


1

Outline

- Optimization
- Variational Monte Carlo
- Stochastic search and optimization
- Random search
- Simulated annealing
- Parallel tempering



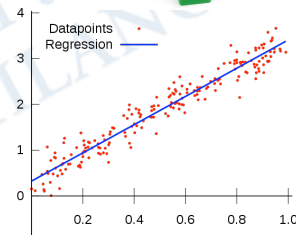
UNIVERSITÀ
DEGLI STUDI
DI MILANO

2

2

Optimization

- **Optimization problems** are very common in everyday life and mathematical techniques of search and optimization are aimed at providing a formal means for making the **best decisions** in many real situations.
- For example frequently **optimization is used as a tool**: when a function with various parameters is fitted onto experimental data points, then one searches for the parameters which lead to the **best fit**.
- During the last decades **many problems** in physics, and not only in physics, have turned out to be in fact optimization problems, or **can be transformed into optimization problems**
- In the simplest sense, an optimization can be considered as a **minimization or maximization** problem. E.g. the function $f(x)=x^2$ has a minimum $f_{\min}=0$ at $x=0$. In general we can use the first derivative to determine the potential locations, and use the second derivative to verify if the solution is a maximum or minimum. However, for nonlinear, multimodal, multivariate functions, this is not an easy task.

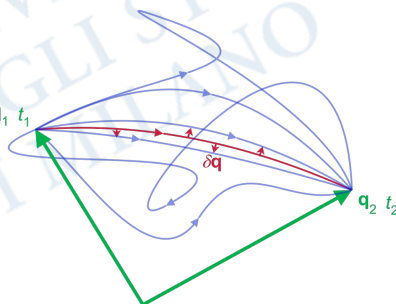


3

Minimization principles

Note also that in Physics many systems are governed by **minimization principles**.

- In thermodynamics, a system coupled to a reservoir always equilibrate towards the state with **minimal free energy**
- The **principle of least action** is a variational principle that, when applied to the action of a mechanical system, can be used to obtain the **equations of motion**. This principle can be used even q_1, t_1 in general relativity where the Einstein-Hilbert action yields the Einstein field equations. Schwinger and Feynman, demonstrated how this principle can also be used in quantum calculations using path integration
- We have seen that in calculating the ground state behaviour of quantum systems, quite often a **variational approach** is applied: the energy of a test state vector is minimized with respect, to some parameters



4

Quantum Monte Carlo: Variational MC



- Monte Carlo simulation of quantum many-body systems dates from McMillan's observation (1965) that the integrals required to evaluate expectations, including the total energy, can be carried out by using the Metropolis algorithm
- The **variational** method has proved to be a very useful way of computing (approximate) ground-state properties of many-body systems.
- Conceptually it is quite simple; the **variational principle** tells us that for any function ψ_T , the variational energy E_T , defined as

$$E_T = \frac{\langle \psi_T | \hat{H} | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle} = \frac{\int d\vec{r}_1 \cdots d\vec{r}_N \psi_T^*(\vec{r}_1 \cdots \vec{r}_N) \hat{H} \psi_T(\vec{r}_1 \cdots \vec{r}_N)}{\int d\vec{r}_1 \cdots d\vec{r}_N |\psi_T(\vec{r}_1 \cdots \vec{r}_N)|^2} \geq E_0 = \frac{\langle \psi_0 | \hat{H} | \psi_0 \rangle}{\langle \psi_0 | \psi_0 \rangle}$$

will be a **minimum** when ψ_T is the **exact ground-state** solution (ψ_0) of the Schrödinger equation, being H the Hamiltonian of the system

- The method then consists in constructing a family of functions $\psi_T(\mathbf{a})$ and optimizing the vector parameter (\mathbf{a}) so that the previous energy functional is minimized for $\mathbf{a}=\mathbf{a}^*$

5

5

- The variational energy is a rigorous upper bound to the ground state energy and if the family of functions is well chosen, $\psi_T(\mathbf{a}^*)$ will be a **good approximation** to ψ_0 . Thus one can use $\psi_T(\mathbf{a}^*)$ to study the ground state properties of the system
- The task now is to evaluate the multidimensional integrals to get expectation values, in particular to calculate the variational energy E_T which should be minimized
- When the **Metropolis Monte Carlo method** is used to compute E_T and other expectation values, the method is called **Variational Monte Carlo (VMC)**:

