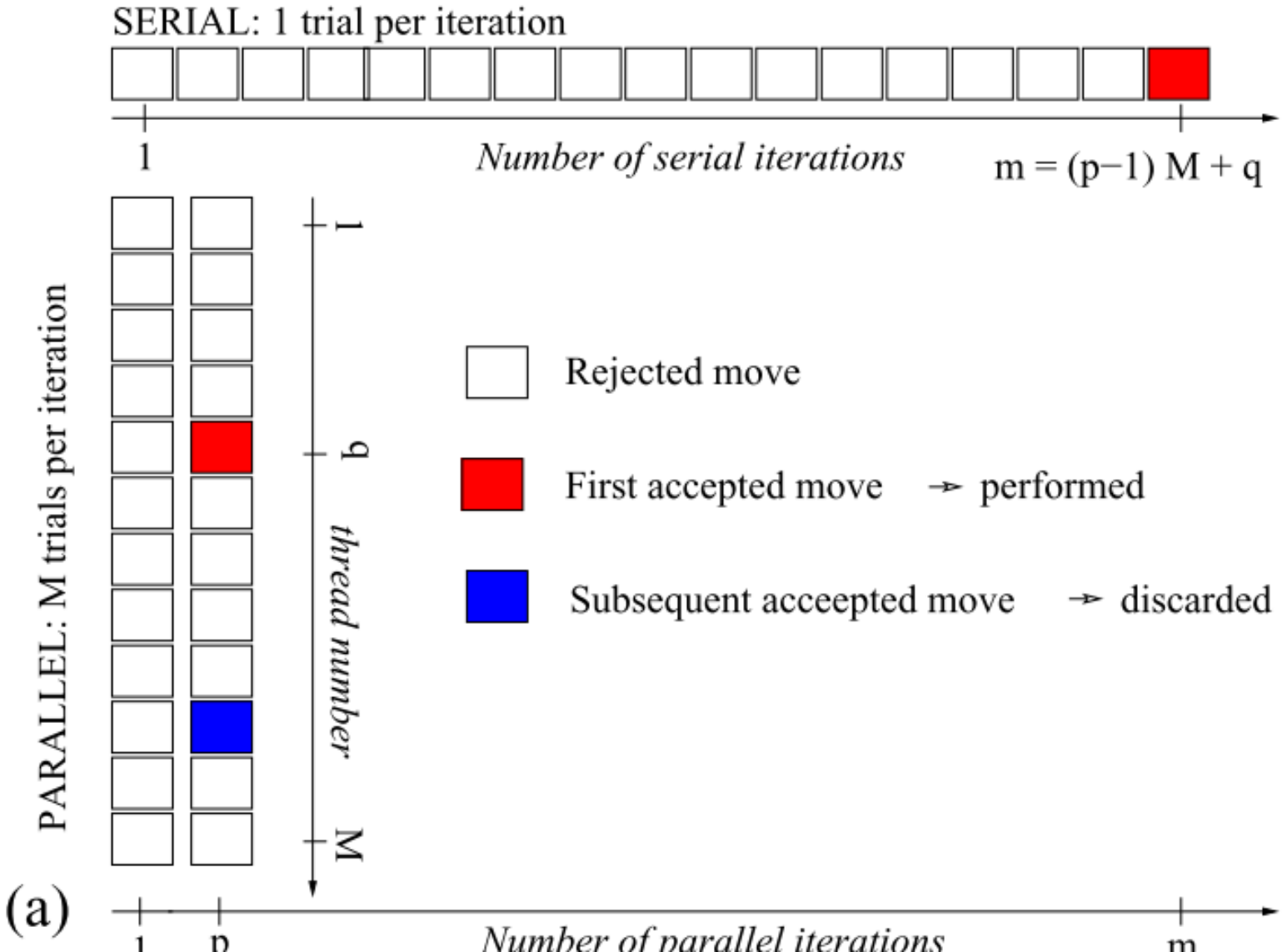
II) Fail rate of Monte Carlo steps: the Parallel KMC Solution

As we converge to the Ground State, more and more jump attempts fail. ("Glassy" dynamics, logarithmic evolution).
In: N^2 possible jumps.
Out: 1 allowed jump.

