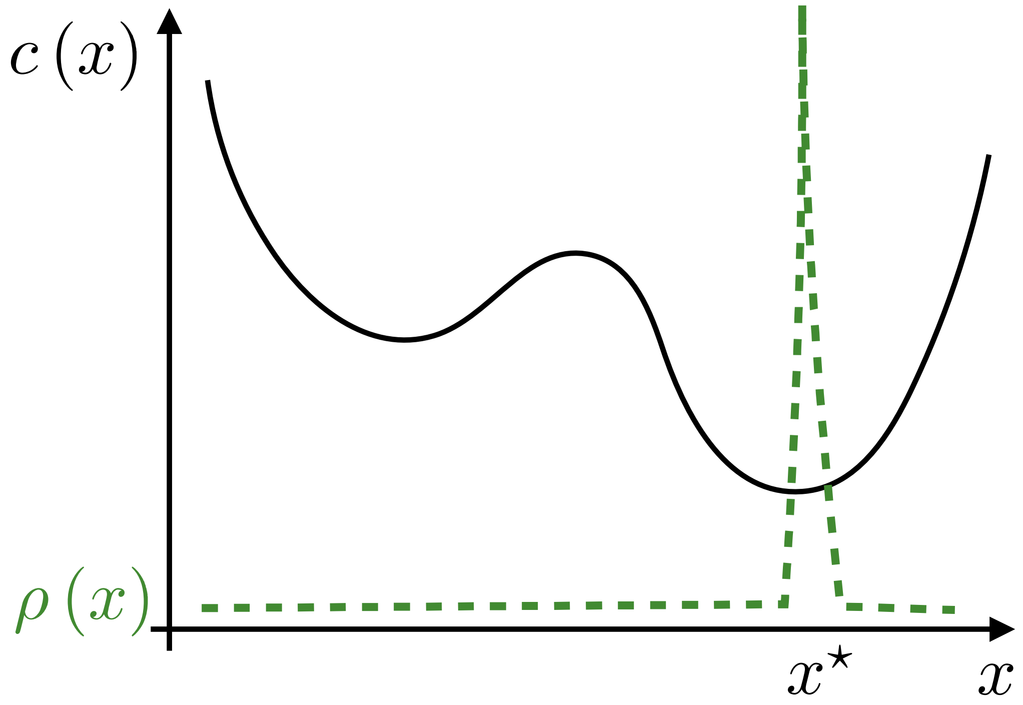
Simulated Annealing

Minimization by MCMC?

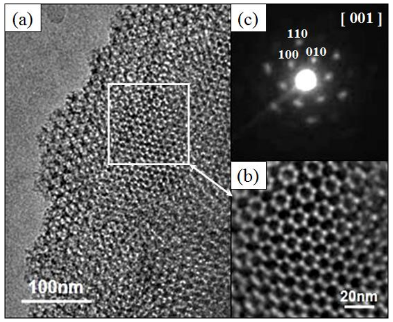
- Given a distribution $\rho(x)$, we can use MCMC to sample from it
- But we are given a cost function $c(x)$ to minimize instead
- Can we find a $\rho(x)$ which is concentrated in the minimum of $c(x)$?



Consider a chunk of metal

- High temperature ⟶ molten, glowing, etc. ≡ high internal energy

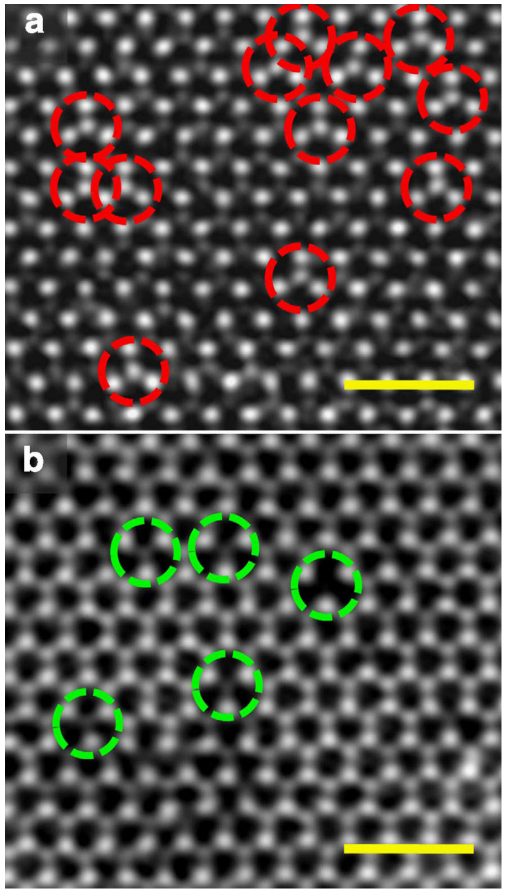
- Low temperature ⟶ stable crystalline structure ≡ low internal energy



