Lecture 5:

Global Optimization and Simulated Annealing

Week of January 30, 2023

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

Spring 2023

Prof. Teresa Head-Gordon

Departments of Chemistry, Bioengineering, Chemical and Biomolecular Engineering

SGD is based on gradients taken over subsets of the learning data (mini-batch), thereby only serving as an approximation to the true loss function which are the test examples! This is equivalent to adding noise to the derivatives. While not being a pure local minimization, it can jump minima to become a proxy global minimization approach (more on that today!)

$$\vec{w}_{t+1} = \vec{w}_t - \eta \vec{\nabla} f(\vec{w}_t)$$

AdaGrad is adaptive SGD

$$w_{\mathrm{t+1,i}} = w_{t,i} + \frac{\eta}{\sqrt{D_{t,ii} + \epsilon}} d_t(w_{\mathrm{i}})$$

$$D_{t,ii} = Diag\left[\frac{1}{t}\sum_{\tau=1}^{t} \nabla_i f(\vec{w}_\tau) \cdot \nabla_i f(\vec{w}_\tau)^T\right]$$

Adagrad's main benefits is that it eliminates the need to manually tune η can be set to almost any small value. Adagrad's main weakness is its accumulation in the denominator which keeps growing during training (squared gradients are positive).

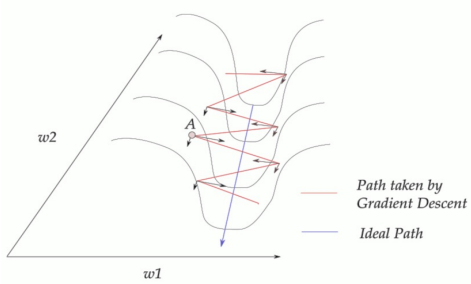
Review of Last Lecture

We showed that many of the optimization approaches used in machine learning for a standard quadratic loss function were based on variations of the steepest descent, conjugate gradients, and 2nd order algorithms, with their own language.

SGD with momentum (**SGDM**) is an approximate and stochastic version of CG with no true H-orthogonality, but just short-term to long-term memory of productive components of previous search directions to move well along difficult trough functions.

$$d_t(\vec{w}) = -\nabla f(\vec{w}_{t-1}) - \gamma d_{t-1}(\vec{w})$$

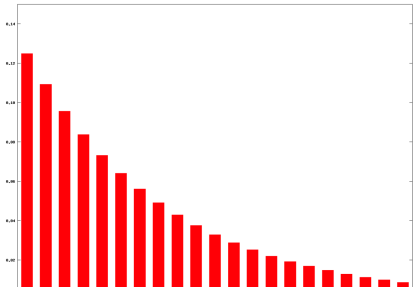
$$\vec{w}_{t+1} = \vec{w}_t + \eta d_t(\vec{w})$$



A similar idea is applied to the preconditioner of AdaGrad which gives **RMSProp**. I.e. we adapt the preconditioner $\underline{D}(\vec{w}_t)$ so that we only keep a finite history of the previous squared gradients through the use of an exponential moving average (EMA)

$$D'_{t,ii} = \alpha D'_{t-1,ii} + (1-\alpha) diag[\nabla_i f(\vec{w}_\tau) \cdot \nabla_i f(\vec{w}_\tau)^T]$$

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{D'_{t,ii} + \epsilon}} \overrightarrow{\nabla_i f(\vec{w}_t)}$$



Review of Last Lecture

AdaDelta recognizes a "unit correction" that is not taken into account by RMSProp. AdaDelta eliminate the learning rate hyperparameter η by creating a new EMA

$$\Delta^2_{\tau,i} = \alpha\Delta^2_{\tau-1,i} + (1-\alpha)\Delta^2$$

to get a "unit correct" method for weight parameter updates.

$$w_{t+1,i} = w_{t,i} - \frac{\sqrt{\Delta_{t,i} + \epsilon}}{\sqrt{D'_{t,ii} + \epsilon}} \vec{\nabla}_i f(\vec{w}_t)$$

Adam. In addition to storing an exponentially decaying average of past squared gradients like AdaDelta and RMSprop, keeps an exponentially decaying average of past gradients or momentum.

$$d_{t,i} = \gamma\, d_{t-1,i} + (1-\gamma)\, \vec{\nabla}_i f(\vec{w}_t)$$

that creates a history of previous directions analogous to SGDM!