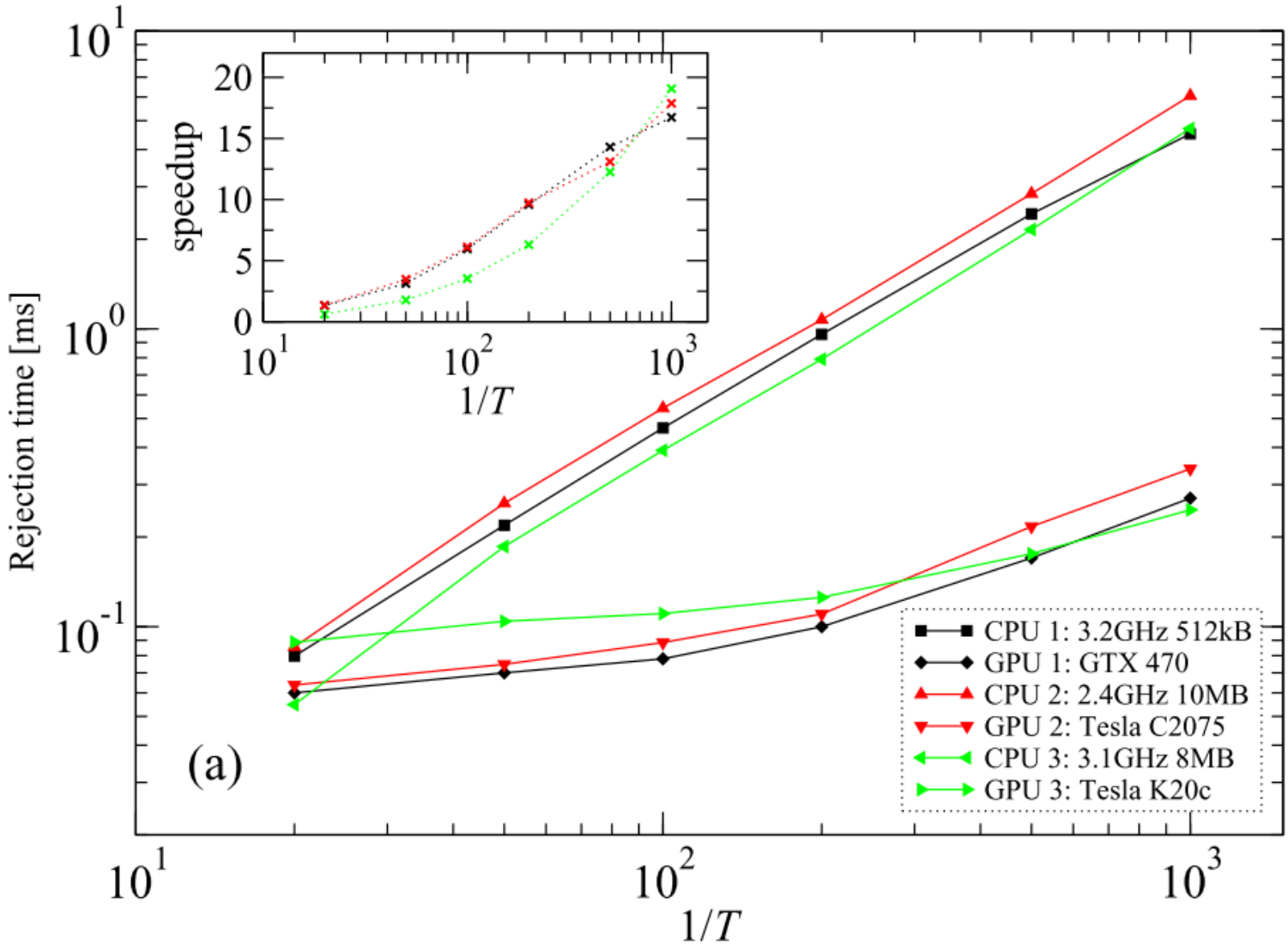
Solution: *Kinetic Monte Carlo*:
M threads run in parallel, making independent random attempts. The first one (in natural order) to succeed is done. Other successes are thrown away.

