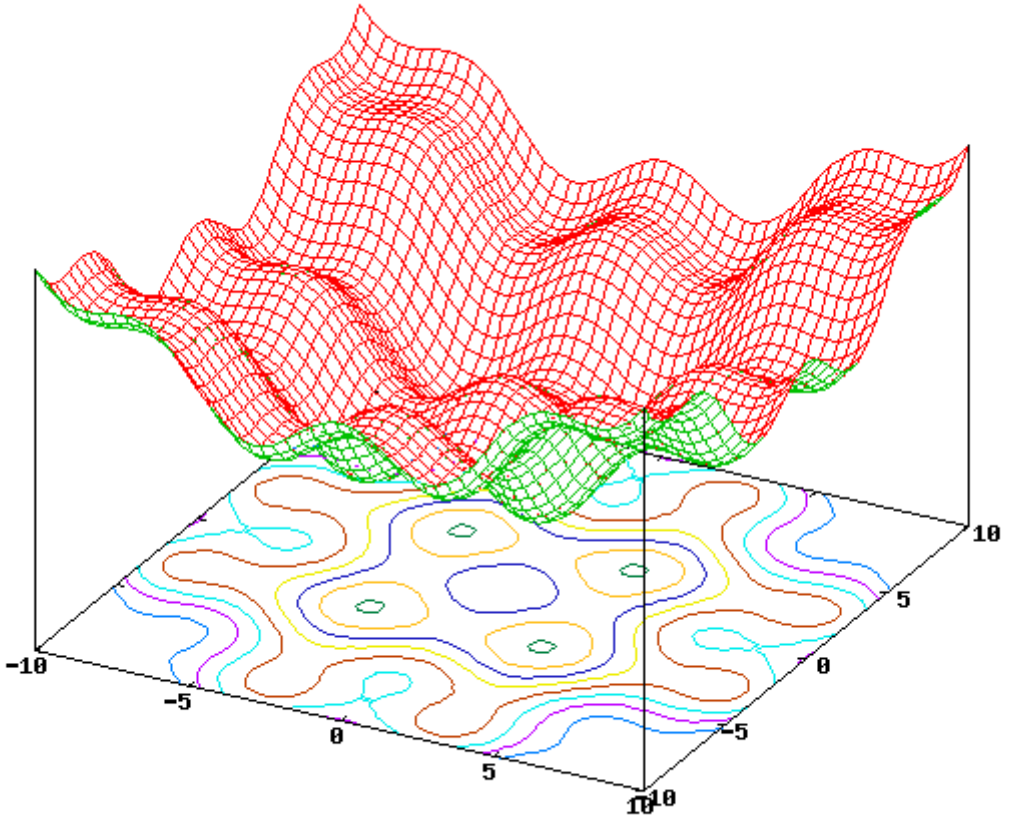
Part 15

Global optimization

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g_i(x) = 0, \qquad i = 1, \ldots, n_e \\ & h_i(x) \geq 0, \qquad i = 1, \ldots, n_i \end{aligned}$$

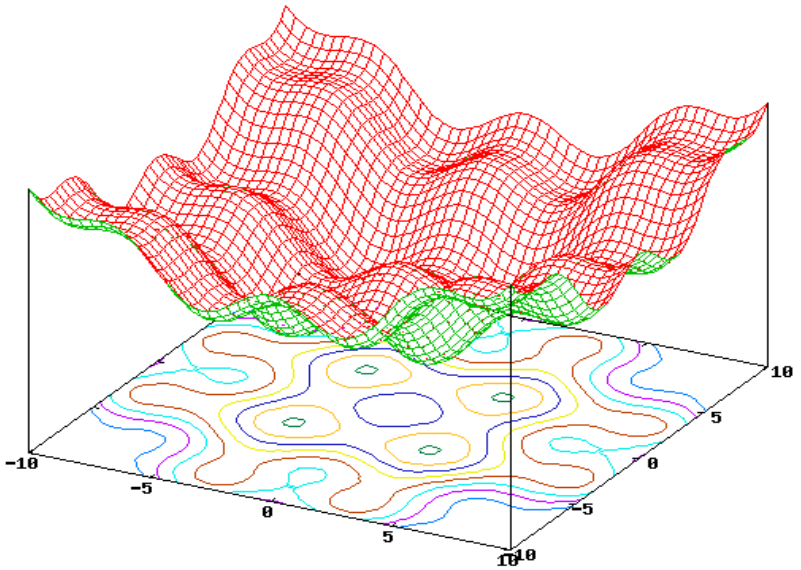
Motivation

What should we do when asked to find the (global) minimum of functions like this:



$$f(x)=\frac{1}{20}(x_1^2+x_2^2)+\cos(x_1)+\cos(x_2)$$

A naïve sampling approach



Naïve approach: Sample at *M*-by-*M* points and choose the one with the smallest value.

Alternatively: Start Newton's method at each of these points to get higher accuracy.

Problem: If we have *n* variables, then we would have to start at M^n points. This becomes prohibitive for large *n*!

Monte Carlo sampling

A better strategy ("Monte Carlo" sampling):

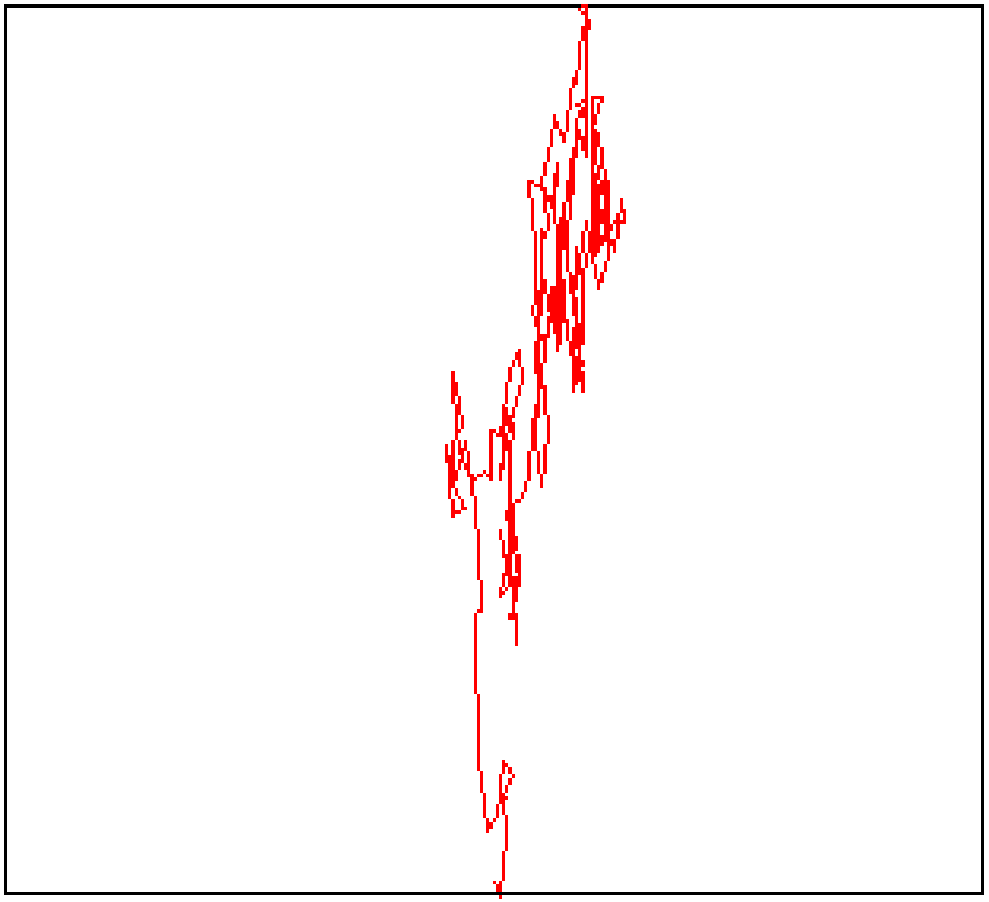
- Start with a feasible point x_0
- For *k=0,1,2,...*:
  - Choose a trial point x_t
  - If $f(x_t) \leq f(x_k)$ then $x_{k+1} = x_t$ [*accept* the sample]
  - Else:
    . draw a random number *s* in *[0,1]*
    . if $\exp\left[-\frac{f(x_t)-f(x_k)}{T}\right] \geq s$

    then
    $x_{k+1} = x_t$ [*accept* the sample]

    else
    $x_{k+1} = x_k$ [*reject* the sample]