- Thus: lowering the temperature ≈ minimizing the internal energy

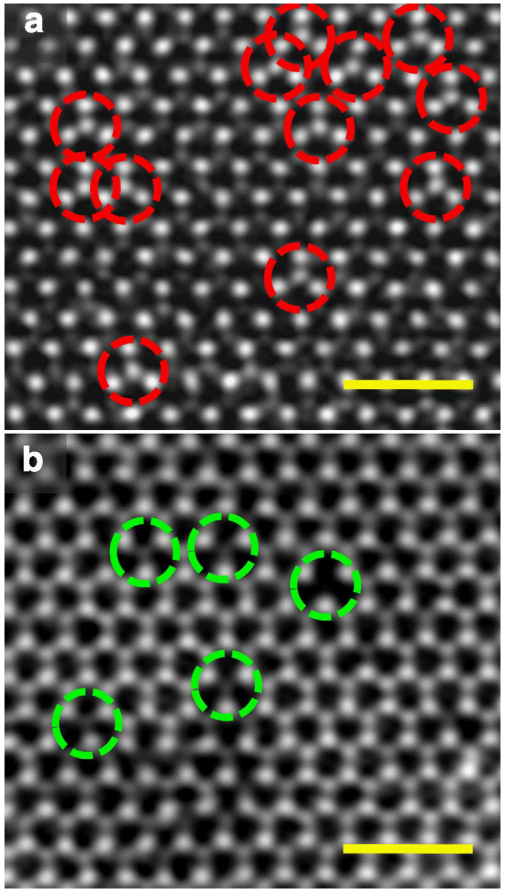
Not so fast

- Just lower the temperature to zero $\longrightarrow$ state of minimum energy?
- Not quite: do it too fast $\longrightarrow$ defects in the structure!
- One needs to let it cool slowly $\equiv$ **annealing**



Not so fast

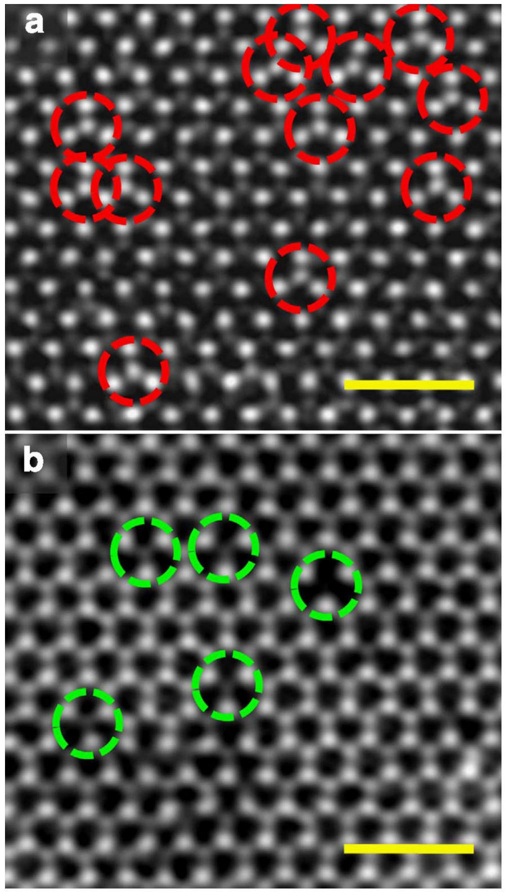
- Just lower the temperature to zero $\longrightarrow$ state of minimum energy?
- Not quite: do it too fast $\longrightarrow$ defects in the structure!
- One needs to let it cool slowly $\equiv$ **annealing**



Can we simulate the annealing process?

