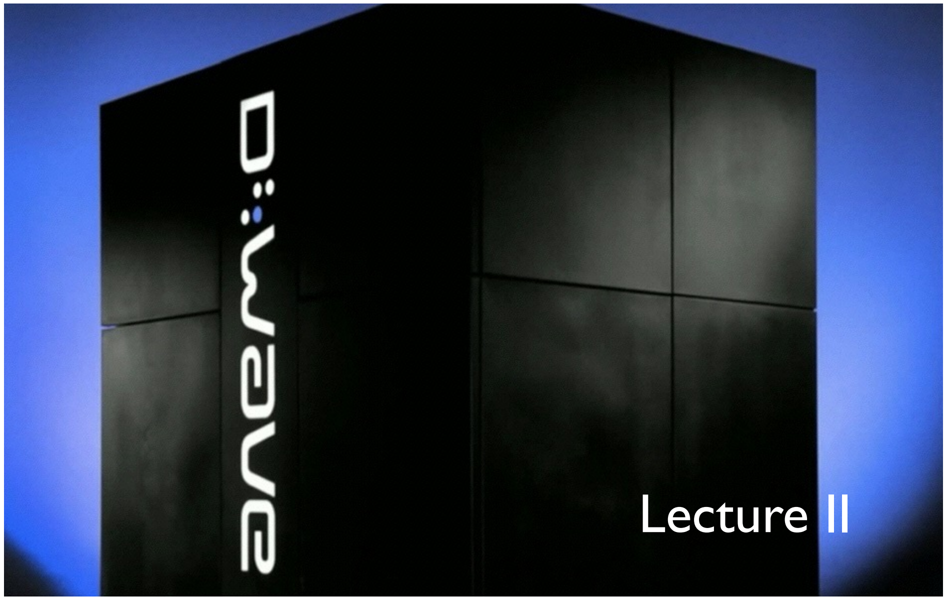


Lecture II

Helmut G. Katzgraber

TEXAS A&M UNIVERSITY

Advanced Monte Carlo & Spin Glasses

Outline

- Monte Carlo in statistical physics
  - Metropolis algorithm
  - Ising model implementation
  - Equilibration times
  - Autocorrelation times
- When does Monte Carlo fail?
  - Critical slowing down (*Krauth*)
  - Low temperatures
- Speedup at low temperatures
  - Spin glasses
  - Parallel tempering
- Optimization & Complexity
  - P versus NP
  - Exact versus heuristic
- Optimization algorithms:
  - Simulated annealing
  - Parallel tempering
  - Genetic algorithms
- Other optimization methods
  - Quantum annealing, …

Literature used

- Monte Carlo, spin glasses & optimization:
  - "*Introduction to Monte Carlo Algorithms*" – Krauth
  - "*Introduction to Monte Carlo Methods*" – HGK (arXiv:0905.1629)
  - "*Monte Carlo Methods in Statistical Physics*" – Newman & Barkema
  - "*Optimization Algorithms in Physics*" – Hartmann & Rieger
  - "*Scientific Programming*" – Zachary
  - "*Statistical Mechanics of Phase Transitions*" – Yeomans
  - "*Spin glasses and complexity*" – Stein & Newman
  - "*New Optimization Algorithms in Physics*" – Hartmann & Rieger
  - "*The Nature of Computation*" – Moore & Mertens
  - "*Phase Transitions in Combinatorial Opt.*" – Hartmann & Weigel
  - … and many more books…

Monte Carlo in statistical physics…

… or how do we measure observables?

Where has Monte Carlo been successful?

- *Monte Carlo-like sampling* can be applied to problems across disciplines:

Chemistry	Chemical reactions, …
Physics	Statistical mechanics, nuclear physics, …
Biology	Biomolecules, …
Sociology	Social networks, …
Economy	Market simulations, …
Engineering	Structural integrity simulations, …
Geology	Water seepage, …
Linguistics	Pattern matching in texts, …
Medicine	Disease spreading, …
Astronomy	Exoplanet detection, …
…	

Recall importance sampling…

- Goal: Compute the average of an observable *O*

$$\langle \mathcal{O} \rangle = \frac{\sum_s \mathcal{O}(s) e^{-\mathcal{H}(s)/kT}}{\sum_s e^{-\mathcal{H}(s)/kT}}$$

- Extend this with a distribution (think importance sampling):

$$\langle \mathcal{O} \rangle = \frac{\sum_s [\mathcal{O}(s)/\mathcal{P}(s)] e^{-\mathcal{H}(s)/kT}}{\sum_s [1/\mathcal{P}(s)] e^{-\mathcal{H}(s)/kT}}$$

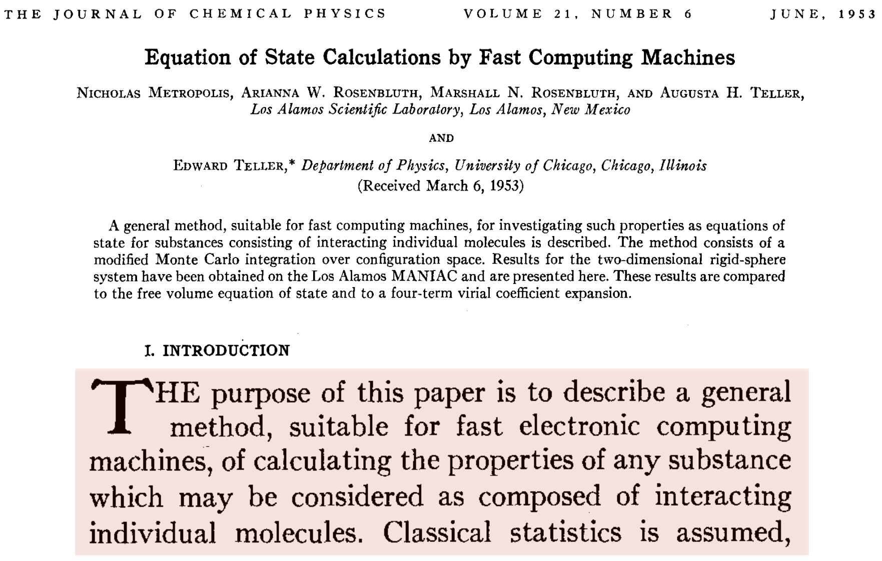
- If $\mathcal{P}(s)$ is the Boltzmann distribution we obtain

$$\langle \mathcal{O} \rangle = \frac{1}{M} \sum_i \mathcal{O}(s_i)$$ sum of *P*-distributed measurements!

where the states s_i are selected *according to a Boltzmann distribution.*

Sure… But how do we sample a Boltzmann distribution?

Metropolis paper



THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

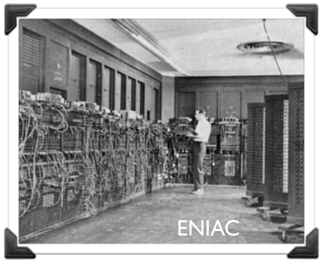
I. INTRODUCTION

THE purpose of this paper is to describe a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules. Classical statistics is assumed,

History behind the Metropolis paper

Phys. Plasmas 12, 057303 (05)

- 50 years later at a Los Alamos meeting:
  - Only M. Rosenbluth attended, although with terminal cancer.
  - Metropolis mainly contributed CPU time on MANIAC.
  - von Neumann and Ulam invented the Monte Carlo method in 1946 and pointed out that it could be used for simulations.
  - Teller: Statistical averages can be made as ensemble averages.
  - Interesting author list: two couples. How often does this happen?



- Why Los Alamos?
  - The US was building the atomic bomb. At least one good thing came out of this.