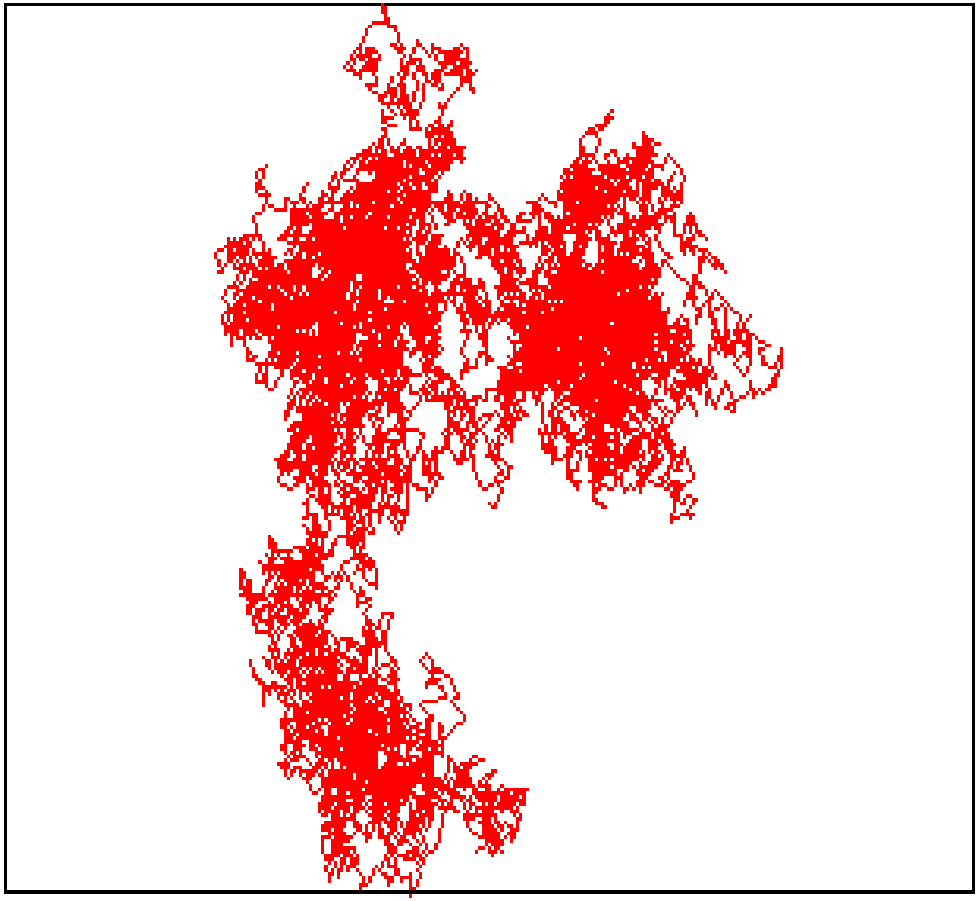
Monte Carlo sampling

Example: The first 200 sample points



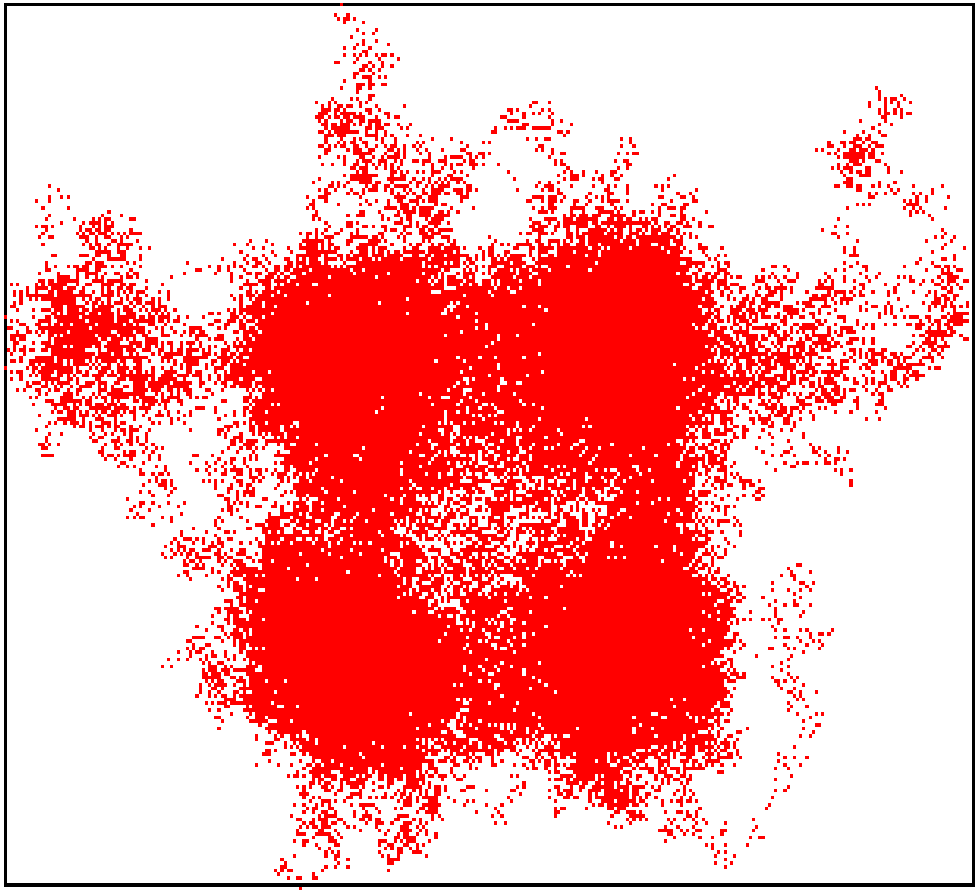
Monte Carlo sampling

Example: The first 10,000 sample points



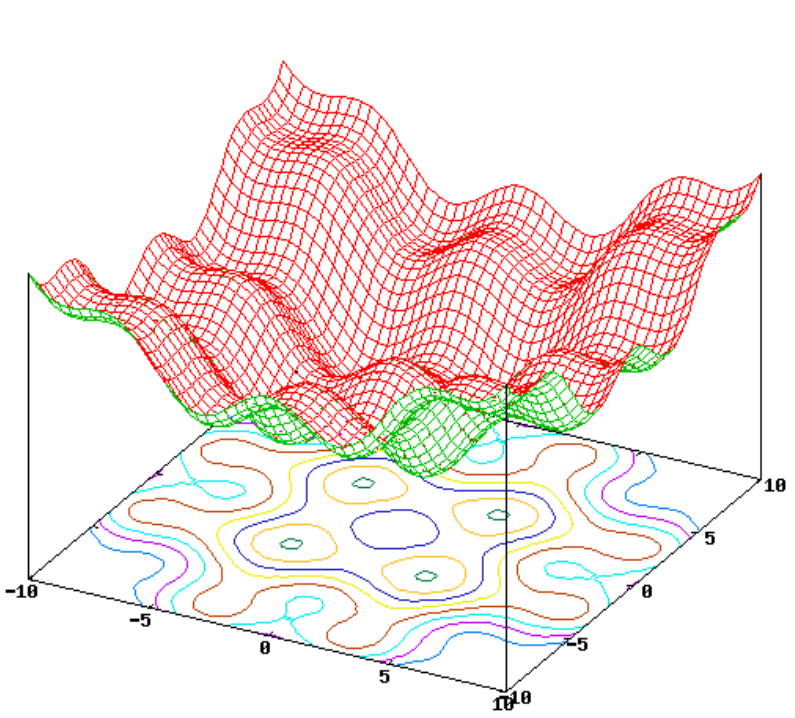
Monte Carlo sampling

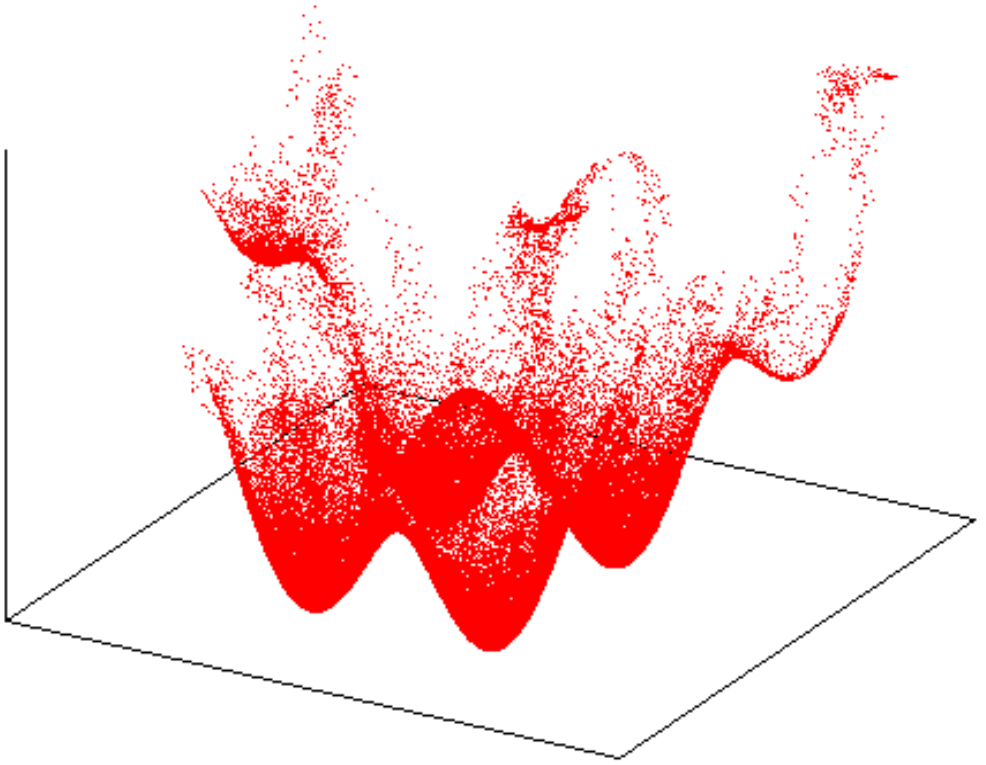
Example: The first 100,000 sample points



Monte Carlo sampling

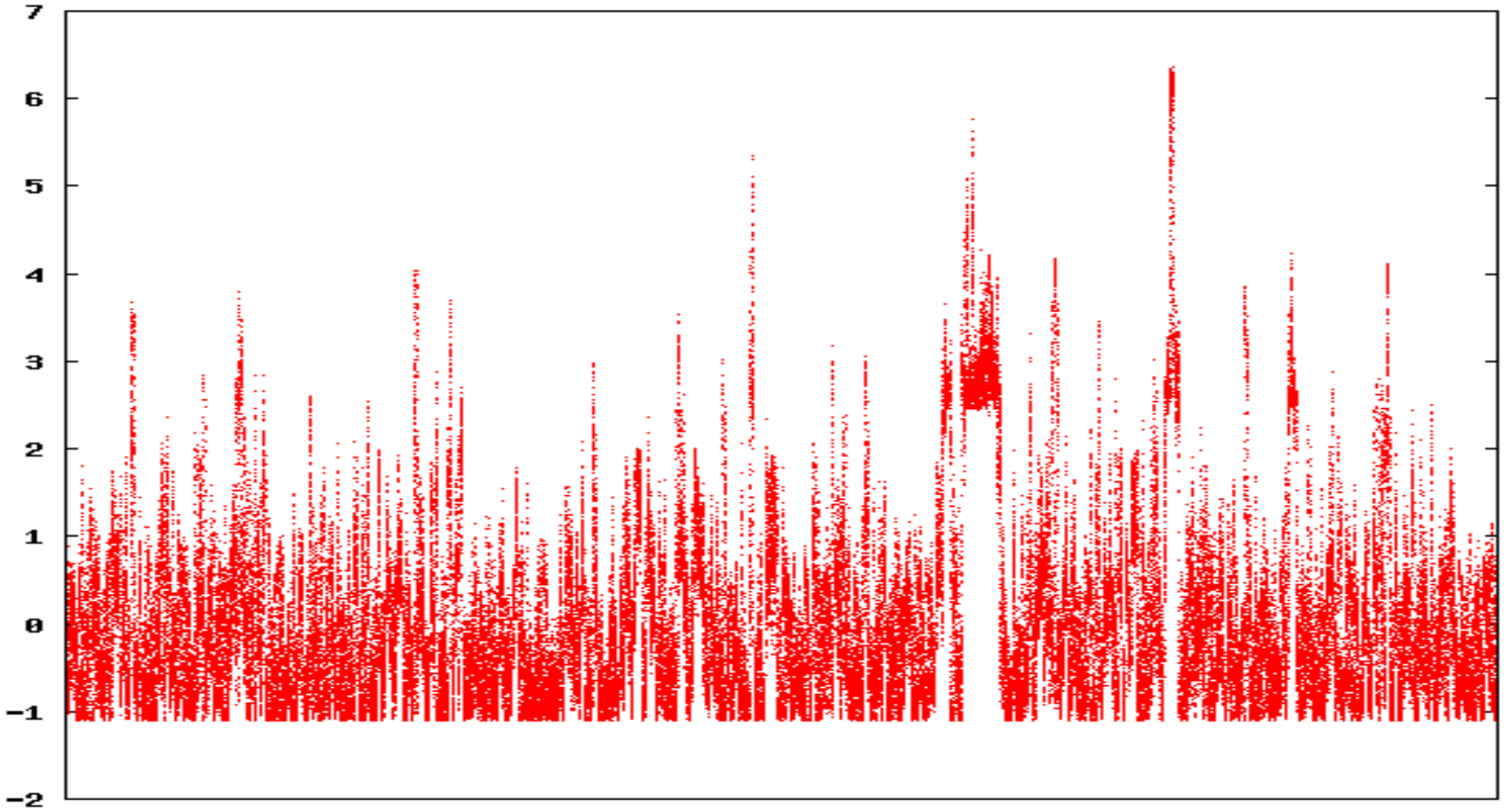
Example: Locations and values of the first 10^5 sample points