Not so fast

- Just lower the temperature to zero ⟶ state of minimum energy?
- Not quite: do it too fast ⟶ defects in the structure!
- One needs to let it cool slowly ≡ **annealing**



Can we simulate the annealing process?

Yes, with MCMC

Simulated Annealing: overview

- Inspired by an analogy with metallurgy (🤘):
  - Take a piece of metal, heat it until it's molten, then cool it gradually and let it settle down into a nice crystal structure
  - Internal energy $\Longleftrightarrow$ cost
- Based on stochastic processes and MCMC
- Similar to the greedy approach. But less greedy
- **Cons**:
  - it's heuristic, no (useful) guarantees
  - requires some judgment, trial and error, fiddling with parameters, ...
- **Pros**:
  - very general, not too hard to implement
  - often works quite well
  - interesting programming techniques

Simulated Annealing: the analogy

(very loose, somewhat sloppy)

- Temperature: T (positive)
- Atomic configuration: $x \in \mathcal{X}$, corresponding internal energy: $E(x)$
- Constant thermal agitation: the configuration x changes but the probability is stable. It depends on the temperature
  - $P_T(x) =$ "Prob. to find the atoms arranged as x (at temperature T)"
- Physically: the lower the temperature, the more likely that an x picked at random will have low energy
- This phenomenon has very little to do with physics and all to do with the mathematical relation between $P_T(x)$ and $E(x)$:
  - **Boltzmann distribution**: $$P_T(x) = \frac{1}{Z_T} e^{-\frac{1}{T} E(x)}$$
  - Z_T: normalization constant, to ensure $\sum_{x \in \mathcal{X}} P_T(x) = 1$

Main properties of the Boltzmann distribution

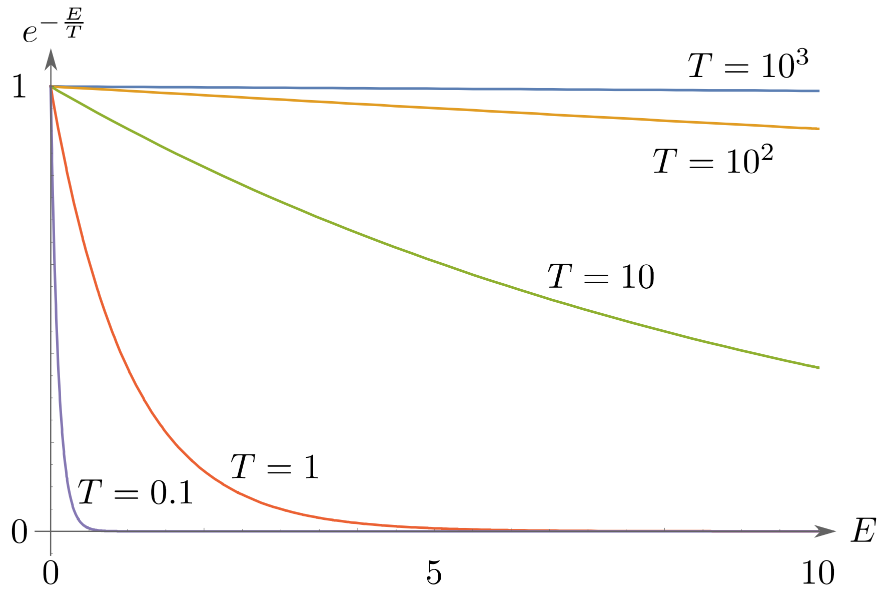
$$\boxed{P_T(x) = e^{-\frac{1}{T}E(x)} / \sum_{x' \in \mathcal{X}} e^{-\frac{1}{T}E\left(x'\right)}}$$

- Denominator
  - Constant
  - Could be hard to compute (many terms)
- Numerator
  - Monotonically decreasing with the energy
  - Almost constant for large T
  - Effect is more pronounced for small T

Main properties of the Boltzmann distribution

$$\boxed{P_T(x) = e^{-\frac{1}{T}E(x)} \Big/ \sum_{x' \in \mathcal{X}} e^{-\frac{1}{T}E(x')}}$$