Metropolis algorithm

- Start by generating a *Markov chain* of successive states

$$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$$

the new state is generated with a probability $\mathcal{P}_{\rm eq}(s) = \frac{1}{Z} e^{-\mathcal{H}(s)/kT}$

- A state s occurs with a probability $\mathcal{P}_k(s)$ at the k-th step, described by the *master equation*:

$$\mathcal{P}_{k+1}(s) = \mathcal{P}_k(s) + \sum_{s'} [\mathcal{T}(s' \rightarrow s)\mathcal{P}_k(s') - \overset{\text{trans. prob.}}{\mathcal{T}}(s \rightarrow s')\mathcal{P}_k(s)]$$

states to s

states from s

- For $k \rightarrow \infty \quad \mathcal{P}_k(s) \rightarrow \mathcal{P}_{\rm eq}$.

- Detailed balance:

$$\mathcal{T}(s' \rightarrow s)\mathcal{P}_{\rm eq}(s') = \mathcal{T}(s \rightarrow s')\mathcal{P}_{\rm eq}(s) \quad \text{all terms in sum = 0}$$

this ensures that the process is reversible (*ergodic*)!

Metropolis algorithms contd.

- When the system is in *thermal equilibrium*:

$$\frac{\mathcal{T}(s \rightarrow s')}{\mathcal{T}(s' \rightarrow s)} = \exp[-(\mathcal{H}(s') - \mathcal{H}(s))/kT] = \exp[-\Delta\mathcal{H}(s,s')/kT]$$

- There are different choices for $\mathcal{T}$ that satisfy the general equation:

$$\mathcal{T}(x)/\mathcal{T}(1/x) = x \qquad x = \exp(-\Delta\mathcal{H}/kT)$$

- Metropolis-Hastings algorithm: $\mathcal{T}(x) = \min(1,x)$

$$\mathcal{T}(s \rightarrow s') = \begin{cases} \Gamma, & \text{if } \Delta\mathcal{H} \leq 0 \\ \Gamma e^{-\Delta\mathcal{H}(s,s')/kT}, & \text{if } \Delta\mathcal{H} \geq 0 \end{cases} \qquad \Gamma^{-1} \sim \text{time}$$

- Heat-bath algorithm: $\mathcal{T}(x) = x/(1+x)$

see Newman & Barkema for details.

Example: Algorithm for the Ising model

- Remember:

$$\mathcal{H} = -\sum_{\langle ij \rangle} J_{ij} S_i S_j$$

- Updates:
  - The states s correspond to spin configurations $\{S_i\}$.
  - The move between s and s' can be arbitrary.
  - If s and s' are too far apart, the move will not be accepted.
  - Common choice: Flip one randomly-chosen spin S_i with

$$\mathcal{T}(S_i \rightarrow -S_i) = \begin{cases} \Gamma, & \text{for } S_i = -\text{sign}(h_i) \\ \Gamma e^{-2S_i h_i/kT}, & \text{for } S_i = \text{sign}(h_i) \end{cases}$$

where $h_i = \sum_{j \neq i} J_{ij} S_j$ is the *effective field* felt by S_i.

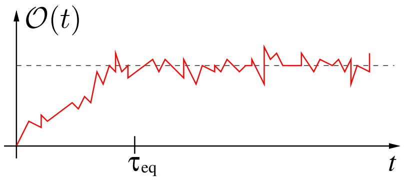
Practical implementation

- Bare-bones implementation:
  - If the change in energy is favorable, we always flip the spin.
  - If the change in energy is not favorable, we flip with a given probability.
  - For infinite time this converges to the estimate of an observable *O*.

```
algorithm ising_metropolis(T,steps)
    initialize starting configuration S
    initialize O = 0

    for(counter = 1 ... steps) do
        generate trial state S'
        compute p(S -> S',T)
        x = rand(0,1)
        if(p > x) then
            accept S'
        fi

        O += O(S')
    done
```

- Some considerations swept under the rug so far…
  - Is this sampling the *equilibrium* distribution?
  - What about *autocorrelation* effects in the Markov chain?

Things to consider: equilibration…



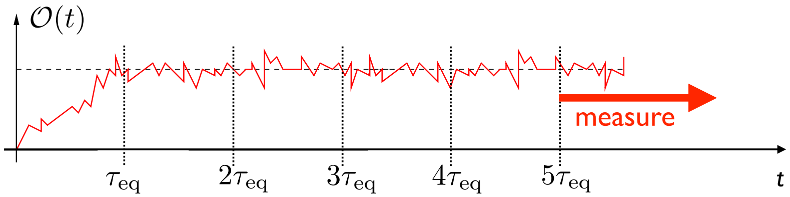
- The initial configuration is *arbitrary*.
- To obtain a correct estimate of *O*, we need to ensure we are sampling the *equilibrium* state.

- How do we check for this?
  - Monitor *all* observables as a function of time, e.g., *O*(*t*). Why all?
  - The time it takes for *O*(*t*) ~ "constant" is the *equilibration time.*

- Properties of $t_{\rm eq}$:
  - Increases with the number variables *N*.
  - Increases with decreasing temperature.
  - Measured in *Monte Carlo sweeps*: 1 MCS = *N* update attempts.

Equilibration time contd.

- Recommendations for simulations:
  - Always store time-dependent measurements every 2^k steps.
  - Once $\langle \mathcal{O}(t=\infty) - \mathcal{O}(t)\rangle \sim 0$, do *not* start measuring. Let the system thermalize for *at least* an additional 5 – 10 times longer to ensure full thermalization.

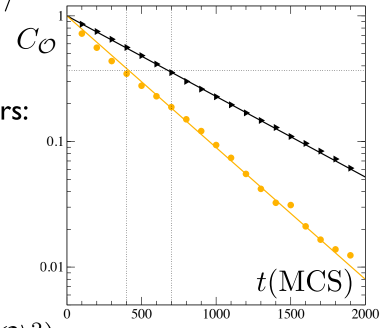


- Note:
  - It can be shown analytically that the equilibration time is the maximum of all autocorrelation times.

Things to remember: autocorrelations…

- To avoid correlations between measurements, study autocorrelation functions for observables *O*:

$$C_{\mathcal{O}}(t) = \frac{\langle \mathcal{O}(t_0)\mathcal{O}(t_0+t)\rangle - \langle \mathcal{O}(t_0)\rangle\langle \mathcal{O}(t_0+t)\rangle}{\langle \mathcal{O}^2(t_0)\rangle - \langle \mathcal{O}(t_0)\rangle^2} \sim \exp(-t/\tau_{\rm auto})$$



- This ensures that measurements are independent.
- Autocorrelation effects influence errors:

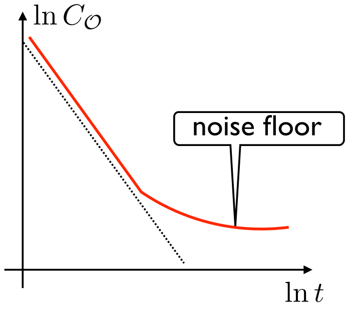
$$\Delta\mathcal{O} = \sqrt{\frac{\langle \mathcal{O}^2\rangle - \langle \mathcal{O}\rangle^2}{(M-1)}(1+2\tau_{\rm auto})}\ .$$

- Integrated autocorrelation time:

$$\tau_{\rm auto}^{\rm int} = \frac{\sum_{t=1}^{\infty}\left(\langle \mathcal{O}(t_0)\mathcal{O}(t_0+t)\rangle - \langle \mathcal{O}\rangle^2\right)}{\langle \mathcal{O}^2\rangle - \langle \mathcal{O}\rangle^2}$$

Practical approach: Binning

- Measuring autocorrelation times in simulations can be tedious:
  - The "noise floor" depends on the model and must be excluded.
  - Autocorrelation functions might not be pure exponentials.
  - The shape of the function might change with time.
- Binning:
  - Divide the *M* measurements into *p* bins.
  - If $M/p \gg \tau_{\rm auto}$ then the averages computed in each bin over *M*/*p* measurements should be relatively uncorrelated ⟶ statistical error bar.