$$\begin{aligned} E_T &= \frac{\int d\vec{r}_1 \cdots d\vec{r}_N \psi_T^*(\vec{r}_1 \cdots \vec{r}_N) \hat{H} \psi_T(\vec{r}_1 \cdots \vec{r}_N)}{\int d\vec{r}_1 \cdots d\vec{r}_N |\psi_T(\vec{r}_1 \cdots \vec{r}_N)|^2} = \frac{\int d\vec{r}_1 \cdots d\vec{r}_N \psi_T^*(\vec{r}_1 \cdots \vec{r}_N) \hat{H} \psi_T(\vec{r}_1 \cdots \vec{r}_N)}{\int d\vec{r}_1 \cdots d\vec{r}_N |\psi_T(\vec{r}_1 \cdots \vec{r}_N)|^2} = \\ &= \int d\vec{r}_1 \cdots d\vec{r}_N \frac{|\psi_T(\vec{r}_1 \cdots \vec{r}_N)|^2}{\int d\vec{r}_1 \cdots d\vec{r}_N |\psi_T(\vec{r}_1 \cdots \vec{r}_N)|^2} \frac{\hat{H} \psi_T(\vec{r}_1 \cdots \vec{r}_N)}{\psi_T(\vec{r}_1 \cdots \vec{r}_N)} = \int d\vec{r}_1 \cdots d\vec{r}_N p(\vec{r}_1 \cdots \vec{r}_N) E_{loc}(\vec{r}_1 \cdots \vec{r}_N) \end{aligned}$$

- Given a trial wave-function ψ_T (and given a model Hamiltonian) the evaluation of E_T is "exact", i.e., it gives the value of E_T within the statistical uncertainty of the Monte Carlo calculation

6

6

The optimization problem

7

- Let $\mathbf{x} = (x_1, \dots, x_n)$ be a vector with n elements which can take values from a domain $S(\equiv X^n)$: $x_i \in X$. The domain X can be either **discrete**, for instance $X = \{0,1\}$ or $X = \mathbb{Z}$ or X can be **continuous**, for instance $X = \mathbb{R}$
- A general optimization problem could be formalized as:

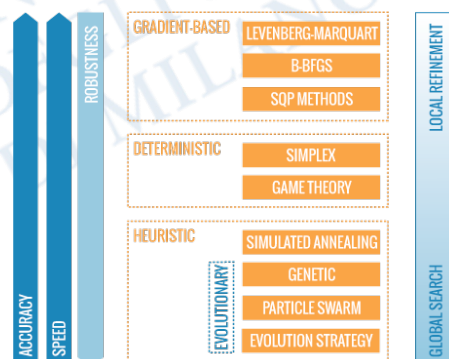
$$\begin{aligned} &\text{minimize } L_1(\mathbf{x}), \dots, L_I(\mathbf{x}), \mathbf{x}=(x_1, \dots, x_n), \\ &\text{subject to } h_j(\mathbf{x})=0, (j=1, \dots, J) \text{ and } g_k(\mathbf{x}) \leq 0, (k=1, \dots, K) \end{aligned}$$
- Where L_1, \dots, L_I are the **objectives or cost/loss functions**, while h_j and g_k are the **equality and inequality constraints**, respectively. Obviously, the simplest case of optimization is unconstrained function optimization. In the case when $I=1$, it is called **single-objective optimization** problem. When $I \geq 2$, it becomes a **multi-objective optimization** problem
- In general, all the functions L_i , h_j and g_k are nonlinear. In addition, some of the functions L_i may have discontinuities, and thus **derivative information cannot be obtained**. This may pose various challenges to the optimization procedure
- To solve an optimization problem, efficient search or **optimization algorithms** are needed. There are many optimization algorithms which can be classified in many ways, depending on the focus and characteristics

7

Classification

8

- If the derivative or gradient of a function is the focus, optimization can be classified into **gradient-based** algorithms and **derivative-free** or gradient-free algorithms. Gradient-based algorithms use derivative information, and they are often very efficient. Derivative-free algorithms use the values of the function itself.
- Optimization algorithms can also be classified as **deterministic** or **stochastic/heuristic**. If an algorithm works in a mechanical deterministic manner without any random nature, it is called **deterministic**. For such an algorithm, it will reach the **same final solution if we start with the same initial point**. On the other hand, if there is some randomness in the algorithm, the algorithm will usually reach a different point every time the algorithm is executed, even though the same initial point is used.



8

Optimization is huge!