Monte Carlo sampling

Example: Values of the first 100,000 sample points



Note: The exact minimal value is -1.1032... . In the first 100,000 samples, we have 24 with values *f(x)<-1.103*.

Monte Carlo sampling

How to choose the constant *T*:

- If *T* is chosen too small, then the condition

$$\exp\left[-\frac{f(x_t)-f(x_k)}{T}\right] \geq s, \qquad s \in U([0,1])$$

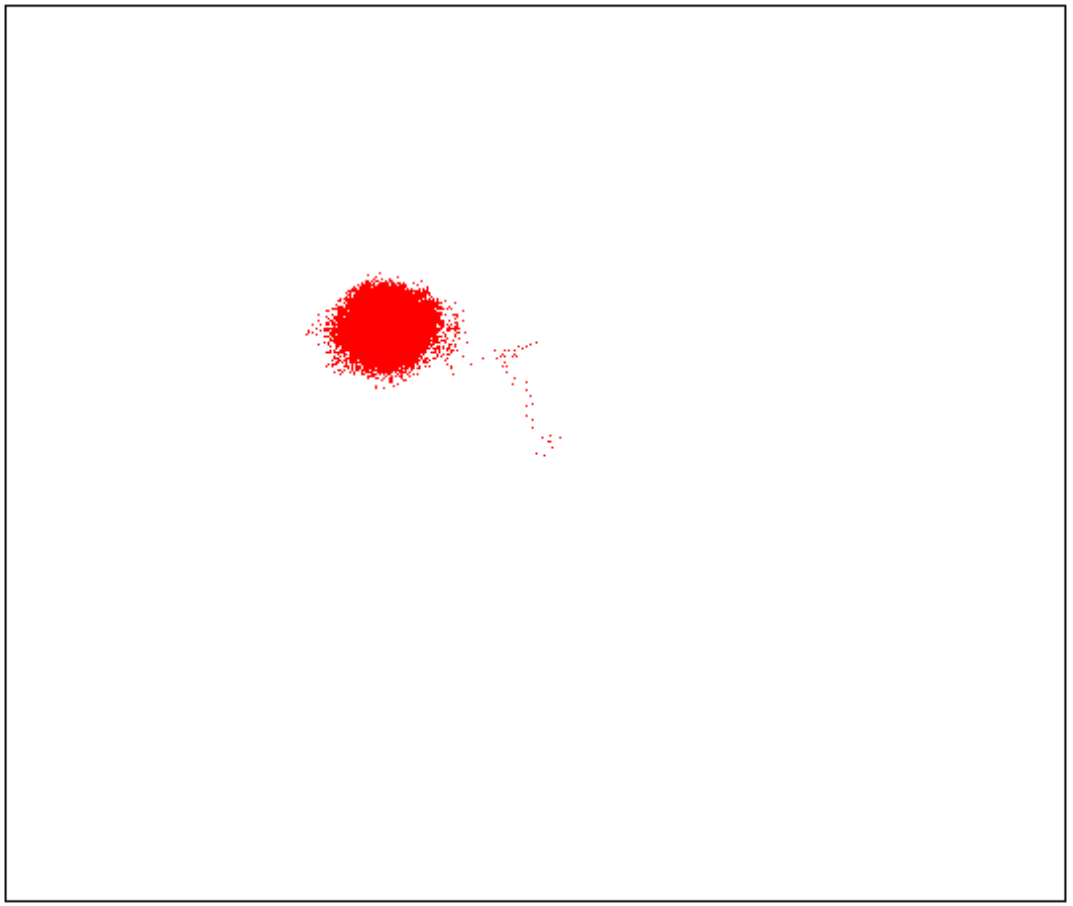
  will lead to frequent rejections of sample points for which *f(x)* increases.
  Consequently, we will get stuck in local minima for long periods of time before we accept a sequence of steps that gets "us over the hump".

- On the other hand, if *T* is chosen too large, then we will accept nearly every sample, irrespective of $f(x_t)$.

  Consequently, we will perform a *random walk* that is no more efficient than uniform sampling.