- Alternative to estimate error bars:
  - Do *M* different simulations with *different* initial conditions & average.

Further MC-like algorithms? Many!

Variations…? Many!

- Cluster algorithms (see *Krauth* lecture):
  - Help overcome critical slowing down at phase transitions.

  Wolff, Swendsen & Wang (87)
  Houdayer (01)
- Flat-histogram methods:
  - Multicanonical method, broad histogram method, Wang Landau, …
  - Allow for the computation of the free energy.

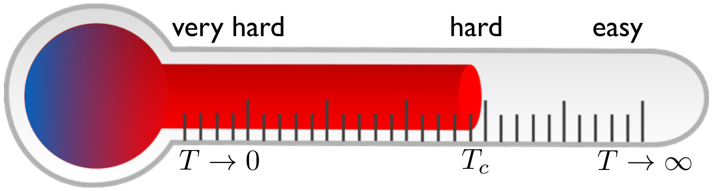
  Berg (91)
  Wang & Ladau (01)
- Quantum Monte Carlo:
  - Extension to quantum systems.

  Suzuki (93)
- Simulated/Quantum annealing:
  - Minimization routine based on the reduction of fluctuations.

  Das (03)
  Kirkpatrick et al. (83)

Where does simple Monte Carlo "fail"?

Regimes where MC sampling is inefficient



- At phase transitions autocorrelation times diverge. This effect is known as *critical slowing down.* ⟶ Cluster Algorithms (see *Krauth* lecture)
- Close to the ground state (zero temperature) sampling becomes inefficient because $\mathcal{T} = \min(1, e^{-\Delta E/T})$ is very small when $T \to 0$.
- Rough energy landscapes where ΔE is large and therefore, again, acceptance probabilities are small.

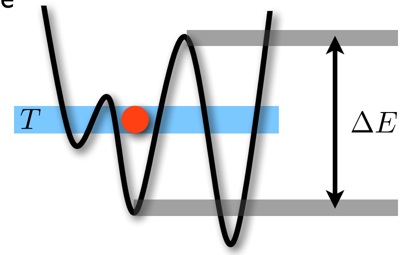
Slow convergence at low temperatures … and rough energy landscapes

Monte Carlo & Rugged energy landscapes

- Systems with rugged energy landscapes (metastable states).
- At low temperature, when ΔE is large

  $$\mathcal{T} = \min(1, e^{-\Delta E/T})$$

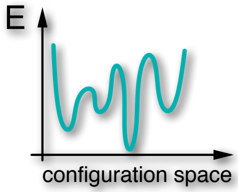
  is “never” accepted.
- Sampling all of phase space becomes inefficient.



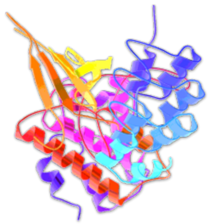
- How can we resolve the problem?
  - Tunnel trough barrier.
  - Heat up the system to overcome the barrier.
- Where does this happen?
  - All over the place… Especially in bio applications and optimization.

Typical problems with complex phase space

- Several physical problems have rugged energy landscapes.
- Randomness or frustration produce competing interactions and thus a complex energy landscape.



- Examples:
  - Spin glasses: $\mathcal{H} = -\sum_{ij} J_{ij} S_i S_j$ $\mathcal{P}(J_{ij})$ random
  - Structural glasses
  - Polymers in random media (interfaces)
  - Biomolecules (proteins)
  - Quantum wave function reconstruction
  - Reconstruction of geological structures from seismic measurements, ...

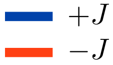


Nontrivial toy model: Spin glasses

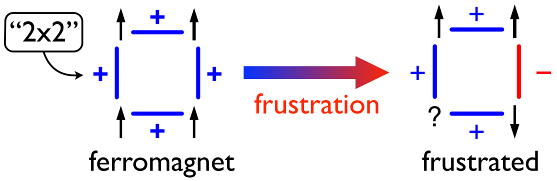
Adding frustration…

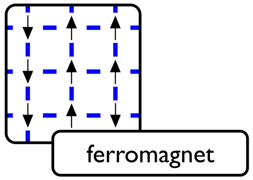
- General Hamiltonian:

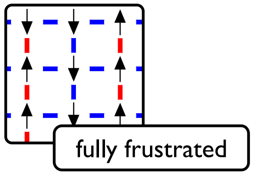
$$\mathcal{H} = -\sum_{\langle ij \rangle} J_{ij} S_i S_j$$

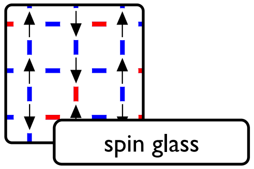


- Introduce frustration between the spins:




ferromagnet


fully frustrated


spin glass

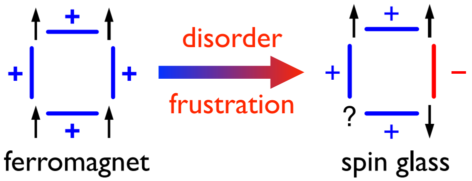
- Properties of the fully-frustrated Ising model:
  - Huge ground-state degeneracy.
  - Complex energy landscape, T_c = 0 in 2D.
  - $\prod_{\square} J_{ij} < 0 \quad \forall\, i,j$
- What happens if we add randomness, too?

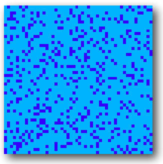
Spin glasses: (*Magnetic*) *Frustration*

- Add disorder… Edwards-Anderson spin glass

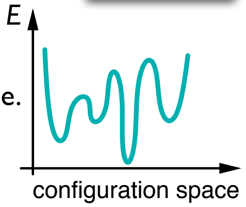
$$\mathcal{H} = -\sum_{ij} J_{ij} S_i S_j - h \sum S_i \qquad J_{ij} \text{ random}$$

- … obtain loads of frustration:

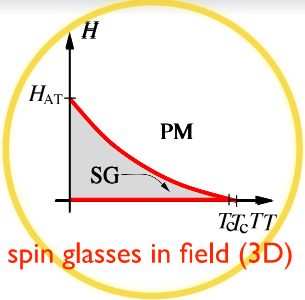




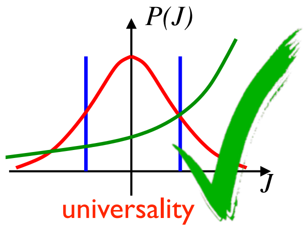
- Many metastable states, slow relaxation.
- Nontrivial aging, memory effects, rough landscape.
- NP hard – perfect for testing algorithms!
- No transition below *d* = 3, mean field for *d* ≥ 6.