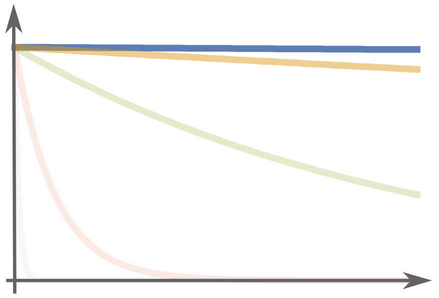
- Denominator
  - Constant
  - Could be hard to compute (many terms)
- Numerator
  - Monotonically decreasing with the energy
  - Almost constant for large T
  - Effect is more pronounced for small T

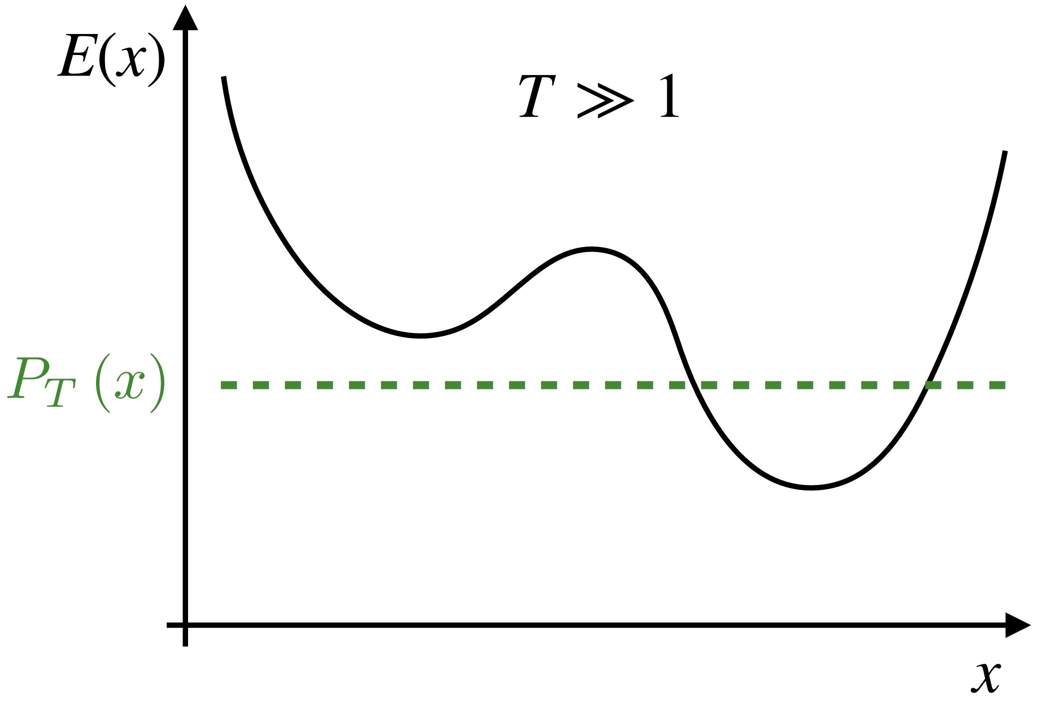


Main properties of the Boltzmann distribution

$$\boxed{P_T\left(x\right) = e^{-\frac{1}{T}E(x)} / \sum_{x' \in \mathcal{X}} e^{-\frac{1}{T}E\left(x'\right)}}$$