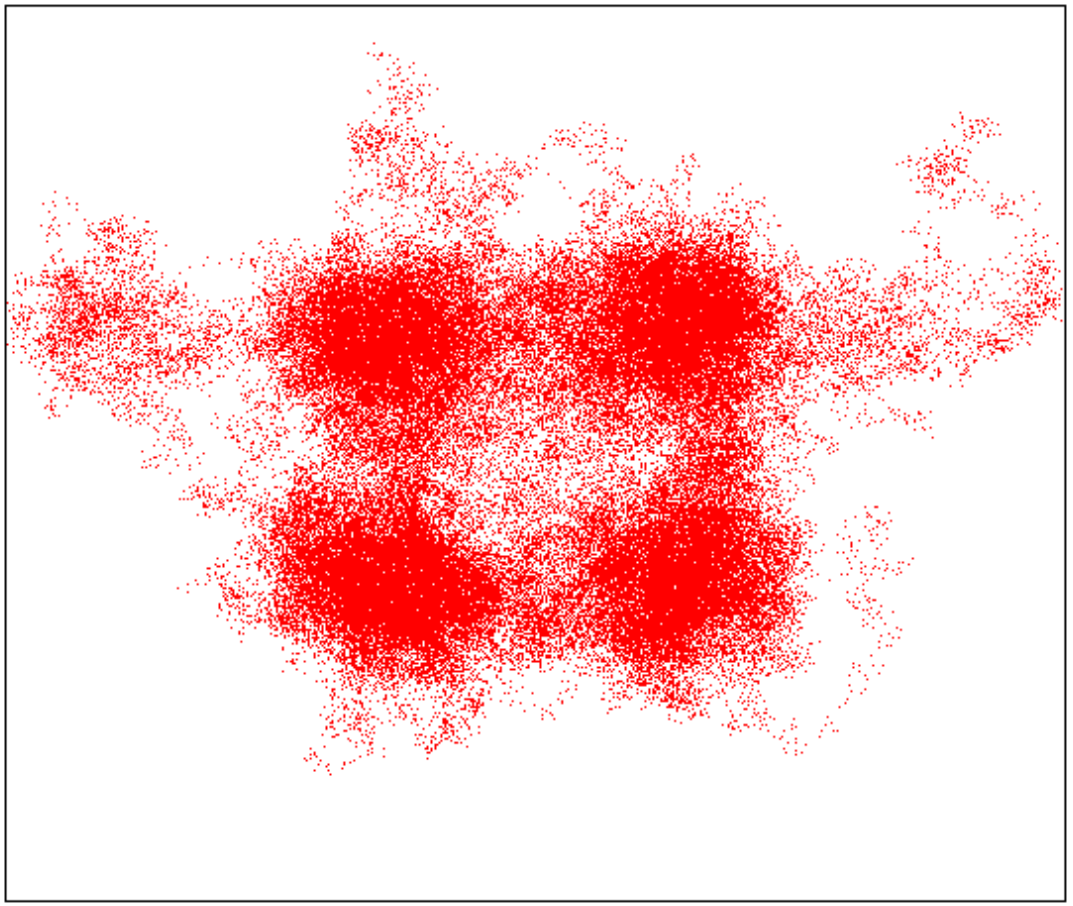
Monte Carlo sampling

Example: First 100,000 samples, T=0.1



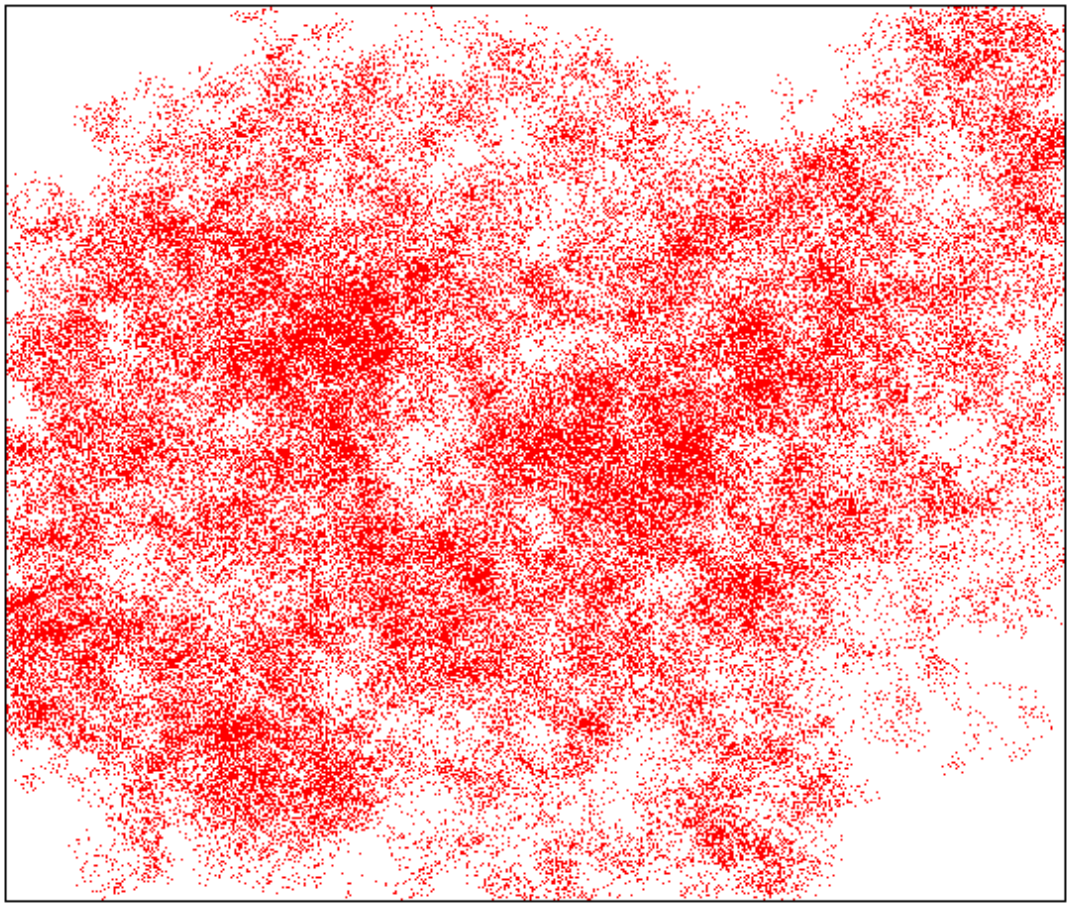
Monte Carlo sampling

Example: First 100,000 samples, T=1



Monte Carlo sampling

Example: First 100,000 samples, T=10



Monte Carlo sampling

Strategy: Choose *T* large enough that there is a reasonable probability to get out of local minima; but small enough that this doesn't happen too often.

Example: For $f(x)=\frac{1}{20}(x_1^2+x_2^2)+\cos(x_1)+\cos(x_2)$

the difference in function value between local minima and saddle points is around 2. We want to choose *T* so that

$$\exp\left[-\frac{\Delta f}{T}\right] \geq s, \qquad s\in U([0,1])$$

is true maybe 10% of the time.

This is the case for *T=0.87.*

Monte Carlo sampling

How to choose the next sample x_t:

- If x_t is chosen independently of x_k then we just sample the entire domain, without exploring areas where *f(x)* is small. Consequently, we should choose x_t "close" to x_k.
- If we choose x_t too close to x_k we will have a hard time exploring a significant part of the feasible region.
- If we choose x_t in an area around x_k that is too large, then we don't adequately explore areas where *f(x)* is small.

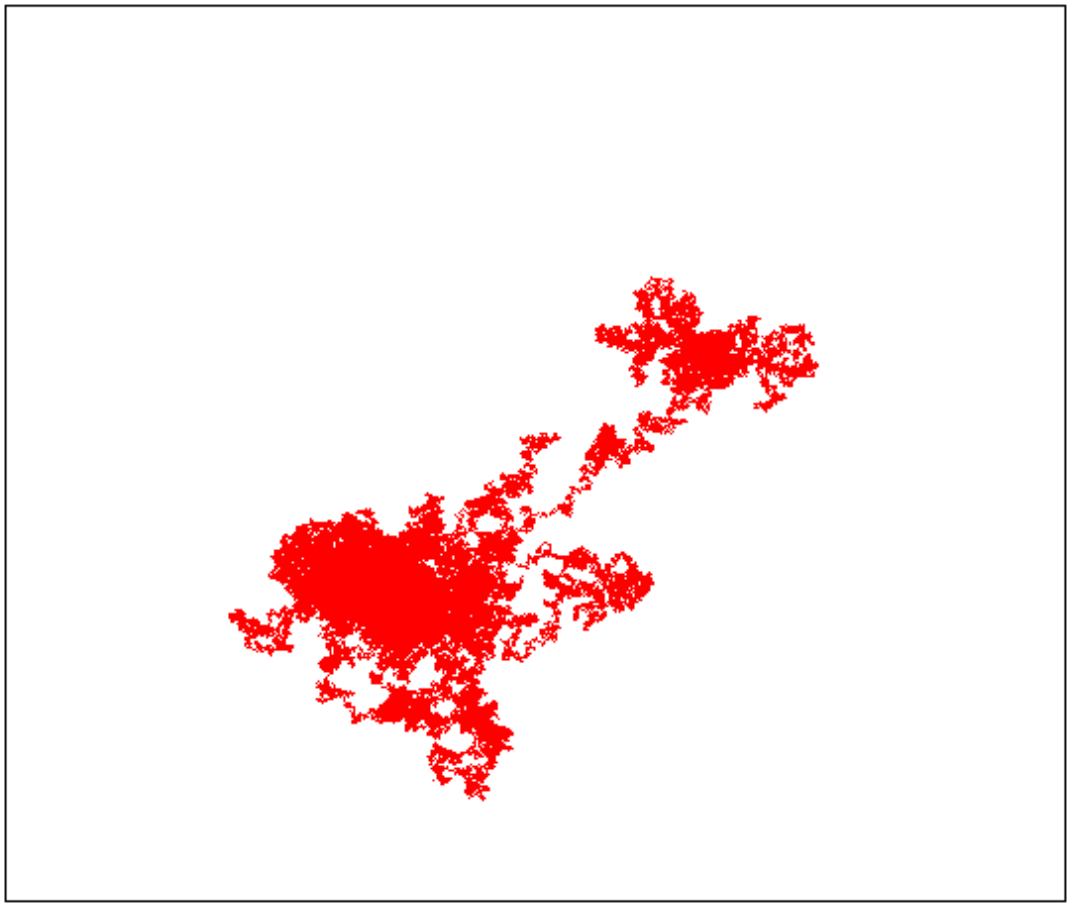
Common strategy: Choose

$$x_t = x_k + \sigma\, y, \qquad y \in N(0, I) \text{ or } U([-1,1]^n)$$

where σ is a fraction of the diameter of the domain or the distance between local minima.

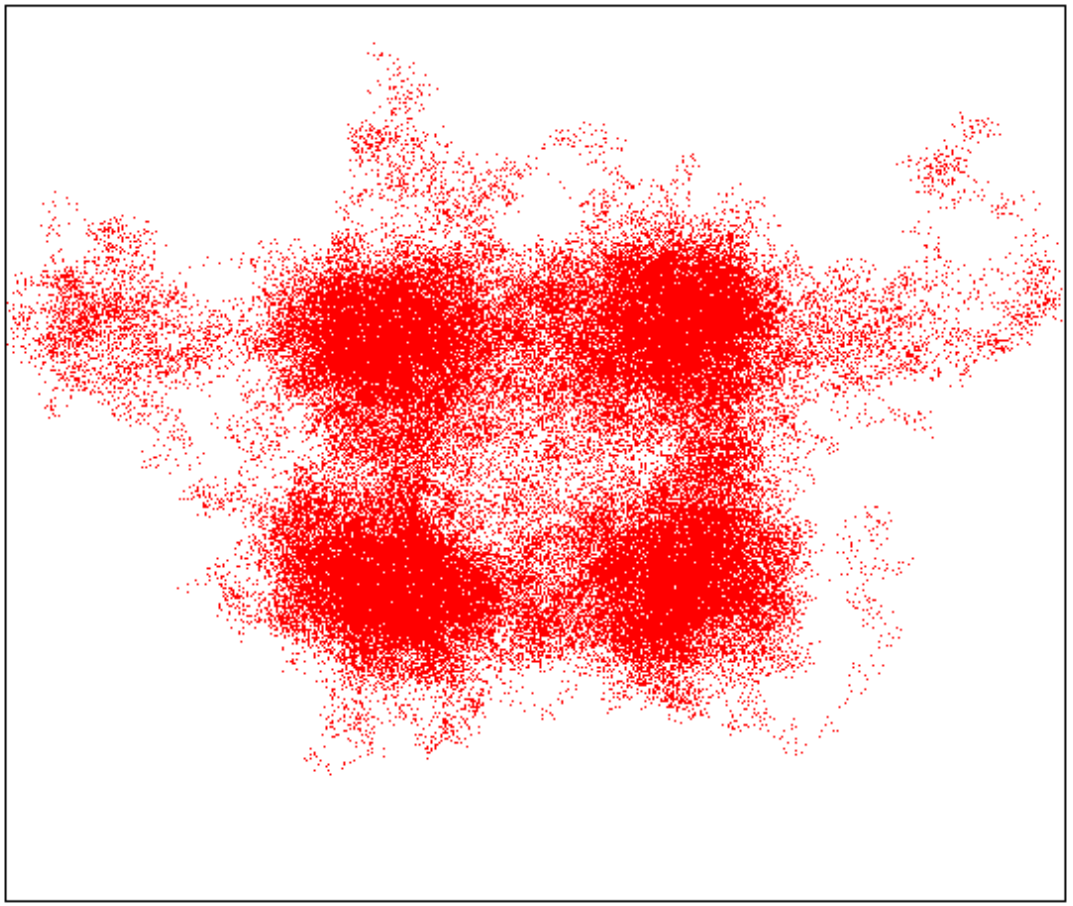
Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=0.05