9

- Optimization algorithms can be classified also into **trajectory-based** and **population-based**. A trajectory-based algorithm typically uses a single agent at a time, which will trace out a path towards a solution as the iterations continue. On the other hand, population-based algorithms use multiple agents which will interact and trace out multiple paths
- Search capability** can also be a basis for algorithm classification: **local search** and **global search** algorithms. Local search algorithms typically converge towards a local optimum, not necessarily (often not) the global optimum, and such an algorithm is often deterministic and has no ability to escape from local optima. On the other hand, for global optimization, local search algorithms are not suitable, and global search algorithms should be used. In essence, randomization is an efficient component for global search algorithms.
- Obviously, algorithms may not exactly fit into each category. It can be a so-called **mixed type or hybrid**, which is some combination of deterministic components with randomness, or combines one algorithm with another so as to design more efficient algorithms.

9

Example: the traveling salesman problem

10

A **combinatorial optimization** problem is an optimization problem, where an optimal solution has to be identified from a finite set of solutions

- The **traveling salesman problem (TSP)**: consider n cities distributed randomly in a plane. The optimization task is to **find the shortest round-tour through all cities which visits each city only once**. The tour stops at the city where it started.
- The problem is described by:

$$X = \{1, 2, \dots, n\} \quad L(\vec{x}) = \sum_{i=1}^n d(x_i, x_{i+1})$$
 where $d(x_i, x_{i+1})$ is the distance between cities x_i and x_{i+1} (and $x_{n+1} = x_1$).
- The optimum order of the cities for a TSP depends on their exact positions, i.e. on the random values of the distance matrix d , which represent an **instance** of the general optimization problem
- The **constraint** that every city is visited only once can be realized by constraining the vector \mathbf{x} to be a permutation of the sequence $[1, 2, \dots, n]$

10

- Because there is a finite number of possible paths, this is a **combinatorial** (discrete & finite) **optimization** problem, although one with a potentially very large number of elements
- A tour with $n \geq 3$ cities has $(n-1)!/2$ **possible unique tours** (Unique here refers to fundamentally different ordering. For example, if $n = 3$, the tour 1-2-3-1 is considered equivalent to 1-3-2-1 by the symmetry of the cost between cities, where the indicated numbers 1, 2, or 3 represent the labels for the three cities).
- The TSP also belongs to a class of minimization problems for which the objective function L has **many local minima**.
- In practical cases, it is often enough to be able to choose from these a minimum which, even if not absolute, cannot be significantly improved upon. If one absolutely wants to get **the global optimum** for n very large, the only way is to let the computer run for a very, very long time
- In the language of combinatorial optimization, the problem is proved to be **NP-hard**. **How to solve such problems?** There is **no universal method**; when the number of variables n is large, **computers are used to obtain a solution**.

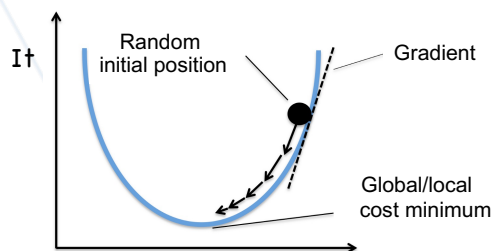
11

Gradient descent

- **Gradient descent (steepest descent)** is the simplest deterministic gradient-based iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, **one takes steps proportional to the negative of the gradient of the function at the current point**
- If, instead, one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as **gradient ascent**.
- Gradient descent is based on the observation that if the multi-variable cost function $L(\mathbf{x})$ is defined and **differentiable** in a neighborhood of a point \mathbf{x} then $L(\mathbf{x})$ **decreases fastest if one goes from \mathbf{x} in the direction of the negative gradient of L at \mathbf{x} , $-\nabla L(\mathbf{x})$** . follows that, if

$$\bar{\mathbf{x}}_{n+1} = \bar{\mathbf{x}}_n - \gamma \nabla L(\bar{\mathbf{x}}_n)$$

for $\gamma > 0$ small enough, then $L(\mathbf{x}_n) \geq L(\mathbf{x}_{n+1})$.