- Very high temperature: basically uniform

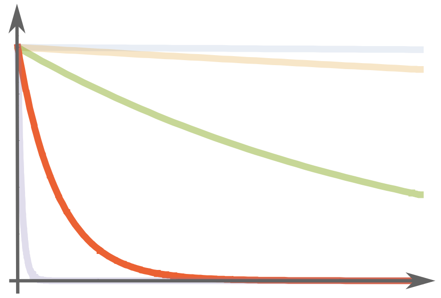


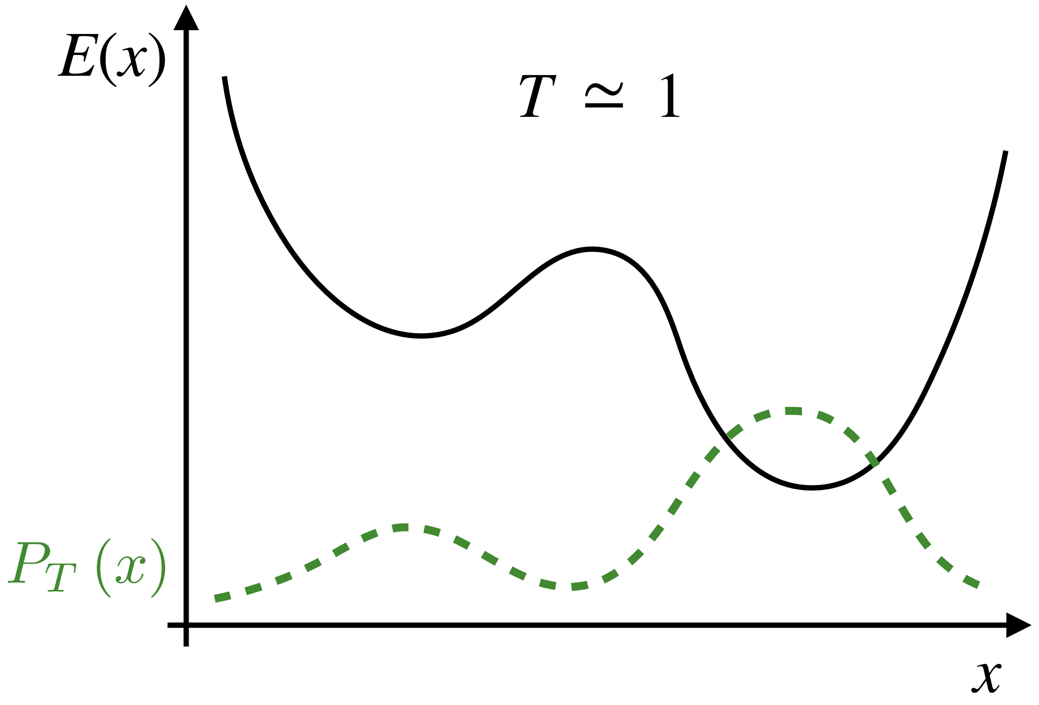


Main properties of the Boltzmann distribution

$$\boxed{P_T(x) = e^{-\frac{1}{T}E(x)} / \sum_{x' \in \mathcal{X}} e^{-\frac{1}{T}E(x')}}$$