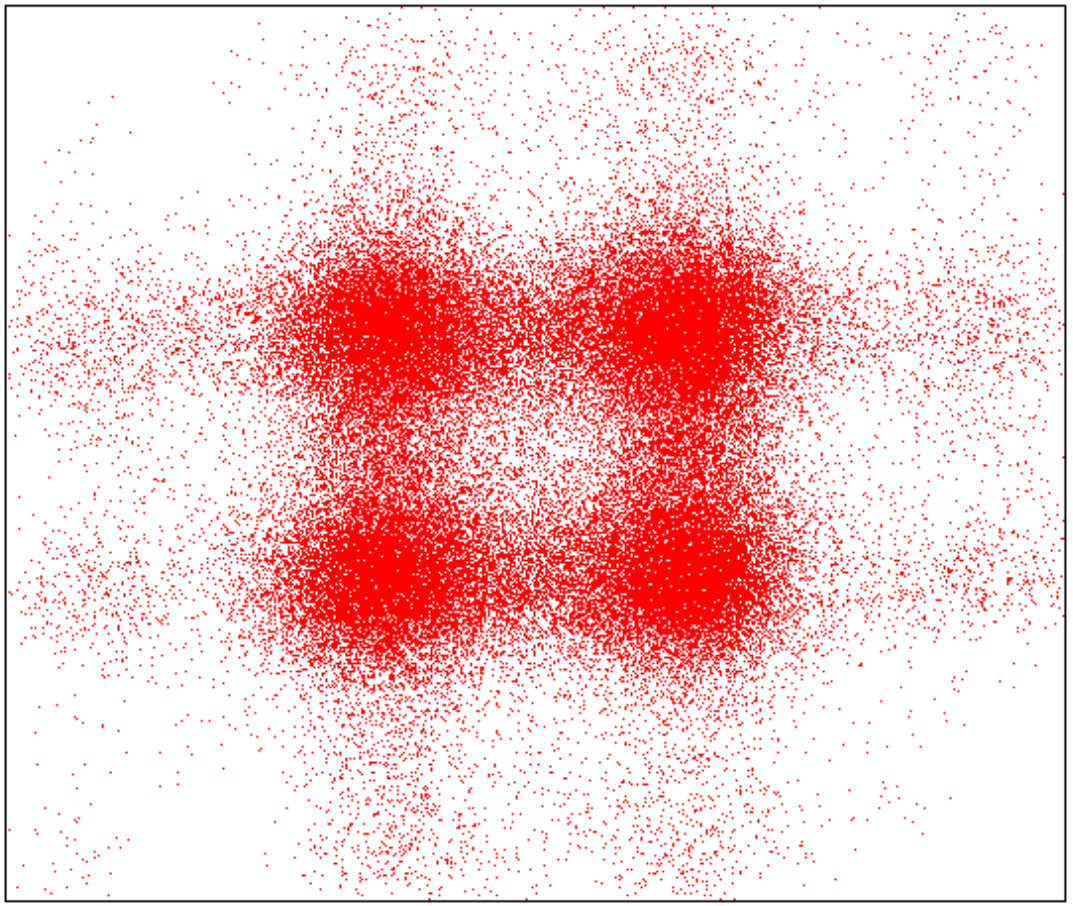
Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=0.25



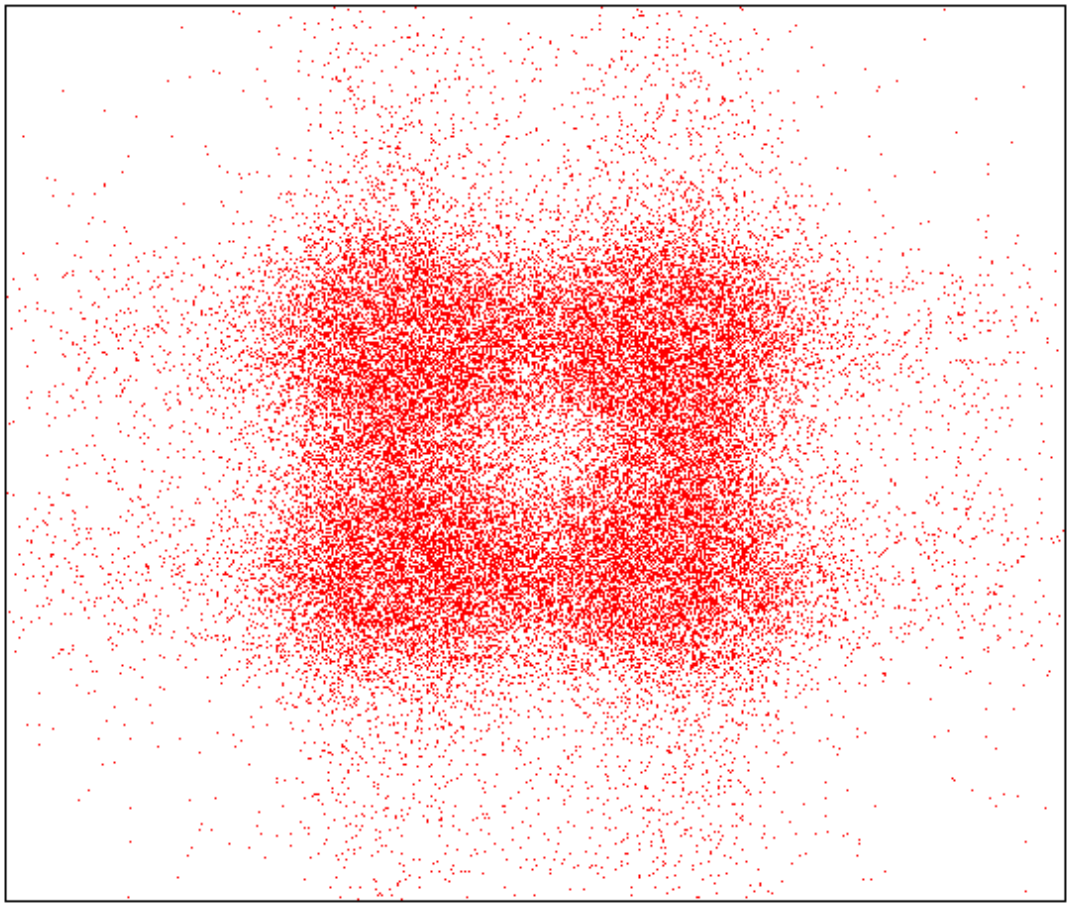
Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=1



Monte Carlo sampling

Example: First 100,000 samples, T=1, σ=4



Monte Carlo sampling with constraints

Inequality constraints:

- For simple inequality constraints, modify sample generation strategy to never generate infeasible trial samples
- For complex inequality constraints, always reject samples for which

$$h_i(x_t) < 0 \qquad \text{for at least one } i$$

Monte Carlo sampling with constraints

Inequality constraints:

- For simple inequality constraints, modify the sample generation strategy to never generate infeasible trial samples
- For complex inequality constraints, always reject samples:

  - If $Q(x_t) \leq Q(x_k)$ then $x_{k+1} = x_t$

  - Else:

    . draw a random number s in *[0,1]*

    . if $$\exp\left[-\frac{Q(x_t) - Q(x_k)}{T}\right] \geq s$$

    then $$x_{k+1} = x_t$$

    else $$x_{k+1} = x_k$$

  where

$$Q(x) = \infty \text{ if at least one } h_i(x) < 0, \qquad Q(x) = f(x) \text{ otherwise}$$

Monte Carlo sampling with constraints

Equality constraints:

- Generate only samples that satisfy equality constraints

- If we have only linear equality constraints of the form

$$g(x) = Ax - b = 0$$

then one way to guarantee this is to generate samples using

$$x_t = x_k + \sigma Z y, \qquad y \in \mathbb{R}^{n-n_e}, \quad y = N(0, I) \text{ or } U([-1,1]^{n-n_e})$$

where Z is the null space matrix of A, i.e. $AZ=0$.

Monte Carlo sampling

Theorem:

Let *A* be a subset of the feasible region. Under certain conditions on the sample generation strategy, then as $k \to \infty$ we have

$$\text{number of samples } x_k \in A \quad \propto \quad \int_A e^{-\frac{f(x)}{T}} dx$$

That is: Every region *A* will be adequately sampled over time. Areas around the global minimum will be better sampled than other regions.

In particular,

$$\text{fraction of samples } x_k \in A \quad = \quad \frac{1}{C} \int_A e^{-\frac{f(x)}{T}} dx + O\left(\frac{1}{\sqrt{N}}\right)$$

Monte Carlo sampling

Remark:

Monte Carlo sampling appears to be a strategy that bounces around randomly, only taking into account the *values* (not the *derivatives*) of *f(x)*.

However, that is not so if sample generation strategy and *T* are chosen carefully: Then we choose a new sample moderately close to the previous one, and we *always* accept it if *f(x)* is reduced, whereas we only *sometimes* accept it if *f(x)* is increased by this step.

In other words: On average we still move in the direction of steepest descent!

Monte Carlo sampling

Remark:

Monte Carlo sampling appears to be a strategy that bounces around randomly, only taking into account the *values* (not the *derivatives*) of *f(x)*.

However, that is not so – because it *compares* function values.

That said: One can accelerate the Monte Carlo method by choosing samples from a distribution that is *biased towards the negative gradient direction* if the gradient is cheap to compute.

Such methods are sometimes called *Langevin samplers*.

Simulated Annealing

Motivation:
Particles in a gas, or atoms in a crystal have an energy that is on average in equilibrium with the rest of the system. At any given time, however, its energy may be higher or lower.

In particular, the probability that its energy is E is

$$P(E) \propto e^{-\frac{E}{k_B T}}$$

Where k_B is the Boltzmann constant. Likewise, probability that a particle can overcome an energy barrier of height ΔE is

$$P(E \to E + \Delta E) \propto \min\left\{1, e^{-\frac{\Delta E}{k_B T}}\right\} = \begin{cases} 1 \text{ if } \Delta E \le 0 \\ e^{-\frac{\Delta E}{k_B T}} \text{ if } \Delta E > 0 \end{cases}$$

This is exactly the Monte Carlo transition probability if we identify

$$E = f\, k_B$$

Simulated Annealing

Motivation:
In other words, Monte Carlo sampling is analogous to watching particles bounce around in a potential *f(x)* when driven by a gas at constant temperature.

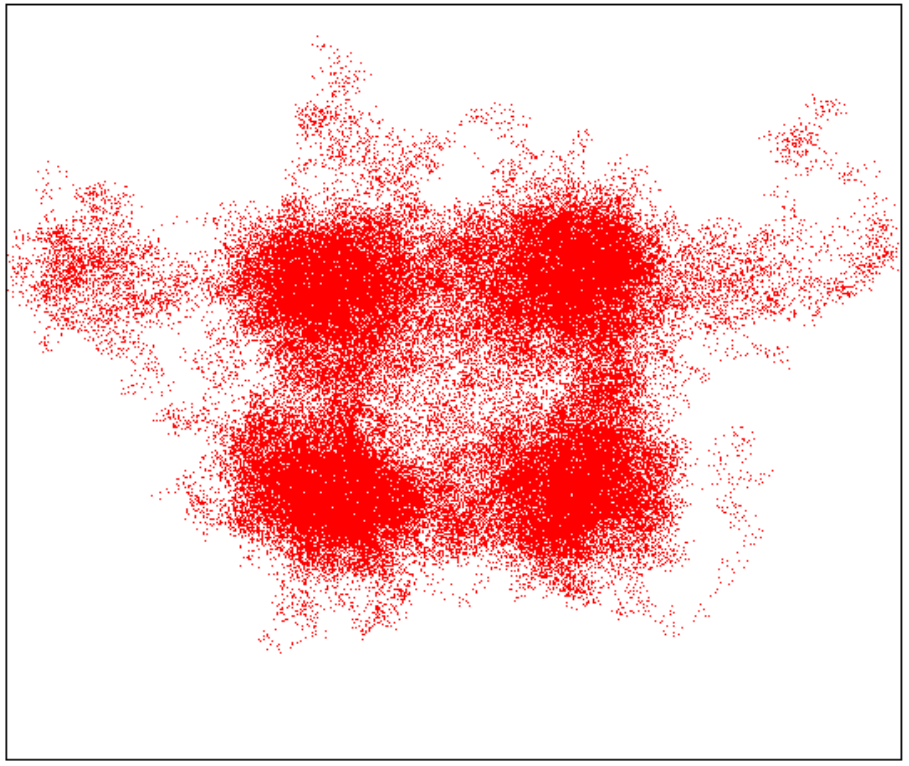
On the other hand, we know that if we slowly reduce the temperature of a system, it will end up in the ground state with very high probability. For example, slowly reducing the temperature of a melt results in a perfect crystal. (On the other hand, reducing the temperature too quickly results in a glass.)