Ferrero, E. E., Kolton, A. B., & Palassini, M. (2014). *Parallel kinetic Monte Carlo simulation of Coulomb glasses.* In 15TH INTERNATIONAL CONFERENCE ON TRANSPORT IN INTERACTING DISORDERED SYSTEMS (TIDS15) (pp. 71 - 76). AIP Publishing.

II) Fail rate of Monte Carlo steps: the Parallel KMC Solution



II) Fail rate of Monte Carlo steps: the Parallel KMC Solution: Performance



Outline

Alternative (Complementary to KMC): energy-based search

Original Idea developed in:

Glatz, A., Vinokur, V. M., Bergli, J., Kirkengen, M., & Galperin, Y. M. (2008). *The Coulomb gap and low energy statistics for Coulomb glasses.* Journal of Statistical Mechanics: Theory and Experiment.

Palassini, M., & Goethe, M. (2012). *Elementary excitations and avalanches in the Coulomb glass.* Journal of Physics: Conference Series.

Alternative (Complementary to KMC): energy-based search

Case (1):

- Separate empty and occupied sites
- Sort occupied sites from higher to lower energy
- Sort empty sites from lower to higher energy
- if $\varepsilon_{max}^{occ} > \varepsilon_{min}^{empty}$
  then $\Delta H < 0$ for sure:
  $\Rightarrow$ do this jump, $a \rightarrow b$.

Jump independent from the distance r_{ab} !

Sorting is $\sim O(N \log N)$,
but also done in parallel by **Thrust**.

Alternative (Complementary to KMC): energy-based search

Case (2):

- *Separate empty and occupied sites*
- *Sort occupied sites from higher to lower energy*
- *Sort empty sites from lower to higher energy*
- $\varepsilon_{max}^{occ} < \varepsilon_{min}^{empty} \quad \Rightarrow$ Case (2).
  $\Rightarrow$ Define $\mu = (\varepsilon_{max}^{occ} + \varepsilon_{min}^{empty})/2$

Particles in site with $\varepsilon_a \in [\mu - 1, mu]$ may be allowed to jump to empty sites with $\varepsilon_b \in [\mu, \varepsilon + 1]$, thanks to the term $-1/r_{ab}$. Other pairs are forbidden to jump.

Sorting is still $\sim O(N \log N)$,
and also is still done in parallel by **Thrust**.

Alternative (Complementary to KMC & Energy-based Search): Systematic Search

After a sufficient number of failures at doing Parallel KMC steps, or if an incomplete exploration of the "case (2)" possibilities yields nothing, try a full search ($\sim N^2$ possibilities !)

$\Rightarrow$ allows to **exactly** check if the simulation is finished.

With GPU, this computation is manageable. (parallelism strategy: $N-1$ attempts per thread).

But in the end, only a debugging tool.

Summary:

Program main functions:

- When a search finds a jump allowed, the update of the system's state is done in **parallel (N threads)**.
- Perform KMC search: $M = 2^8 - 2^{18}$ **threads in parallel**.
- If no success in the KMC search:
  - **Sort** the energies of occupied and empty sites, using the **parallel thrust library**.
  - Find jumps in Case (1) or Case (2).
  - If Case (2) is not exhaustive, and no jump found:
    - *Systematic Search* [optional]

When no jump is found: (pseudo-) **Ground State** reached !

Additional Trick :
Sort the energies once, then attempt several jumps before starting any other kind of search (KMC or re-sorting the jumps).

Thank You !