- Intermediate temperature: lower energy has higher probability

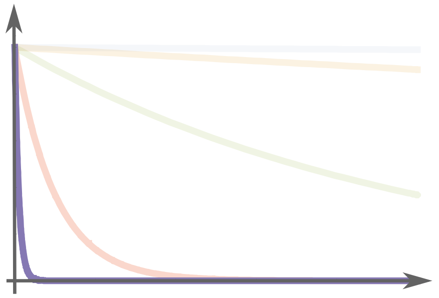


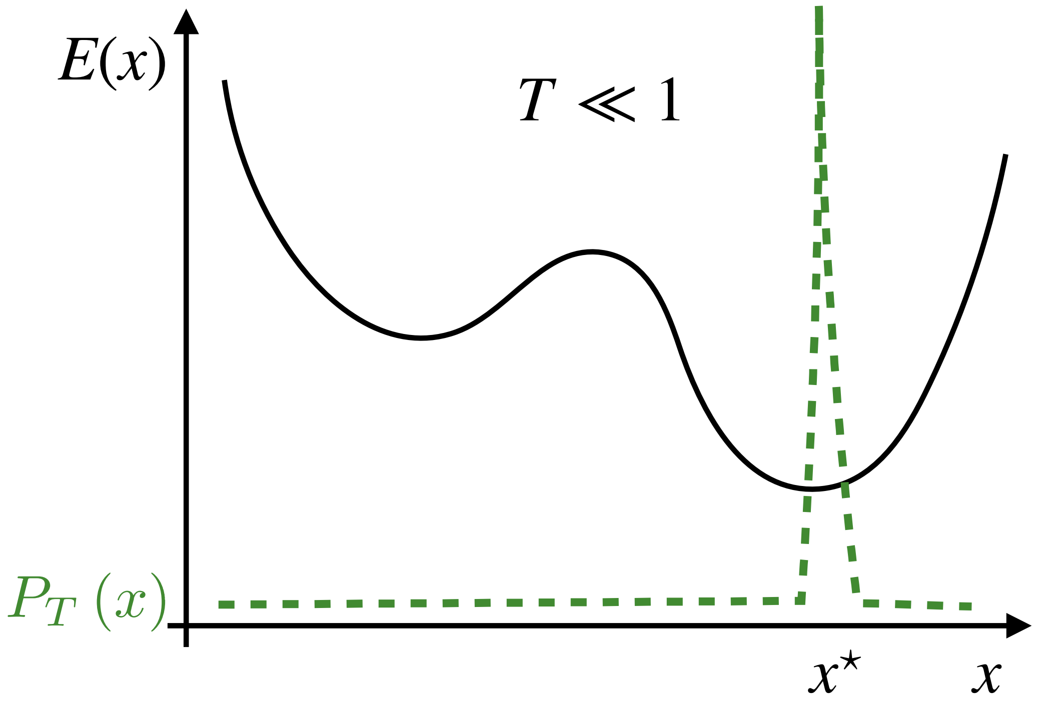


Main properties of the Boltzmann distribution

$$\boxed{P_T(x) = e^{-\frac{1}{T}E(x)} \Big/ \sum_{x' \in \mathcal{X}} e^{-\frac{1}{T}E(x')}}$$

- Very low temperature: probability concentrated on the global minimum





Main properties of the Boltzmann distribution

$$\boxed{P_T(x) = e^{-\frac{1}{T}E(x)} \Big/ \sum_{x' \in \mathcal{X}} e^{-\frac{1}{T}E\left(x'\right)}}$$

- We have written explicitly Z_T
- At very high temperature, $T \to \infty$, the argument of the exponential goes to 0 and all the x have the same probability: $P_\infty(x)$ is uniform
- At very low temperature, $T \to 0$, the probability concentrates on the global minimum of $E(x)$. Say that $x^\star$ is the optimum (say it's unique), and consider

$$P_T(x) = e^{-\frac{1}{T}(E(x)-E(x^\star))} \Big/ \sum_{x' \in \mathcal{X}} e^{-\frac{1}{T}\left(E\left(x'\right)-E(x^\star)\right)}$$

Only the terms with $x \approx x^\star$ survive at low T

- In between, we have intermediate situations: a higher probability on low-energy states, a lower probability on high-energy states. But the low-energy states are often few so the probability of getting some medium energy state is larger just because there are more of those, even if individually they are less likely.

Optimization idea

- You have your cost function $c(x)$ that you want to optimize
- Use $c(x)$ as an energy in the Boltzmann formula. For any T, obtain $P_T(x)$
- For any given T:
  - Sampling from $P_T(x)$ directly is hard because Z_T is hard to compute (sum of an exponential number of terms)
  - But we know how to do MCMC and sample from $P_T(x)$ using M.-H., as long as we know how to choose the proposals. Miracle of M.-H.: no need to compute Z_T!
- Choose the proposals in a way that makes sense for the problem at hand, i.e. from any given configuration propose moves to its neighbors. (Some arbitrariness here.)
- Can we sample directly at $T \to 0$ and get the global minimum then?
  - That's precisely the greedy algorithm that we wrote!
  - So, sometimes yes, for easy enough problems. Often doesn't work, gets trapped in local minima
- Simulate the gradual cooling instead!

Sampling from the Boltzmann distribution with MCMC

- Assume symmetric proposals (not quite necessary, it's just simpler)
- Expression of the M-H acceptance rule when $\vec{\rho} \equiv P_T(x)$ and you have proposed a move from the current x to a candidate x':

$$A(x \to x') = \min\left(1, \frac{\rho_{x'}}{\rho_x}\right) = \min\left(1, \frac{\frac{e^{-\frac{1}{T}c(x')}}{Z_T}}{\frac{e^{-\frac{1}{T}c(x)}}{Z_T}}\right) = \min\left(1, e^{-\frac{1}{T}\left(c(x')-c(x)\right)}\right)$$

A little bit of notation mix-up...

The cost is our "energy"

The Z_T canceled out! (very important)

$$\boxed{A(x \to x') = \min\left(1, e^{-\beta \Delta c(x \to x')}\right)}$$

Defined:
$\beta = 1/T$ **(note: never negative!)**
$\Delta c(x \to x') = c(x') - c(x)$

- **When $\Delta c \leq 0$ (the move reduces the cost) the $\min$ is 1, acceptance is certain**
- **When $\Delta c > 0$ (the move increases the cost) the $\min$ is < 1, acceptance is likely only if move not-too-bad (depends on β, small$\rightarrow$everything goes, large$\rightarrow$picky)**