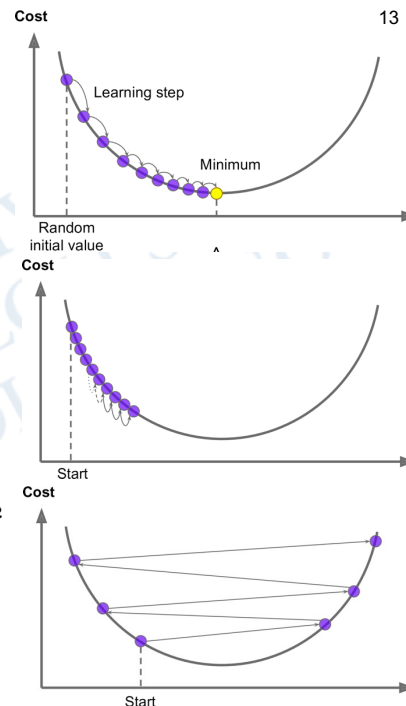


12

- With this observation in mind, one starts with a guess \mathbf{x}_0 for a local minimum of L , and considers the sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ such that

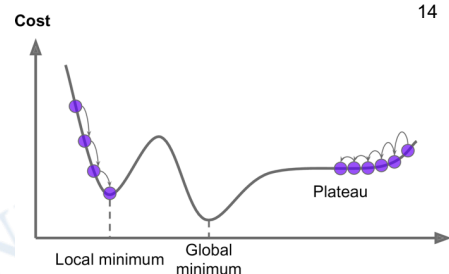
$$\bar{\mathbf{x}}_{n+1} = \bar{\mathbf{x}}_n - \gamma \vec{\nabla} L(\bar{\mathbf{x}}_n) \quad n \geq 0$$

- We obtain a monotonic sequence $L(\mathbf{x}_0) \geq L(\mathbf{x}_1) \geq L(\mathbf{x}_2) \geq \dots$ so hopefully the sequence $\{\mathbf{x}_n\}$ converges to the desired local minimum.
- An important parameter is the size of the steps, determined by the **learning rate hyperparameter γ** . If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time
- On the other hand, if the learning rate is too high, you might jump across the valley and end up on the other side, possibly even higher up than you were before.



13

- Finally, **not all cost functions look like nice regular bowls**. There may be holes, ridges, plateaus, and all sorts of irregular terrains, making convergence to the minimum very difficult. The figure shows the two main challenges with gradient descent: if the random initialization starts the algorithm on the left, then it will **converge to a local minimum**, which is not as good as the global minimum. If it starts on the right, then it will take a **very long time to cross the plateau**, and if you stop too early you will never reach the global minimum.



- Convergence to a local minimum can be guaranteed. When the function L is **convex**, all local minima are also global minima, so in this case gradient descent can converge to the global solution.
- In many situation, however, the optimization is **multi-objective** and one can build a single cost/loss function by adding the whole set of objectives to obtain the **sum-cost/loss function**:

$$L(\vec{\mathbf{x}}) = \sum_{i=1}^n L_i(\vec{\mathbf{x}})$$

14

Stochastic gradient descent

15

- When used to minimize the **sum-cost/loss function**, a standard (in this case usually called "**batch**" / group / set) gradient descent (**BGD**) method would perform the following iterations :

$$\vec{x}_{n+1} = \vec{x}_n - \gamma \vec{\nabla} L(\vec{x}_n) = \vec{x}_n - \gamma \sum_{i=1}^I \vec{\nabla} L_i(\vec{x}_n) \quad n \geq 0$$

where γ is the learning rate.

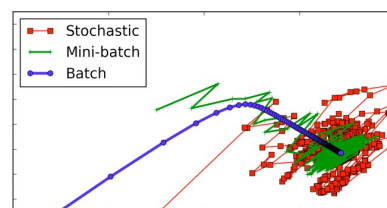
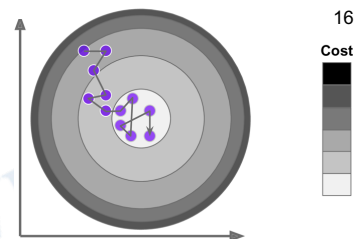
- In many cases, the summand functions have a simple form that enables inexpensive evaluations of the sum-cost function and the sum gradient. However, in other cases, **evaluating the sum-gradient may require expensive evaluations** of the gradients from all summand functions L_i .
- At the opposite extreme, **stochastic gradient descent (SGD)** just **picks a random instance in the cost function set at every step and computes the gradients based only on that single instance**. Obviously this makes the algorithm much faster since it has very little data to manipulate at every iteration.
- Moreover, when the cost function is very irregular, this can actually help the algorithm **jump out of local minima**, so SGD has a better chance of finding the global minimum than BGD does.