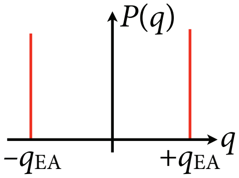
Selected big challenges



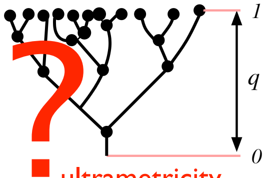
spin glasses in field (3D)



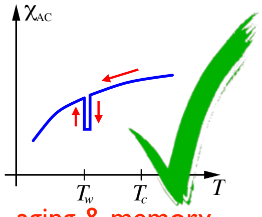
universality



nature of the spin-glass state



ultrametricity



aging & memory

A brief word on the history…

Giorgio Parisi

Daniel Fisher

David Huse

and many more…

- Brief incomplete history…
  - mid 70's: Edwards-Anderson Ising spin glass model (J_{ij} random):

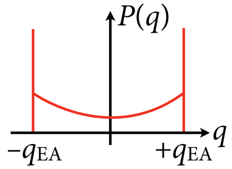
$$\mathcal{H} = -\sum_{\langle ij\rangle} J_{ij}S_iS_j \xrightarrow{\text{mean-field approx.}} \sum_{\langle ij\rangle} \to \sum_{i,j}$$

  - mid 70': Mean-field Sherrington-Kirkpatrick (SK) spin glass.
  - 70's: Parisi mean-field solutio ry breaking - RSB).
  - 80's: Scaling-li (DP) for short-range systems.
  - 90's: Chaotic p picture (CP) by Newman & Stein.

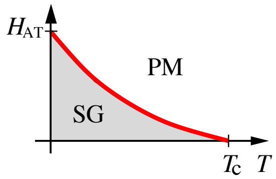
To date controversial…

RSB vs DP

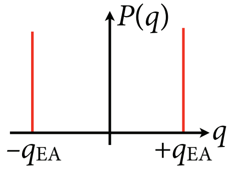
- Countable infinity of pure states in the thermodyn. limit.



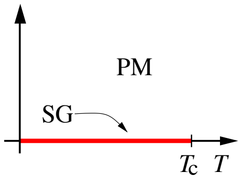
- Nontrivial ground state
- Spin-glass state in a field:



- One pair of pure states in the thermodyn. limit.



- Trivial ground state
- No spin-glass state in a field:



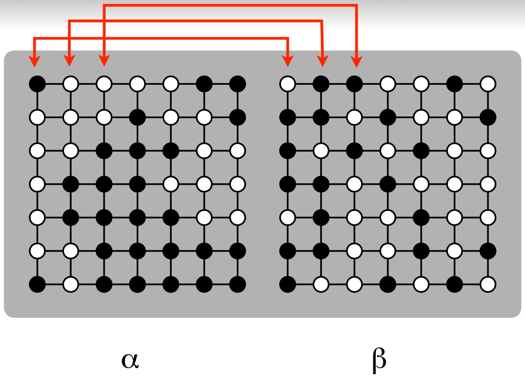
Incidentally, how do we measure "order"?

- The ground state has *no spatial order* (m = 0).
- Above T_c spins fluctuate.
- Below T_c spins frozen.
- Compare spins at time t_0 with spins at time $t + t_0$:

$$q = \frac{1}{N}\sum_{i=1}^{N} S_i(t_0)S_i(t+t_0)$$

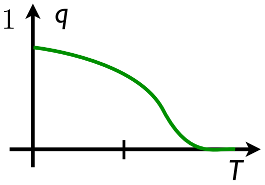
- Not practical in simulations. Better:

$$q = \frac{1}{N}\sum_{i=1}^{N} S_i^{\alpha}S_i^{\beta} \qquad (m_{\mathrm{ferro}} \to q_{\mathrm{glass}})$$



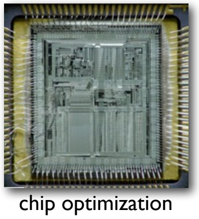
● $S_i = +1$

○ $S_i = -1$



Applications beyond disordered magnets

- The models can describe *different materials* and many systems with competing interactions on a graph:
  - Computer chips:
    - S_i component
    - J_{ij} wiring diagram
  - Economic markets:
    - S_i agent inclination
    - J_{ij} portfolio interactions



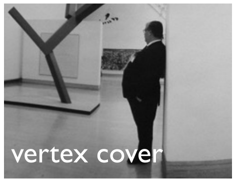
chip optimization

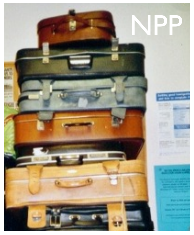
markets

- Other applications:
  - Quantum error correction (topological quantum computing).
  - Neural networks.
  - Optimization problems …

Importance of spin glasses in optimization

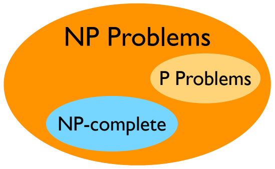
- Selected common optimization problems:
  - k-SAT $(x_{11} \text{ OR } x_{12} \text{ OR } x_{13}) \text{ AND } (x_{21} \text{ OR } x_{22} \text{ OR } x_{23}) \text{ AND}\ldots$
  - Number partitioning (NPP)
  - Minimum vertex covers
  - Spin glasses, proteins, …


vertex cover


NPP

- What do these have in common?
  - They are typically problems in NP.
  - They have a very rough energy/cost function landscape.
  - They map onto spin-glass Hamiltonians:

$$\mathcal{H}(S_i) = \sum_{i \neq j}^{N} Q_{ij} S_i S_j \qquad S_i \in \{\pm 1\}$$

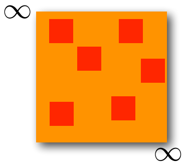

NP Problems

P Problems

NP-complete

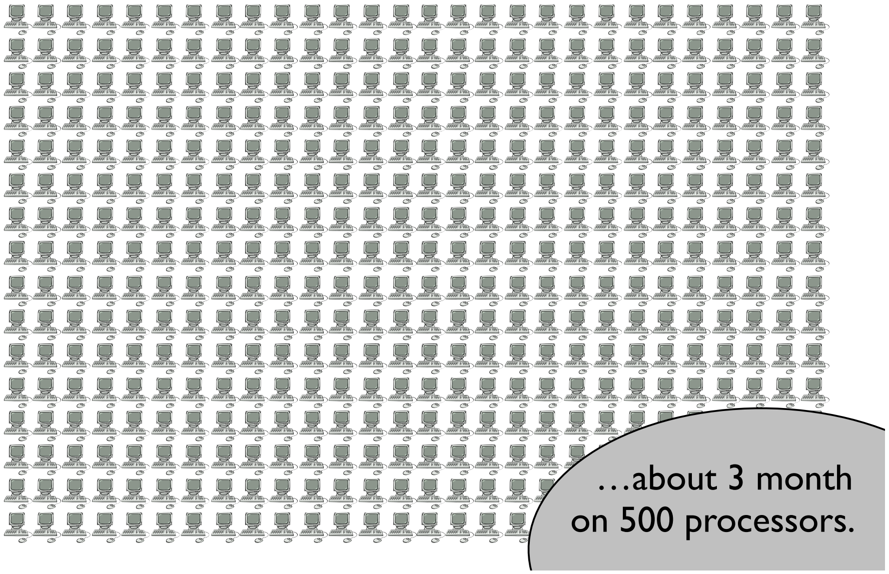
How can we study these systems?

- Analytically: only mean-field solution or qualitative descriptions.



- Numerically: Optimal problem for huge computers.
  - Challenges:
    - Exponential number of competing states (usually NP hard).
    - Relaxation times diverge exponentially with the system size.
    - Extra overhead due to disorder averaging.
    - This means small systems only.
  - Any study requires…
    - … clever models,
    - … better algorithms,
    - … very large computer clusters.

How large is large? A typical project takes…



…about 3 month on 500 processors.