The *Simulated Annealing* algorithm uses this analogy by using the modified transition probability

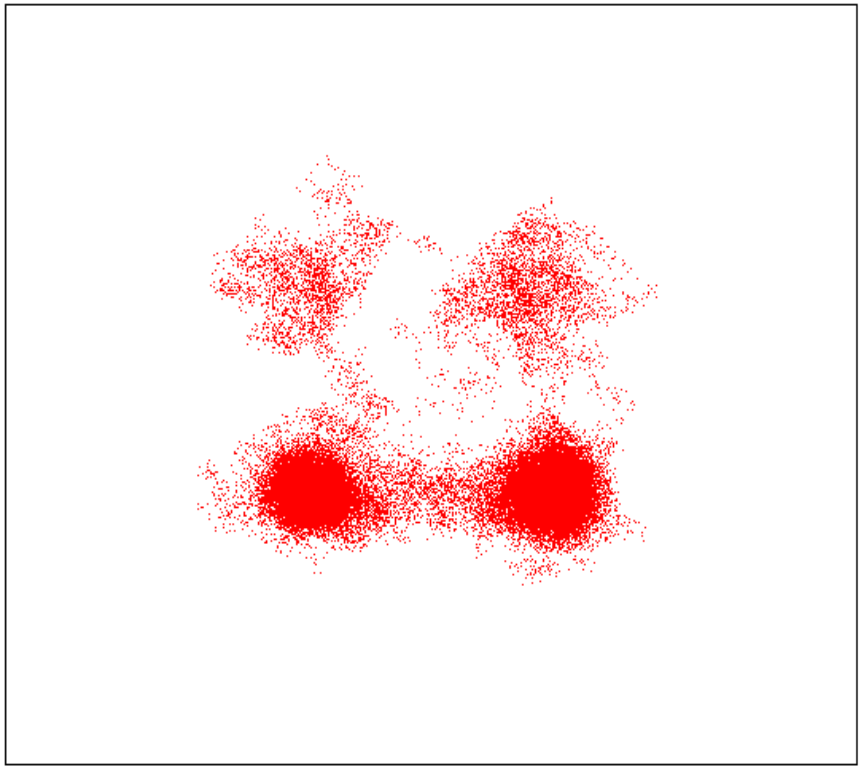
$$\exp\left[-\frac{f(x_t)-f(x_k)}{T_k}\right] \geq s, \qquad s \in U([0,1]), \qquad T_k \to 0 \text{ as } k \to \infty$$