15

- On the other hand, due to its stochastic (i.e., random) nature, this algorithm is much less regular than batch gradient descent: instead of gently decreasing until it reaches the minimum, the cost function with SGD will bounce up and down, decreasing only on average.
- To improve on this feature, at each step, instead of computing the gradients based on the full training set (as in Batch GD) or based on just one instance (as in Stochastic GD), **Mini-batch SGD computes** the gradients on small random sets of instances called **mini-batches**.

$$\vec{x}_{n+1} = \vec{x}_n - \gamma \vec{\nabla} L(\vec{x}_n) = \vec{x}_n - \gamma \sum_{i \in \text{Random}(M)} \vec{\nabla} L_i(\vec{x}_n)$$

- The algorithm's progress in parameter space is less erratic than with SGD. As a result, Mini-batch SGD will end up walking around a bit closer to the minimum than SGD. But, on the other hand, it may be harder for it to escape from local minima



16

Stochastic search and optimization

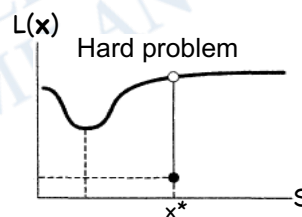
- Given the difficulties in many real-world problems and the inherent uncertainty in information that may be available for carrying out the task, stochastic search and optimization methods have been playing a rapidly growing role.
- DEFINITION:** Stochastic optimization methods are optimization algorithms which incorporate probabilistic elements, either in the problem data (the objective function, the constraints, etc.), or in the algorithm itself (through random parameter values, random choices, etc.), or in both. The concept contrasts with the deterministic optimization methods, where the values of the objective function are assumed to be exact, and the computation is completely determined by the values obtained so far.
- The two main problems of interest are:
 - Find the value(s) of $\mathbf{x} \in S = \mathbb{X}^n$ that minimize a scalar-valued loss function $L(\mathbf{x})$
 - Find the value(s) of $\mathbf{x} \in S = \mathbb{X}^n$ that solve the equation $g(\mathbf{x})=0$ for some function $g(\mathbf{x})$
- The second problem can be converted directly into an optimization problem by noting that a \mathbf{x} such that $g(\mathbf{x})=0$ is equivalent to a \mathbf{x} such that $L(\mathbf{x})=\|g(\mathbf{x})\|^2$ is minimized for any vector norm $\|\cdot\|$

17

17

Finding a global minimum?

- One of the major distinctions in optimization is between global and local optimization. In practice, however, a global solution may not be available and one must be satisfied with obtaining a local solution.
- For example, $L(\mathbf{x})$ may be shaped such that there is a clearly defined minimum point over a broad region of the domain S , while there is a very narrow spike at a distant point...
- This example illustrates that, in general, one cannot be guaranteed of ever obtaining a global solution. Without prior knowledge about the possible existence of such a point, no typical algorithm would be able to find this solution because there is nothing near \mathbf{x}^* to indicate the presence of a minimum.
- Because of the inherent limitations of the vast majority of optimization algorithms, it is usually only possible to ensure that an algorithm will approach a local minimum with a finite amount of resources being put into the optimization process.



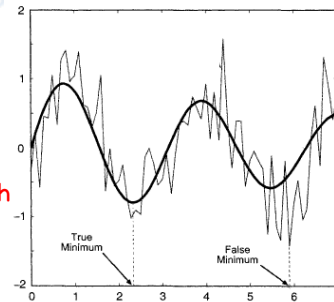
18

18

Stochastic data or procedures

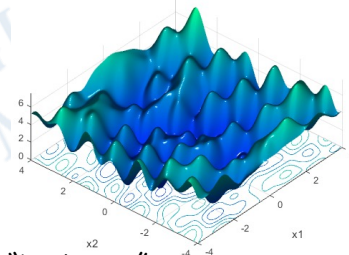
19

- However, since the local minimum may still yield a significantly improved solution (relative to no formal optimization process at all), **the local minimum may be a fully acceptable solution** for the resources available (human time, money, computer time, etc.) to be spent on the optimization. However, we will also consider some algorithms (random search, simulated annealing, genetic algorithms, etc.) that are in principle able to find global solutions from among multiple local solutions.
- Stochastic search and optimization applies ineluctably where there is random **noise in the measurement** of $L(\mathbf{x})$ or $g(\mathbf{x})$
- Partly-random input data arise in situation like simulation-based optimization where **Monte Carlo simulations** are run as estimates of an actual system, and problems where there is experimental **error in the measurements** of the criterion.
Note that in the presence of noise the search for a global minimum could have no meaning at all!