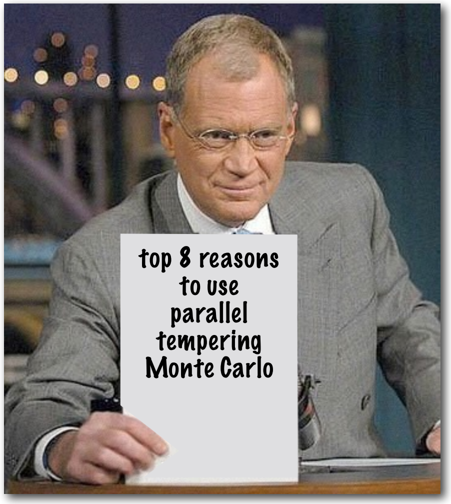
Speeding up simulations:

Parallel tempering Monte Carlo

Top 10 reasons to use parallel tempering

Geyer (91)
Hukushima & Nemoto (96)

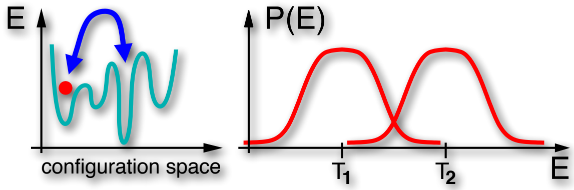
1. Very efficient.
2. Simple to implement.
3. Only few parameters.
4. It is practical (several *T*'s).
5. Small numerical overhead.
6. It is easy to parallelize.
7. Mix with other algorithms.
8. It is '*Made in Japan*.'



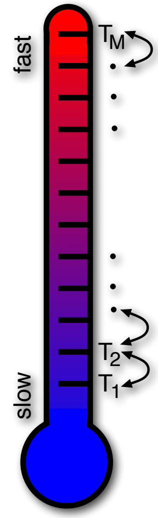
Exchange (parallel tempering) Monte Carlo

Hukushima & Nemoto (96)

- Idea:
  - Simulate *M* copies of the system at different temperatures with $T_{\max} > T_c$ (typically $T_{\max} \sim 2T_c^{MF}$).
  - After *each* lattice sweeps, attempt to swap neighbors: easy crossing of barriers.



- What has to be tuned?
  - Number of temperatures *M*.
  - Position of the temperatures.



Parallel tempering: algorithm and details

- Outline of the algorithm:
  - Perform a Monte Carlo update between *neighboring* replicas:

$$\mathcal{T}[(E_i, T_i) \to (E_{i+1}, T_{i+1})] = \min\{1, \exp[\Delta E_{i+1,i} \Delta\beta_{i+1,i}]\}$$

$$\Delta E_{i+1,i} = E_{i+1} - E_i$$

[obeys detailed balance]

$$\Delta\beta_{i+1,i} = 1/T_{i+1} - 1/T_i$$

- Pseudo code implementation:

note: keep *T*'s, swap pointers

```
algorithm parallel_tempering(*energy,*temp,*spins)

    for(counter = 1 ... (num_temps - 1)) do
        delta = (1/temp[i] - 1/temp[i+1])*(energy[i] - energy[i+1])
        if(rand(0,1) < exp(delta)) then
            swap(spins[i],spins[i+1])
            swap(energy[i],energy[i+1])
        fi
    done
```

Example: Ising spin glass in *d* = 3

- Equilibration times:

$$\tau_{\rm eq}^{\rm PT} \approx 300\,{\rm MCS}$$

$$\tau_{\rm eq}^{\rm SM} \approx 10^6\,{\rm MCS}$$

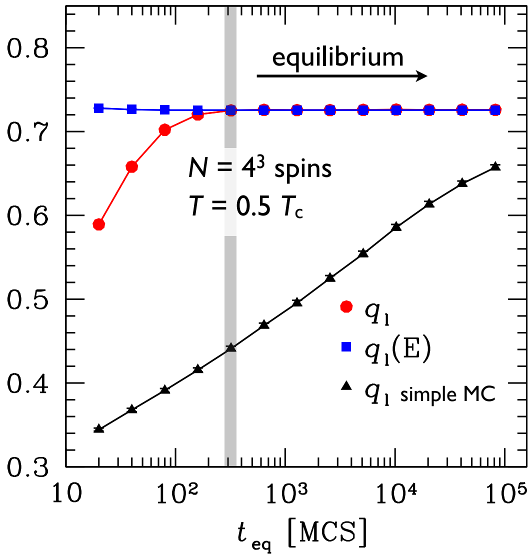
- Equilibration test (Gaussian disorder):

$$q_l(E) = \frac{2T|E|}{z} + 1$$

$$z = 2D$$

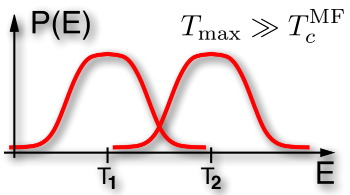
Once both agree, the system is in equilibrium.

Katzgraber *et al.* PRB (01)



How many temperatures do we need?

- Two possible scenarios:
  - Temperatures too far apart: parallel simple Monte Carlo chains.
  - Temperatures too close: overhead.



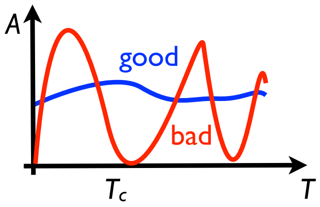
- What determines the number *M* of temperatures?
  - The energy distributions of the system at T_1 and T_2 have to overlap.
  - Because $\Delta E \sim C_{\rm V} \longrightarrow M \sim \sqrt{N^{1+\alpha/d\nu}}$
  - Note: Systems for which $C_{\rm V}|_{T\to 0} \to 0$ require many temperatures.
  - In principle, we need as many temperatures such that the method traverses the energy landscape. Measure? *Acceptance probabilities*.

Measuring acceptance probabilities

- Definition:

$$A = \frac{N_{\rm accept}}{N_{\rm trial}}$$

- Traditional wisdom: Tune the temperature set such that...
  - ... 0.2 ≤ *A* ≤ 0.9.
  - ... *A* is approximately independent of temperature.



  - Detailed implementation which gives flat acceptance rates: Incomplete beta function law [uses $A = f(C_{\rm V})$].

Predescu *et al.*, JSTAT (03)

- Notes:
  - A quick run (no need to equilibrate) will immediately produce stable acceptance rates (easy tuning by hand).
  - It has been claimed that *A* ~ 0.3 is optimal.

Rathore *et al.*, J. Chem. Phys. (05)

Practical approach when C_V ~ const.

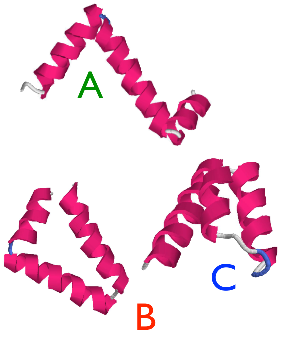
- Geometric progression:
  - Works well when C_V ~ const (like in spin glasses).
  - Iteratively construct a temperature set and tune *M* with λ.

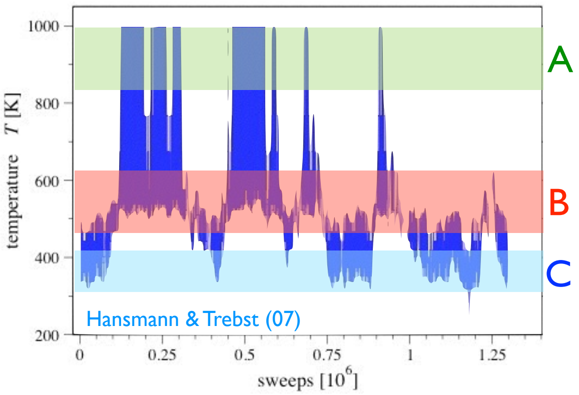
$$\frac{1}{T_i} = \lambda R^{i-1}\frac{1}{T_{\rm min}} \qquad R = \left[\frac{T_{\rm min}}{T_{\rm max}}\right]^{1/(M-1)}$$

- By hand:
  - If C_V diverges strongly, start from a geometric progression.
  - Interlace extra temperatures by hand.
  - Tedious, but can be automatized.
- What if C_v diverges? Katzgraber *et al.*, JSTAT (06)
  - Optimize the diffusion of temperatures to overcome bottlenecks.
  - Replicas should do a random walk in temperature space.