Simulate the gradual cooling

- Start at small β (large T) and sample (run MCMC long enough). This is easy, $P_T(x)$ is almost uniform, acceptance rate is high. But you get x with pretty bad $c(x)$ typically
- Increase β (lower T) and keep going, starting from where you had left and running for more iterations. Now the samples get better, they have lower $c(x)$ typically.
- Increase β again and keep going, they get better still. Increase β again, etc. Every time, start from where you just left.
- By the point you get to $\beta \to \infty$ $(T \to 0)$, you're now running the greedy algorithm. But now you are initializing it from a configuration sampled from a previous $P_T(x)$ which should be already pretty good!
- Ideally, in the limit of infinite MCMC samples and infinitely slow annealing you'd be guaranteed to find the optimum. This of course is not helpful, one might as well check all configurations one at a time...
- There are better guarantees actually, which still requires exponential number of steps
- In practice you cut it short and approximate (heuristic approach).

Reminder: inverse temperature
$\beta = 1/T$ **(must go from** $\simeq 0$ **to** ∞ **)**

The Simulated Annealing scheme: setup

- We are given a function $c(x)$, defined over a finite discrete set $\mathcal{X}$, that we wish to optimize
- Choose a way to represent x in the computer
- Choose a way to produce a random initial configuration
- Choose a (symmetric) proposal scheme: a function that, given an x, proposes a similar but different x', at random.
  - Requirements: x' cannot be the same as x; connectedness (must be able to cover all $\mathcal{X}$ when jumping around); aperiodicity (must be random enough)
  - Symmetric means: the chance that x' gets proposed given x must be the same as the chance that x gets proposed given x'.
- Choose an "annealing schedule": A sequence of increasing betas (decreasing temperatures) and corresponding number of MCMC iterations, $[(\beta_0, \mathrm{it}_0), (\beta_1, \mathrm{it}_1), \ldots, (\beta_{L-1}, \mathrm{it}_{L-1})]$
  - We'll speak later about this. Lots of trial and error involved here.

The Simulated Annealing scheme: basic pseudocode

- **INPUTS**: optimization problem over X; annealing schedule [(β_i,it_i) for i in {0,..,L-1}]
- **ALGORITHM:**

```
initialize x in X at random
for i in 0,1,…,L-1:   # annealing step
    set β = β_i           # current β (inverse temperature)
    for t in 1,…,it_i:  # perform this many MCMC steps
        given x, produce a proposed move x'
        compute the cost difference Δc(x→x') # c(x')-c(x)
        compute the acceptance rate A(x→x') from Δc(x→x') and β
        with probability A(x→x'):
            set x = x'  # accept the proposal
return x
```

Simplifications and extensions

- We are going to use a simplified schedule:
  - Give only two values `β0` and `β1` (first and penultimate)
  - Give the total number of annealing steps `L`
  - The first `L-1` steps are equally spaced between `β0` and `β1`. The last one is performed at `β=np.inf`
  - All the annealing steps will have the same number of MCMC iterations
- Instead of returning the last configuration, keep the best one seen throughout the process (for some problems, you might even want to stop the algorithm as soon as you hit some desired cost, e.g. 0).
- We'll need to monitor and output the "acceptance rate": at every annealing step, what fraction of the proposed moves have been accepted? This will be often crucial for setting the parameters. To estimate this somewhat reasonably, we'll need at least 100 MCMC steps per annealing step then...

How to set the parameters?

- The only thing that is valid in general is that `0<β0<β1` because nothing else would make sense in our context.
- Apart from that, there are mostly only rules of thumb.
- Everything depends on the problem at hand. For a given family of problems, things may change a lot with the size (e.g. number of nodes in a graph...) and the characteristic.
- In general, the longer and the slower you let the annealing proceed, the better the results.
- But that takes time. So you'll want to do things just as slowly as necessary for the problem at hand. That requires tuning by trial-and-error.
- You may want to set `β0` such that at the beginning most moves (but not 100%) are accepted, say somewhere between 30% and 80%.
- You'll want to see the acceptance rate decrease gracefully, together with the cost, and reach a few percent by the last step.
- Acceptance rate too high until the end → time wasted in roaming around instead of optimizing (exception for equal-cost states...)
- Acceptance rate too low too early → the system is (nearly) stuck in a local minimum
- All else failing, do more iterations (MCMC or anneal steps, doesn't matter much)