19

- In such cases, knowledge that the function values are contaminated by random "noise" leads naturally to algorithms that use **statistical inference tools** to estimate the "true" values of the function and/or make statistically optimal decisions about the next steps.
- On the other hand, even when the data is exact, it is sometimes beneficial to deliberately **introduce randomness into the search process as a means of speeding convergence** and making the algorithm less sensitive to modelling errors.
- Further, the injected **randomness may provide the necessary impetus to move away from a local solution** when searching for a global optimum. Think about a multidimensional space S full of local minima; any kind of deterministic optimization procedure would be unable to explore all this complex "landscape".
- Indeed, this randomization principle is known to be a simple and effective way to obtain algorithms with almost certain good performance uniformly across all data sets, for all sorts of problems.



20

20

Random search & No free lunch theorem

- **Random search** simply consists in **picking up random potential solutions and evaluating them**. The best solution over a number of samples is the result of random search.
- In general there is a **competition between algorithm efficiency and algorithm robustness**. In essence, algorithms that are designed to be very efficient on one type of problem tend to be not reliably transferred to problems of a different type.
- The **No Free Lunch theorem** (Wolpert & Macready, 1995) states that **no search algorithm is better than a random search on the space of all possible problems**. In other words, if a particular algorithm does better than a random search on a particular type of problem, it will not perform as well on another type of problem, so that all in all, its global performance on the space of all possible problems is equivalent to a random search
- Hence, **there can never be a universally best search algorithm**. One must consider the characteristics of the problem together with the goals of the search and the resources available (computing power, human analysis time, etc.) in choosing an approach.

21

21

Metaheuristics

- **Algorithms with stochastic components** were often referred to as **heuristic** in the past, though the recent literature tends to refer to them as metaheuristics.
- We will follow the convention to call **all modern nature-inspired algorithms metaheuristics**. Loosely speaking, heuristic means to find or to discover by trial and error. Here meta- means beyond or higher level, and metaheuristics generally perform better than simple heuristics.
- Two major components of many metaheuristic algorithms are: **intensification** and **diversification**. Diversification means to generate diverse solutions so as to explore the search space on a global scale, while intensification means to focus the search in a local region knowing that a current good solution is found in this region. A good combination of these two major components will usually ensure that global optimality is achievable
- **Examples of metaheuristic algorithms** are: **simulated annealing, parallel tempering, genetic algorithms, memetic algorithms, tabu search, ant colony optimization, particle swarm optimization** etc. we will discuss simulated annealing, parallel tempering and genetic algorithms

22

22

Optimization: Simulated annealing

23

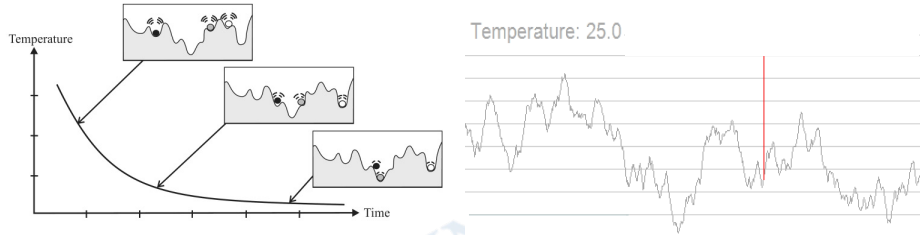
- At the heart of the method of **simulated annealing** is an analogy with thermodynamics, specifically with the way that liquids freeze and crystallize, or metals cool and anneal.
- At high T , the molecules of a liquid move freely with respect to one another. If the liquid is cooled slowly, thermal mobility is lost. The atoms are often able to line themselves up and form a pure crystal that is completely ordered over a distance up to billions of times the size of an individual atom in all directions.
- This **crystal is the state of minimum energy** for this system. The amazing fact is that, **for slowly cooled systems, nature is able to find this minimum energy state**. In fact, if a liquid is cooled quickly it does not reach this state but rather ends up in a polycrystalline or amorphous state having somewhat higher energy.
- So the **essence of the process is slow cooling**, allowing enough time for redistribution of the atoms as they lose mobility. This is the technical definition of annealing, and it is essential for ensuring that a low energy state will be achieved