Simulated Annealing

Example: First 100,000 samples, σ=0.25



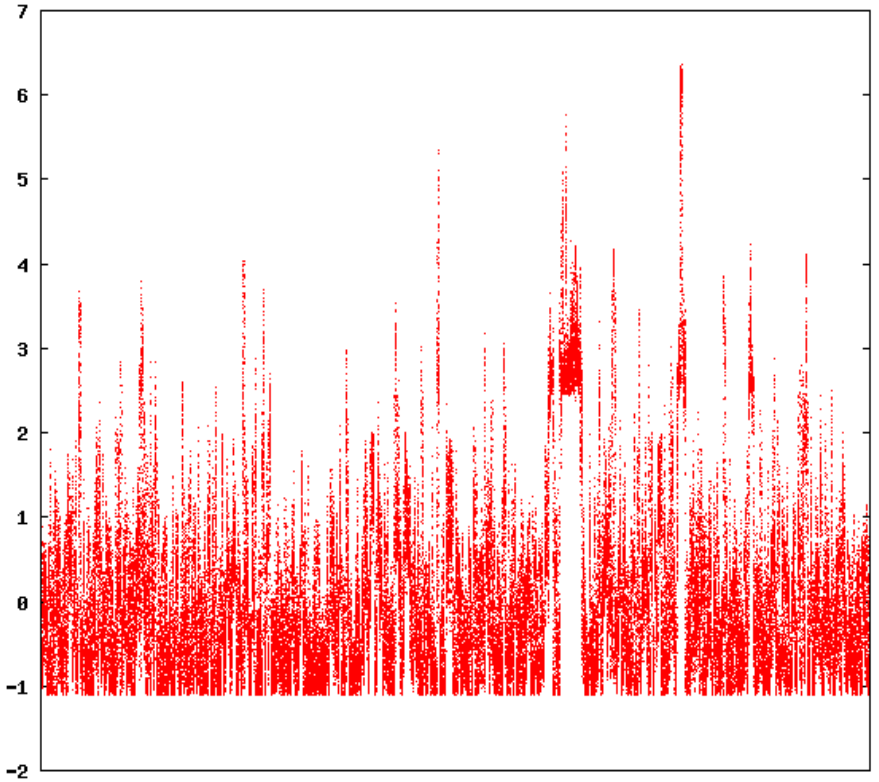
$T=1$

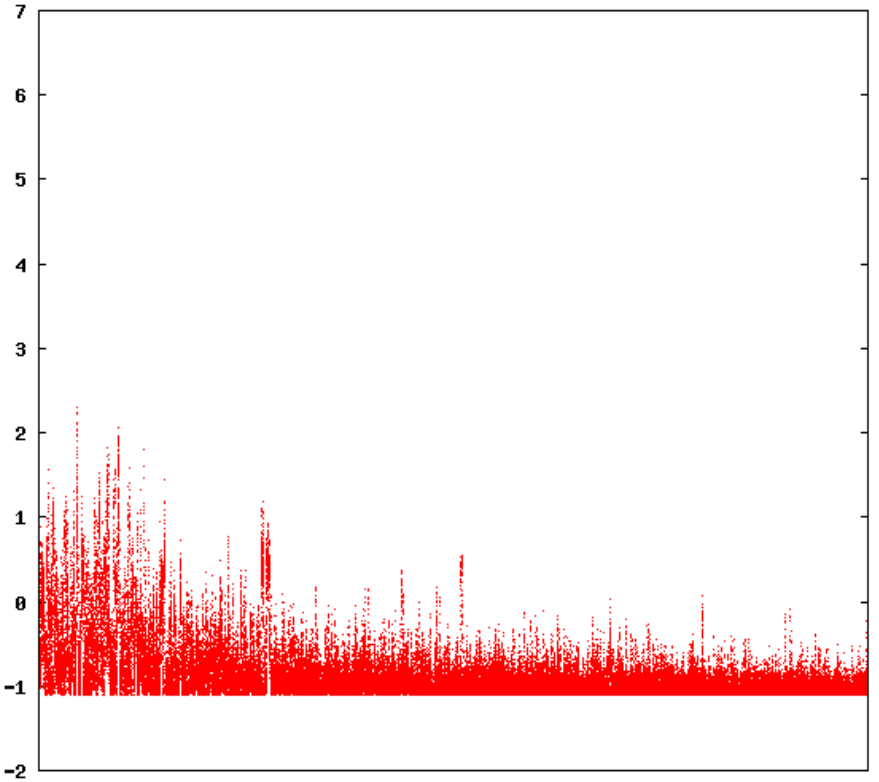


$$T_k = \frac{1}{1+10^{-4}k}$$

Simulated Annealing

Example: First 100,000 samples, σ=0.25





$T=1$

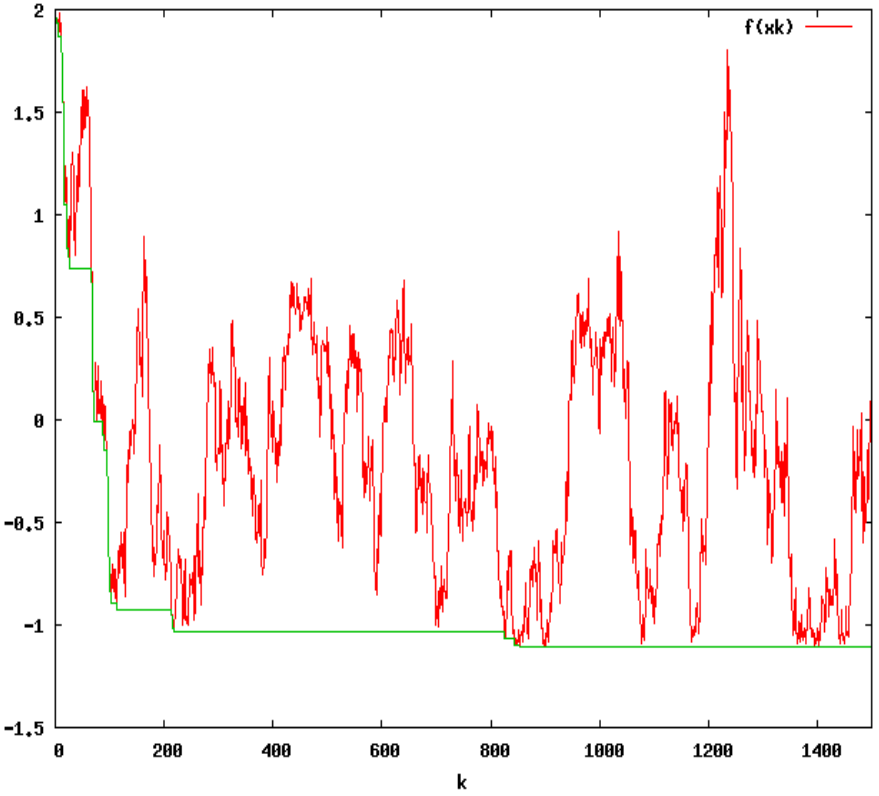
$T_k=\frac{1}{1+10^{-4}k}$

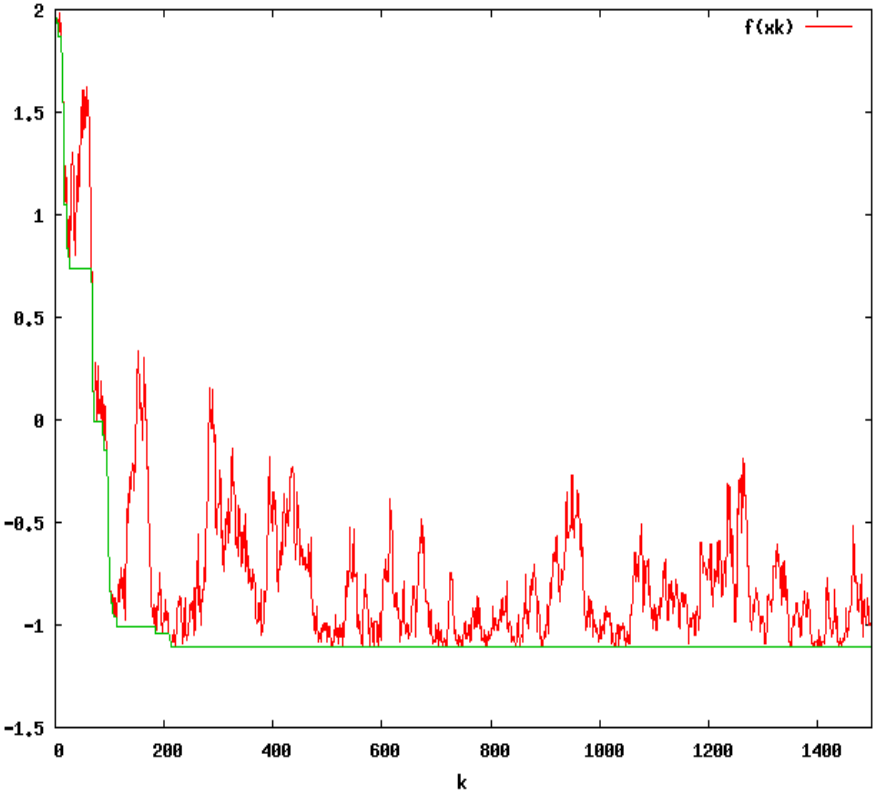
24 samples with *f(x)<-1.103*

192 samples with *f(x)<-1.103*

Simulated Annealing

Convergence: First 1,500 samples, $f(x)=\sum_{i=1}^{2}\frac{1}{20}x_i^2+\cos(x_i)$





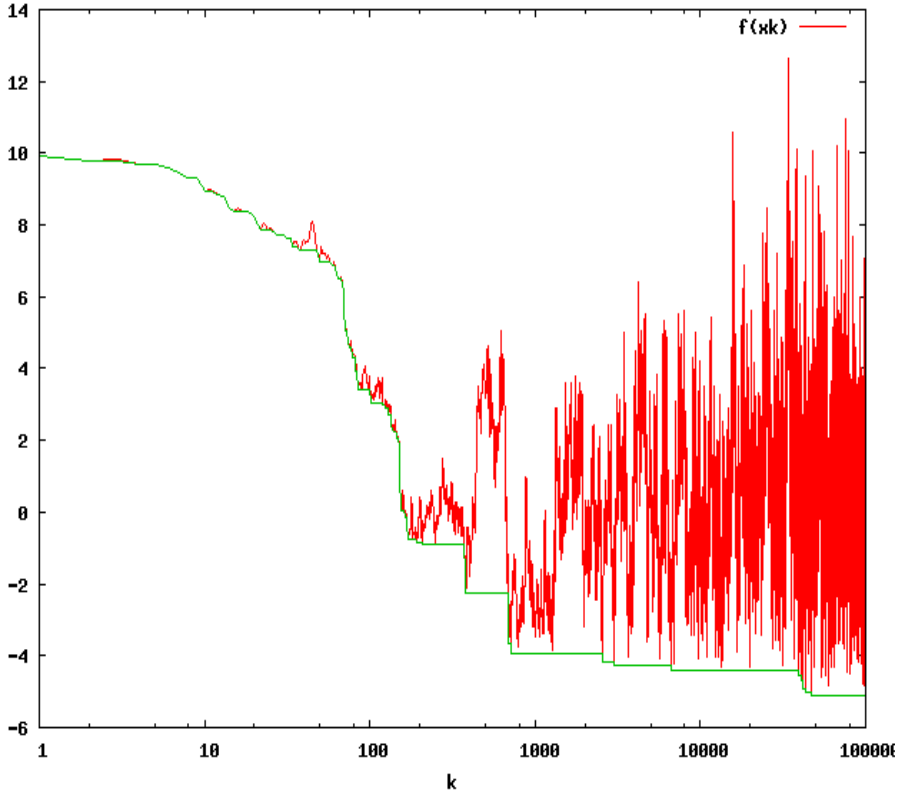
$T=1$

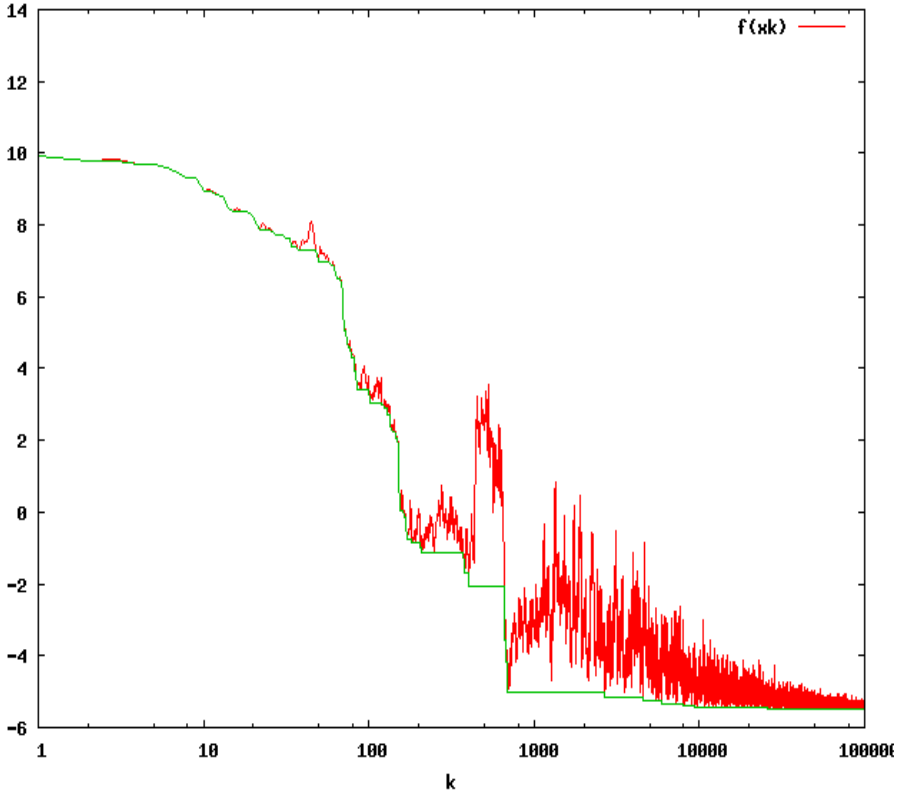
$T_k=\frac{1}{1+0.005\,k}$

(Green line indicates the lowest function value found so far)

Simulated Annealing

Convergence: First 10,000 samples, $f(x)=\sum_{i=1}^{10}\frac{1}{20}x_i^2+\cos(x_i)$





$T=1$

$T_k=\frac{1}{1+0.0005k}$

(Green line indicates the lowest function value found so far)

Simulated Annealing

Discussion:
Simulated Annealing is often more efficient in finding global minima because it *initially explores the energy landscape at large*, and later on explores the areas of low energy in greater detail.

On the other hand, there is now another knob to play with (namely how we reduce the temperature):

- If the temperature is reduced too fast, we may get stuck in local minima (the “glass” state)
- If the temperature is not reduced fast enough, the algorithm is no better than Monte Carlo sampling and may require many many samples.

Very Fast Simulated Annealing (VFSA)

A further refinement:
In *Very Fast Simulated Annealing* we not only reduce temperature over time, but also reduce the search radius of our sample generation strategy, i.e. we compute

$$x_t = x_k + \sigma_k \, y, \qquad y \in N(0, I) \text{ or } U([-1,1]^n)$$

and let

$$\sigma_k \to 0$$

Like reducing the temperature, this ensures that we sample the vicinity of minima better and better over time.

Remark: To guarantee that the algorithm can reach any point in the search domain, we need to choose σ_k so that

$$\sum_{k=0}^{\infty} \sigma_k = \infty$$

Genetic Algorithms (GA)

An entirely different idea:

Choose a set ("population") of N points ("individuals")