Example: Protein

- Protein A:





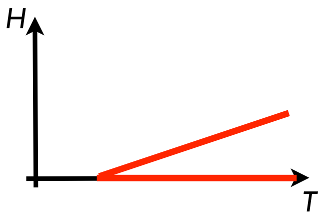
Hansmann & Trebst (07)

- It can happen that the replicas will only move in subspaces of the phase space (A, B, C) using traditional temperature schemes.
- Feedback optimization helps overcome these bottlenecks.

Katzgraber *et al.*, JSTAT (06)

Possible extensions and adaptations

- Any control variable can be used:
  - Field
  - Temperature *and* field
  - Coupling constants in QCD
  - Frequencies (e.g., in a Holstein model)
  - ...



- Combinations with other algorithms possible:
  - Tempering Monte Carlo molecular dynamics (biomolecules).
  - Tempering quantum Monte Carlo (quantum spin glasses).
  - Bayensian periodigrams (planet search in star systems).
  - Iterative search methods (combinatorial problems).
  - Cluster exchange Monte Carlo (diluted spin glasses).
  - Parallel tempering Wang-Landau sampling (biomolecules).

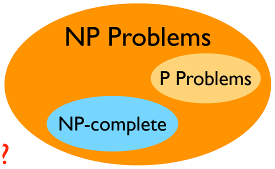
Selected optimization methods:

Monte Carlo based
Evolutionary
Quantum

P versus NP (non-rigorous definitions)

P = NP?

- P (“polynomial”)
  - All decision problems (YES/NO) that can be solved on a deterministic sequential machine in an amount of time polynomial in the input.
- NP (“nondeterministic polynomial”)
  - All decision problems for which the correctness of a *guessed* solution can be verified in polynomial time.
  - NP problems scale worse than any polynomial.
  - Examples: 3-SAT, spin glasses, NPP, …
  - NP-complete: hardest problems in NP.



- How do we show that a problem is either P or NP?
  - Easiest way is to find a *polynomial* mapping to a known problem.

Is P always tractable?

- While in theory P is easier than NP, in practice this is not always true:
  - Pre-factors are ignored when assessing algorithms: $T(N) \sim 10^{10000} N$ is intractable.
  - Exponent size is ignored: $T(N) \sim N^{10000}$ is intractable.
  - Worst-case scenarios ignored: 99.999% P, but 0.001% NP.
- While NP, in general, is harder to solve, there are exceptions:
  - Only *deterministic* solutions considered: The problem might be solved quickly, but with a small error probability.
  - Quantum computers might help solve problems known not to be in P. Example: D-Wave quantum annealer.

Exact versus heuristic

- Exact:
  - The algorithm delivers the exact ground state, *guaranteed*.
  - Difficult task: How can one *prove* this is the true optimum?
  - Often not practical.
  - Examples: Branch & Cut algorithm, exhaustive search, …

- Heuristic:
  - The algorithm *might* deliver the exact ground state.
  - Most algorithms deliver the optimum with high probability.
  - In general, one obtains a good estimate for the optimum.
  - Examples: genetic algorithms, simulated annealing, …

our focus here…

- Note: For most practical purposes heuristic algorithms are enough.

Thermal optimization:

Simulated annealing

Simulated Annealing (SA)

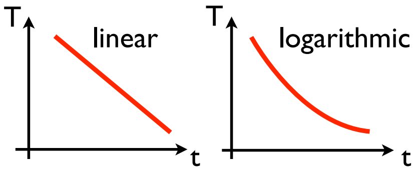
Kirkpatrick *et al.*, Science (83)

- Inspired by cooling of a crystal to avoid defects.
- Stochastically sample the *cost* function $\mathcal{H}(\{S\})$ to obtain a stationary state described by the Boltzmann distribution.
- Once the system is in thermal equilibrium, cool slowly and iterate.
- Typical cooling protocols: $T(t) = a/(b + \log t)$ logarithmic

$T(t) = a - bt$ linear (0.1≤b≤0.2)

$T(t) = a\exp(-bt)$ exp (0.8≤b≤0.99)

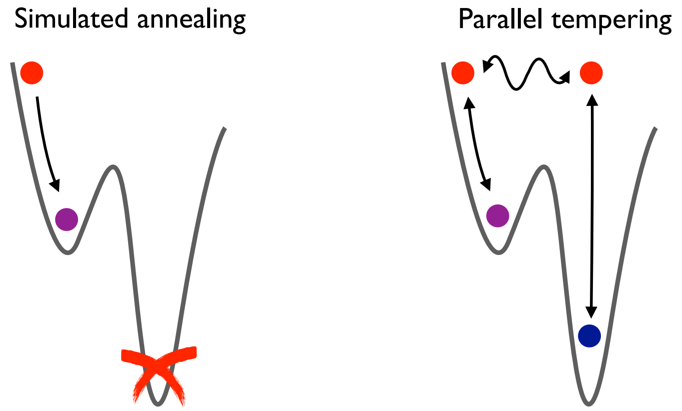
```
ALGORITHM (sim anneal):
choose configuration {$S_i$}
for i = 1 ...  $t_{max}$;
  set temperature $T(t)$;
  MC run until equilibrium;
  [store best solution];
done
return $E$ = $H(\{S_i\})$
```



- The slower the cooling, the better. Infinitely slow will find true optimum.
- Many applications across disciplines (bio, TSP, NPP, …).

Geman & Geman

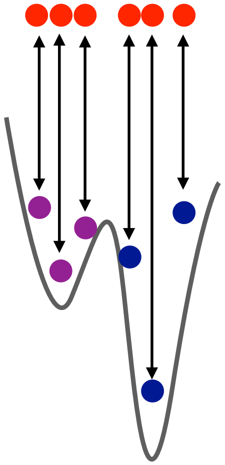
When does simulated annealing fail?



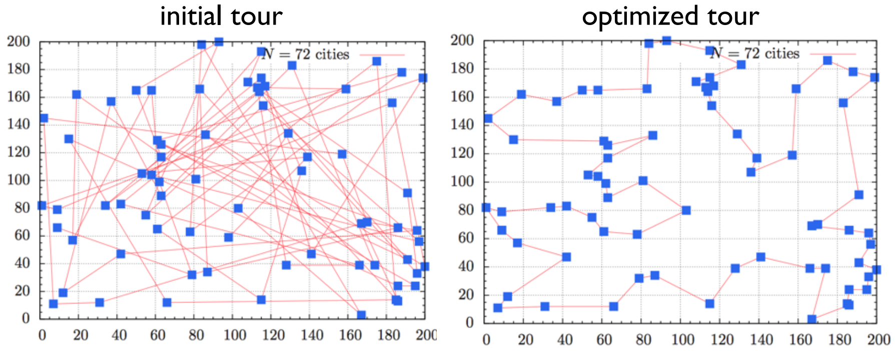
- Simulated annealing is a one-way optimization.
- If the energy landscape is rough, it might get stuck in metastable states.

Improving simulated annealing by repetition

- Empirical observation:
  - The algorithm getting stuck is only slightly dependent on the annealing speed/schedule.
  - Initial conditions can strongly affect the performance.
- Simple solution:
  - Repeat the sampling many times with different initial conditions / Markov chains.
  - Keep track of the states obtained. The distribution of low-lying states should give a clear indication of the ground state.
- Alternative: Use parallel tempering.