23

- Any optimization problem can be 'embedded' in a statistical-mechanics/annealing problem. The idea is to interpret the cost function $L(x)$, $x \in X^n$, as the energy of a statistical-mechanics system and consider the Boltzmann distribution $p(x) = \exp[-\beta L(x)]/Z$
- In the low-temperature limit $\beta \rightarrow \infty$, the distribution becomes concentrated on the minima of $L(x)$, and the original optimization setting is recovered.
- Since the **Monte Carlo method provides a general technique for sampling from the Boltzmann distribution**, one may wonder whether it can be used, in the $\beta \rightarrow \infty$ limit, as an optimization technique.
- The idea of **simulated annealing** consists in letting β vary with time. More precisely, one decides on an **annealing schedule** $\{(\beta_1, n_1); (\beta_2, n_2); \dots (\beta_N, n_N)\}$, with inverse temperatures $\beta_i \in (0, \infty)$ and integers $n_i > 0$.
- The algorithm is initialized on a configuration x_0 and executes n_1 Monte Carlo steps at temperature $1/\beta_1$, n_2 steps at temperature $1/\beta_2$, ..., and n_N steps at temperature β_N . **The final configuration of each cycle i (with $i = 1, \dots, N-1$) is used as the initial configuration of the next cycle.**

24

24



- Mathematically, such a process is a **time-dependent Markov chain**.
- Within a specific temperature $t=1/\beta$ being in the configuration \mathbf{x} with "energy" $L(\mathbf{x})$ one generates a new configuration \mathbf{x}' with energy $L(\mathbf{x}')$ which replaces the original configuration with probability:

$$p = \begin{cases} e^{-\beta(L(\mathbf{x}')-L(\mathbf{x}))} & \text{if } L(\mathbf{x}') > L(\mathbf{x}) \\ 1 & \text{otherwise} \end{cases}$$
 where β is the fictitious inverse temperature.

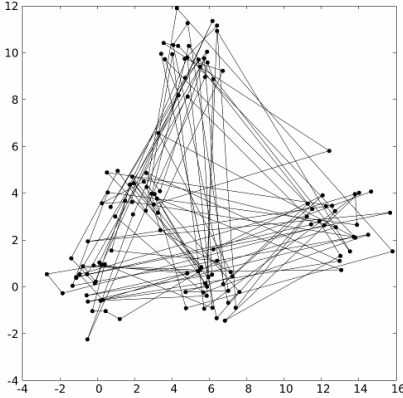
this can be easily sampled with a **Metropolis algorithm**.

25

25

- At any given temperature such an (ergodic) Monte-Carlo process samples the configurations according to their thermodynamic probability. Thus, **at high temperature moves with or against the gradient are accepted with almost equal probability**. **At low temperature only downhill moves are accepted**.
- In simulated annealing one thus starts with high temperature simulation and gradually cools the system to zero temperature.** If ergodicity is not lost during the cooling schedule, the simulation will stop in the global minimum of the energy landscape.
- Usually, due to the finiteness of the simulation time, simulated annealing allows us to obtain configurations **close to the ground states**, but not true ground states. This is however sufficient for most applications.

Simulated annealing and TSP
E= 852 T=125



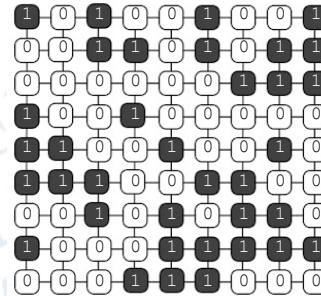
26

26

Spin models as a computational tool

27

- Previously introduced spin models can also turn out to become a useful **computational tool**
- Consider a particular **spin configuration** of an **Ising model**; given such a configuration, like any other, we can think about it as the **ground state of specific class of** (presently unknown/unexpressed) **Hamiltonians**
- We can also associate a 1 to each up spin and a zero to each down spin
- From the point of view of information theory, considering spin as a **binary alphabet**, the given configuration could represent everything, from a number to one or more phrases.
- If this number (or anything else) is the **solution** of a problem we can try to solve our problem in two steps:
 1. Solve an **embedding problem**, i.e. find one Hamiltonian for which this configuration is a ground state (not necessarily an easy task!)
 2. **Annealing**: let your spin model to **evolve toward its ground state** (for example reducing its "temperature") and ... read the solution!