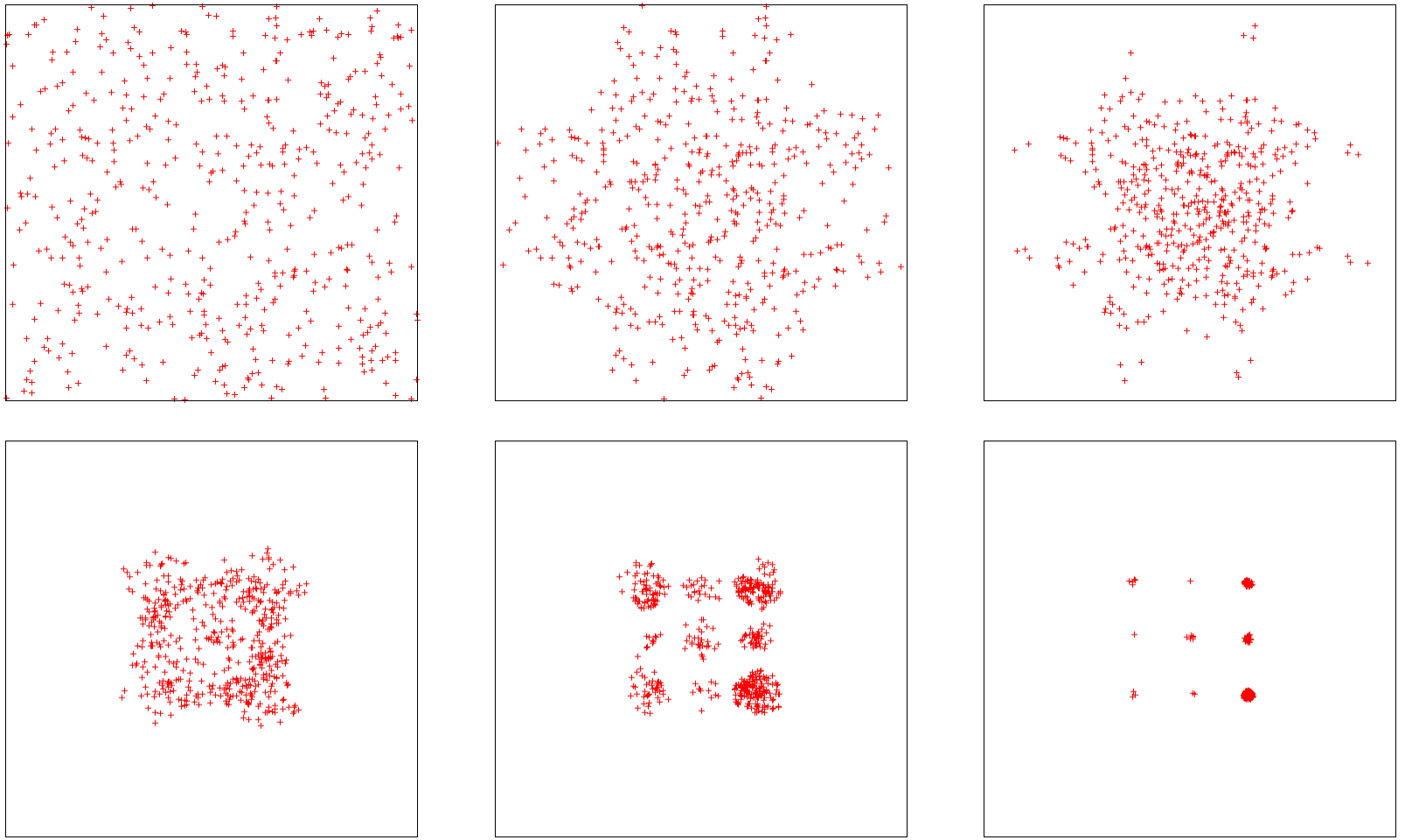
$P_0=\{x_1,...x_N\}$

For $k=0,1,2,...$ ("generations"):

- Copy those $N_f<N$ individuals in P_k with the smallest $f(x)$ (i.e. the "fittest individuals") into P_{k+1}
- While $\#P_{k+1}<N$:
  - select two individuals ("parents") x_a,x_b from among the first N_f individuals in P_{k+1} with probabilities proportional to $e^{-f(x_i)/T}$
  - create a new point x_{new} from x_a,x_b ("mating")
  - perform some random changes on x_{new} ("mutation")
  - add it to P_{k+1}

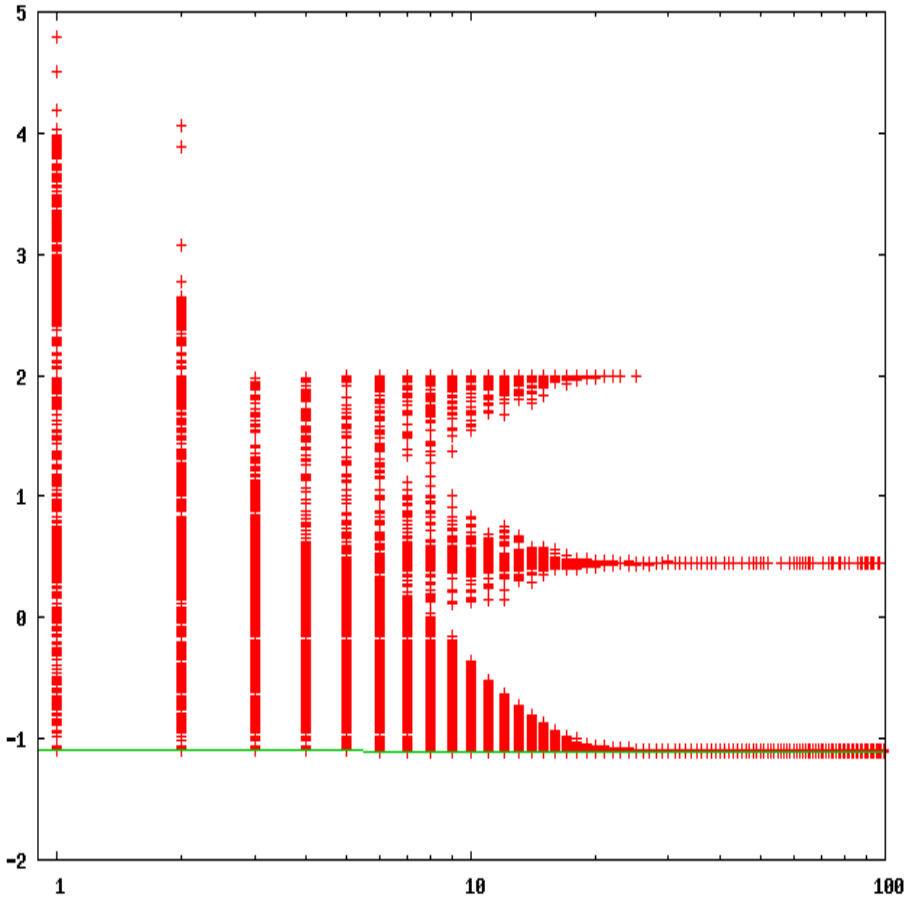
Genetic Algorithms (GA)

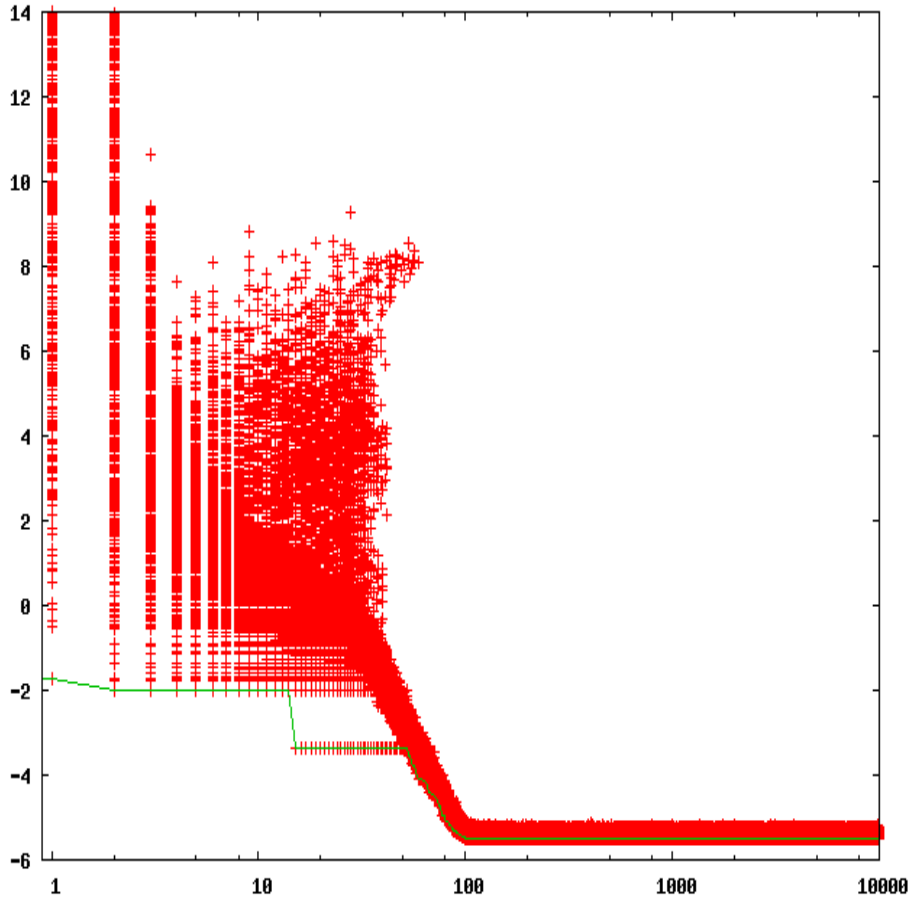
Example: Populations at *k=0,1,2,5,10,20, N=500, N_s=2/3 N*



Genetic Algorithms (GA)

Convergence: Values of the *N* samples for all generations *k*





$$f(x)=\sum_{i=1}^{2}\frac{1}{20}x_i^2+\cos(x_i)$$

$$f(x)=\sum_{i=1}^{10}\frac{1}{20}x_i^2+\cos(x_i)$$

Genetic Algorithms (GA)

Mating:

- Mating is meant to produce new individuals that share the traits of the two parents
- If the variable *x* encodes real values, then mating could just take the mean value of the parents:

$$x_{new} = \frac{x_a + x_b}{2}$$

- For more general properties (paths through cities, which of *M* objects to put where in a suitcase, …) we have to encode *x* in a binary string. Mating may then select bits (or bit sequences) randomly from each of the parents

- There is a huge variety of encoding and selection strategies in the literature.

Genetic Algorithms (GA)

Mutation:

- Mutations are meant to introduce an element of randomness into the process, to explore search directions that aren't represented yet in the population
- If the variable *x* represents real values, we can just add a small random value to *x* to simulate mutations

$$x_{new} = \frac{x_a + x_b}{2} + \epsilon\, y, \qquad y \in \mathbb{R}^n, \quad y = N(0, I)$$

- For more general properties, mutations can be introduced by randomly flipping individual bits or bit sequences in the encoded properties

- There is a huge variety of mutation strategies in the literature.

Part 15

Summary of global optimization methods

$$\begin{aligned} \text{minimize} \quad & f(x) \\ & g_i(x) = 0, \qquad i = 1, \ldots, n_e \\ & h_i(x) \geq 0, \qquad i = 1, \ldots, n_i \end{aligned}$$

Summary of methods

- **Global optimization problems** with many minima **are difficult** because of the curse of dimensionality: the number of places where a minimum could be becomes very large if the number of dimensions becomes large
- There is a **large zoo of methods** for these kinds of problems
- **Most algorithms are stochastic** to sample feasible region
- Algorithms also work for non-smooth problems
- Most methods are not very effective (if one counts number of function evaluations) in return for the ability to get out of local minima

- Global optimization algorithms should *never* be used whenever we know that the problem has only a small number of minima and/or is smooth and convex