Simulated Annealing applied to the TSP



- 72 city tour with random initialization (tour length ~ 7000 steps).
- Optimization with a linear schedule, quenched to $T = 0$.
- Optimal tour (approximately 5 minutes run time) ~ 1320 steps.

Optimization using parallel tempering

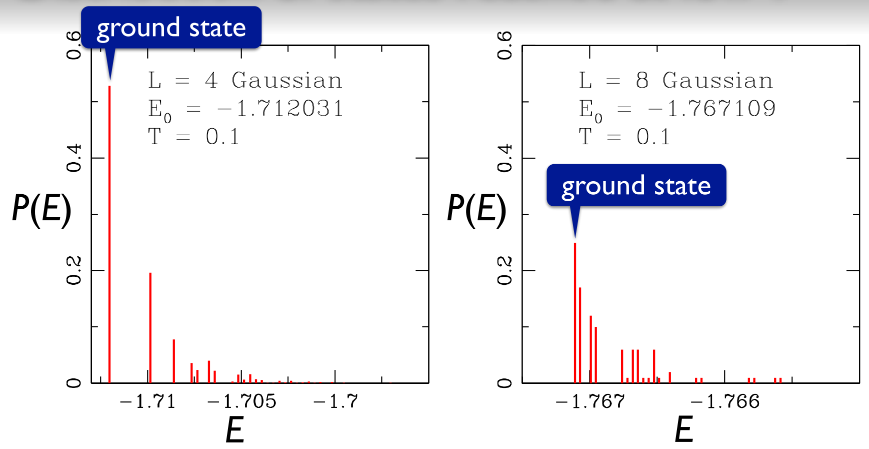
Parallel tempering ground-state searches

- Outline of the approach: Moreno *et al.* (03)
  - Perform a parallel tempering (PT) simulation with T_{min} close to zero typically $T_{\text{min}} \sim 0.1 T_c$.
  - Simulate *two* copies of the system with *different* Markov chains.
  - Run the simulation for time t_{eq} until the system is in equilibrium.
  - During an additional $t_{\text{eq}}/4$, repeat:
    - Before each PT move record the energy (and spin configuration) for the lowest T values if the energies match in both copies.
    - If a lower energy is found, replace the recorded value.

not a must

- Performance of the method:
  - Works best for short-range systems.
  - For intermediate system sizes (up to 500 spins) ≥ 99% accuracy!

Distribution of states reached at low T

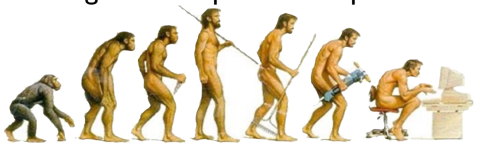


- Data for a 3D Gaussian spin glass instance sampled 10^3 times.
- The ground state is the most populated state, even for ~ 500 spins!

Genetic algorithms

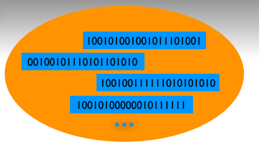
Motivation

- Basic idea:
  - Mimic nature by generating a "*population*" of possible solutions.
  - Evolve the population according to some problem-dependent rules.
  - Survival of the fittest…

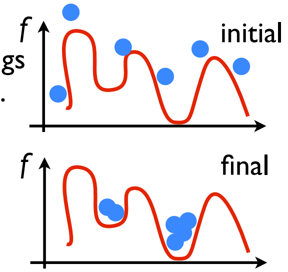


- Application domain:
  - Optimization problems with rough energy landscapes: Using a population allows the algorithm overcome barriers.
  - Outside physics: scheduling, protein ligand docking, code cracking, TSP, model selection, compiler flag optimization.
  - In Physics: statistical mechanics problems, X-ray data analysis, geological data reconstruction, general optimization…
- Note: Here we follow closely the book of Hartmann & Rieger.

Necessary ingredients for a GA



- Population of solutions:
  - Needs to be in a "genetic representation."
  - The larger the population, the better the chance to find a solution.
  - However, the larger, the longer the search could take.
  - *Example:* Minimize of a function $f(x)$
    - Represent the pool of minima as bit strings where mutations are easily accomplished.
    - The initial population is a list with random-bit arrays.



- Fitness function:
  - We need a measure of the quality of a candidate solution.
  - *Example:* Hamiltonian of a physical problem, or for above $f(x)$.

Operations on the population

- Evolution of the population:
  - To converge to a solution, *cheap* operations that randomize the population need to be performed.
- Mutations:
  - Randomly change bits with a (small) probability *p*.

00100101010 → 00100001010

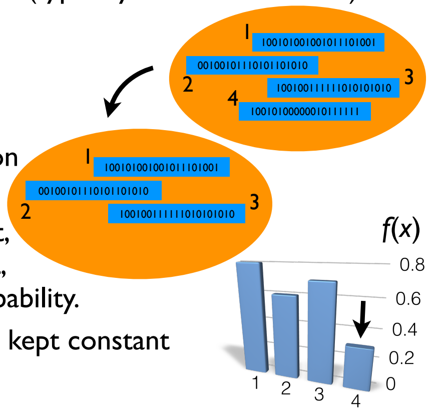
- Crossover:
  - Generate "offsprings" from a set of "parents."

00100101001
+
00010100111
→ 0010 0100111

- Note: In principle, any operation is possible. Only few are good…

Crossover & Natural selection

- Crossover operation:
  - There are many ways, e.g., sequence splitting, keep the fittest, …
  - The number of parents can be varied (typically 2).
  - Randomizes better than mutations (typically called more often).
- Natural selection (fitness testing):
  - Very much problem dependent.
  - Evaluate the fitness of the solution and only keep the fittest.
  - Different schemes: kill 50% worst, kill offspring if worse than parent, kill with a fitness-dependent probability.
  - The population can be shrunk or kept constant (cloning of the fittest).



Pseudocode & Final considerations

- Note:
  - The method is heuristic, and often does not deliver good results.
  - GAs should be combined with other local optimizers to improve results.
  - Recommended to start with a large population that is culled.
- Advantage:
  - Straightforward to implement.
- Disadvantage:
  - Many parameters must be tuned, fitness functions often not available.
  - The choice of parameters/operations depends on the problem.

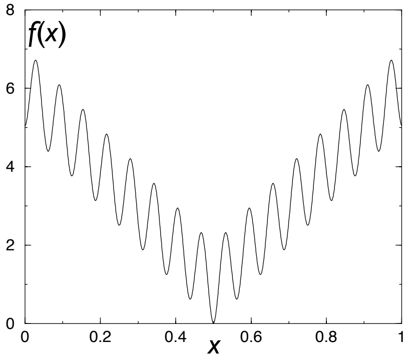
```
ALGORITHM (generic genetic):

Initialize populations $x_1, \ldots x_M$

for $t = 1 \ldots N_{iter}$
    choose a set of parents $\{x_i\}$;
    create offsprings via crossover;
    mutate;
    [local optimization;]
    calculate fitness;
    update population with offsprings;
done
return best individuals from $x_1, \ldots x_M$
```

Case study: function minimization

- Goal:
  - Find the minimum of $f(x) = 10|x-0.5| - \cos(100(x-0.5)) + 1$ in the interval $x \in [0,1]$.
  - Note: this is an academic example because we know $x_0 = 0.5$.
  - We represent $x \in \mathbb{R}$ as bit strings with precision *P*:

$$x_i = \sum_{j=1}^{P} 2^{-j} x_i^j \qquad x_i^j \in \{0,1\}$$

- Steps:
  - Represent numbers as genes.
  - Mutation/Crossover operations.
  - Evolve the population.