27

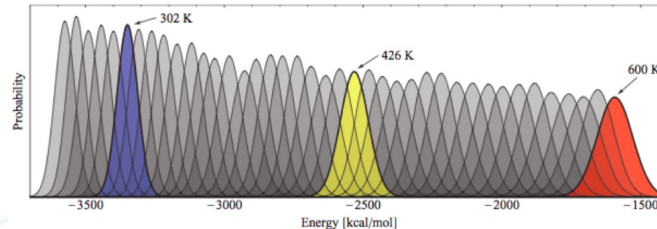
Parallel tempering

- Low temperature simulations on rugged potential energy surfaces can be trapped for long times in similar **metastable configurations** because the energy barriers to structurally potentially competing different conformations can be very high.
- The **parallel (or simulated) tempering** technique was introduced to overcome the difficulties in the evaluation of thermodynamic observables for models with **very rugged potential energy surfaces (PES)** and applied previously in several protein folding studies. As an optimization procedure, it resembles the simulated annealing technique
- In parallel tempering we consider n systems. In each of these systems we perform a simulation in the canonical (NVT) ensemble, but each system is at a different temperature.
- Systems with a sufficiently high temperature pass over the potential. The low-temperature systems, on the other hand, mainly probe the local energy minima. The idea of parallel tempering is to include MC trial moves that attempt to **"swap" systems that belong to different thermodynamic states**, e.g., to swap a high-temperature configuration with a low temperature configuration.

28

28

- If the temperature difference between the two systems is very large, such a swap has a very low probability of being accepted instead of making attempts to swap between a low and a high temperature, **we swap between ensembles with a small temperature difference.**



- Let us call the temperature of system i , $T_i = 1/\beta_i k_B$, and the systems are numbered according to an increasing temperature scale, $T_1 < T_2 < \dots < T_n$. We define an extended ensemble that is the combination of all n subsystems.
- The **partition function** of this **extended ensemble** is the product of all individual NVT_i ensembles:

$$Z_{PT} = \prod_{i=1}^n Z_{NVT_i} = \prod_{i=1}^n \frac{1}{\Lambda_i^{3N} N!} \int d\vec{x}_i e^{-\beta_i L(\vec{x}_i)}$$

29

29

- To sample this extended ensemble it is in principle sufficient to perform NVT_i simulations of all individual ensembles. But we can also introduce a Monte Carlo move, which consists of a swap between two ensembles. The acceptance rule of a swap between ensembles i and j follows from the condition of **detailed balance**.

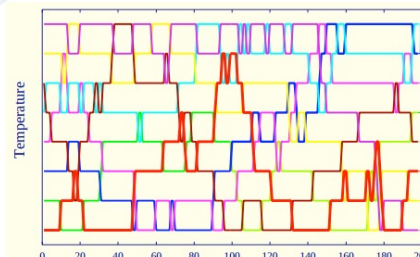
30

- It turns out to be (it should be accepted both for i and j):

$$P = \min\left(1, e^{-\{(\beta_j - \beta_i)[L(\vec{x}_i) - L(\vec{x}_j)]\}}\right) \quad \text{where } \beta_i \text{ and } L(\vec{x}_i) \text{ are the inverse temperatures, energy and configuration of the } i^{\text{th}} \text{ replic}$$

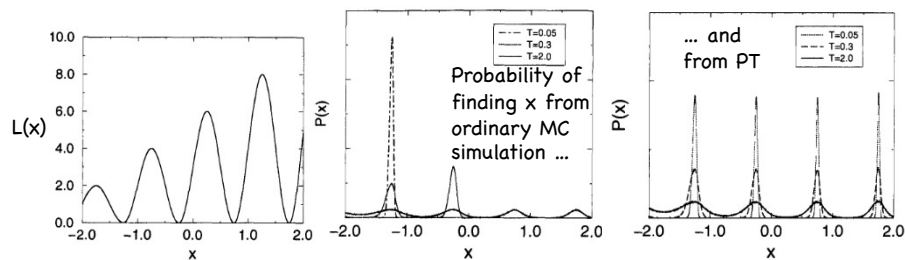
- Since we know the total "energy" of a configuration anyway, these **swap moves are very inexpensive** since they do not involve additional calculations.

- The temperature scale for the highest and lowest temperatures is determined by the requirement to efficiently explore the configurational space and to accurately resolve local minima, respectively.



30

- Once **parallel tempering** is used for a **ordinary simulation**, the exchange mechanism improves the configurational averaging of the low-temperature simulations, because the exchange with high-temperature simulations provides a mechanism to overcome the high energy barriers between low-lying metastable configurations.



- Applied as an **optimization technique**, however the simulation associated with the lowest temperature will typically yield the estimate for the **global optimum**, while all others are required to generate different configurations.
- The computational effort of the method rises linearly with the number of temperatures.

31

31

Lecture 8: Suggested books

- J.C. Spall, *Introduction to Stochastic Search and Optimization*, Wiley (2003)
- A.K. Hartmann & H. Rieger, *Optimization Algorithms in Physics*, Wiley (2002)
- A. Brabazon, M. O'Neill, S. McGarraghy, *Natural Computing Algorithms*, Springer (2015)

32

32