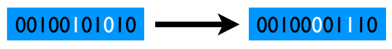
Hartmann & Rieger

Representation of numbers & mutations

- Converting floats to bit sequences:
  - This can be efficiently done with the code snippet on the right.

```
ROUTINE bit_seq(x,P)

f = 0.5;
for q = 1 ... P;
do
    if(x ≥ f)
    then
        $x_q$ = 1;
        x = x - f;
    else
        $x_q$ = 0;
        f = f/2.0;
    done;
done;
return ($x_1$,$x_2$, ...., $x_P$);
```

- Mutations:



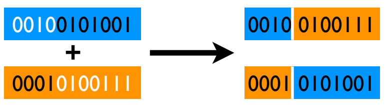
  - Flip bits with a probability p.
  - Note that this can also be biased to work for low or high bits.

```
ROUTINE mutate(x)

for q = 1 ... P;
do
    r = rand(0.,1.)
    if(r < p){
        $x_q$ = 1 - $x_q$;
    done;
done;
return ($x_1$,$x_2$, ...., $x_P$);
```

Crossover

- Details:
  - Select two parents.
  - Select a splicing bit position s.
  - Generate two offsprings.

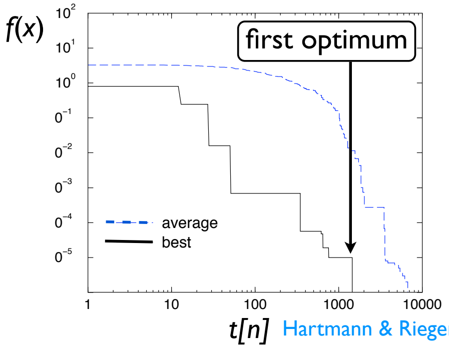
```
ROUTINE crossover($x_1$,$x_2$)

s = rand(1,P)
for q = 1 ... s do
    $c_1^q$ = $x_1^q$;
    $c_2^q$ = $x_2^q$;
done;
for q = s+1 ... P do
    $c_1^q$ = $x_2^q$;
    $c_2^q$ = $x_1^q$;
done;
return ($c_1$,$c_2$);
```



- Note:
  - It is best to select a random crossover point to improve randomization.
  - One could also use three parents, where one plays the role of a "mask" used to select the parental bits for the offspring.

Putting it all together…

- Natural selection: An offspring that has better fitness than the parent automatically replaces the parent.
- Example run:
  - Start with M = 50 genes.
  - Mutation probability: p = 0.1.

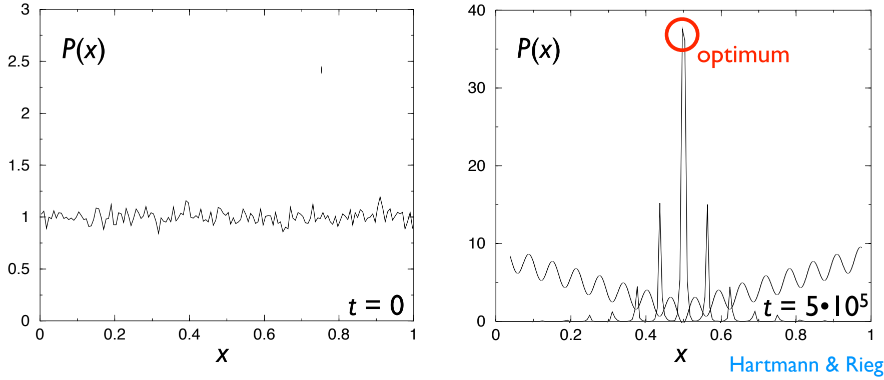


Hartmann & Rieger

```
ALGORITHM genetic(f)
begin
    initialize M strings of length P.
    for t = 1... n do
        choose parents i and j randomly in [1,M];
        ($c_1$,$c_2$) = crossover($x_1$,$x_2$);
        mutation($c_1$,p);
        mutation($c_2$,p);
        if(f($c_1$) < f($x_1$)) then
            $x_1$ = $c_1$;
        done;
        if(f($c_2$) < f($x_2$)) then
            $x_2$ = $c_2$;
        done;
    done;
    return best individual from ($x_1$, ... $x_M$)
end;
```

Tracking the evolution of the population

- Example histogram of genetic population with M = 5000.



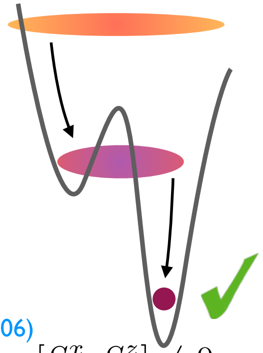
Hartmann & Rieger

- The initial population at t = 0 is random, i.e., $P(x)$ ~ 1 for all x.
- After t = $5{\cdot}10^5$ iterations the population relaxes into the local minima with a large fraction settling for the true optimum.

Quantum enhanced optimization

Quantum Annealing

Kadowaki & Nishimori (98)
Farhi *et al.* (00)

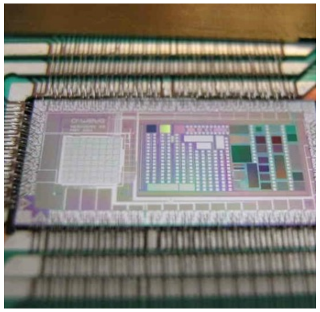


- Idea:
  - Use quantum tunneling & fluctuations.
  - Like SA, but quenches quantum fluctuations.
- Theoretical advantages over SA:
  - Not limited to a local search.
  - Fluctuations determine the "tunneling radius."
- Implementation in DW2: Morita & Nishimori (06)
  - Apply a transverse field that does not commute: $[S^x, S^z] \neq 0$

$$\mathcal{H}(S_i) = \sum_{i\neq j}^{N} Q_{ij} S_i S_j \longrightarrow \mathcal{H}(S_i) = \sum_{i\neq j}^{N} Q_{ij} S_i^z S_j^z - D \sum_{i}^{N} S_i^x$$

  - Reduce quantum fluctuations via a linear protocol $D(t) = a - bt$.

Is this method of general interest? Yes!



Washington chip

Cryogenic mount

D-Wave 2X @ NASA

- First quantum annealing machines with ~1000 qubits.
- Based on *programmable* superconducting flux qubits.
- Currently, large controversy on its speed & quantumness.

currently part of my research.

Final considerations

Final considerations & further methods

- How can we improve optimization algorithms?
    - Tailored *combinations* of algorithms tend to work better.
    - Developing new & efficient algorithms is the holy grail in this field.
    - There are more efficient methods. However, these are very complex.

- Other selected methods:
    - Improved Extremal Optimization (heuristic).
    - Hysteretic Optimization (heuristic, works for high connectivity).
    - Patchwork Dynamics (heuristic, helps with nonplanar graphs).
    - Max-flow methods (heuristic, ideal for random-field models).
    - Matching algorithms (heuristic, planar frustrated systems).
    - Branch & Bound (exact, only small instances tractable).
    - Population Annealing (sequential Monte Carlo, very fast, new SOA?).

HOW'S YOUR QUANTUM COMPUTER PROJECT COMING ALONG?
GREAT!
THE PROJECT EXISTS IN A SIMULTANEOUS STATE OF BEING BOTH TOTALLY SUCCESSFUL AND COMPLETELY UNSUCCESSFUL
© dilbert.com
CAN I OBSERVE IT?
THAT'S A TRICKY QUESTION.

Thank you!

hgk@tamu.edu