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{\widehat{D}_{t,ii} + \epsilon}} \hat{d}_{t,i}$$

Review of Last Lecture

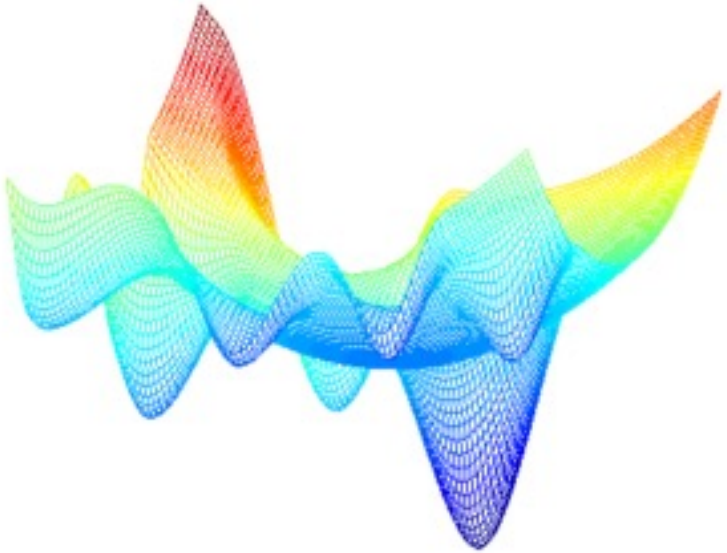
Goals for Today's Lecture

In this new Section of the course we consider several meta-heuristic algorithms for performing optimization to find better solutions than just a random local minimum. In particular, we consider the general problem of global optimization, that will culminate in artificial neural networks.

- Finding the global minimum of the ML. A feed-forward artificial neural network (ANN) with one hidden layer can approximate any smooth function to an arbitrary degree of accuracy given a sufficient number of "neurons". Thus, ANNs are said to be universal approximators of any function and we thus want the "best" minimizer of that function.

- Using global optimization to improve ML: Global optimization is important for hyperparameter tuning, feature selection, and algorithmic configuration to optimize the generalization capability of ML.

- Using ML to help metaheuristic global optimization. The parameters of such algorithms could also be tuned using ML.



Global Optimization

Goals for Today's Lecture

In this new Section of the course we consider several meta-heuristic algorithms for performing optimization to find better solutions than just a random local minimum. In particular, we consider the general problem of global optimization, that will culminate in artificial neural networks.

Exact or Deterministic Methods: some type of guarantee that global optimum is (or is not) found.

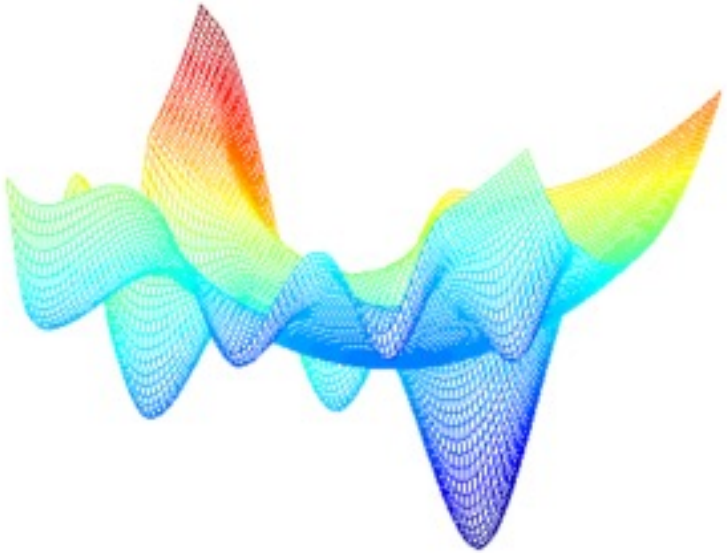
> Dynamic programming (partitioning methods), Branch and Bound

Meta-heuristic or Stochastic Methods: guarantees are formulated in probabilistic terms (global not guaranteed or weakly so)

> Simulated annealing, genetic algorithms, chemical reaction optimization, artificial neural networks

It is important to not place a strong judgment on exact vs. heuristic methods. Most problems of complexity are NP-complete (exponential time complexity), so we content ourselves with "better" solutions!

Today we consider a very popular meta-heuristic stochastic method, simulated annealing.



Global Optimization

Out of many possible good solutions (local minima) we are looking for the best solution (global minimum)

While the Apostle Paul may not have realized it, his advice to Thessalonica is an exact global optimization solution!

"Consider everything. Keep the good. Avoid evil whenever you notice it." (1 Thess. 5:21-22)
http://solon.cma.univie.ac.at/glopt.html

Do Forever "Consider everything"
  Guess $\vec{x}$
  $f(\vec{x}) = E_{curr}$
  If $E_{curr} < E_{best\_so\_far}$ *then*
    $E_{best\_so\_far} = E_{curr}$ "Keep the good."
  Else
    continue "Avoid evil whenever you notice it"
  Endif
End do

Global Optimization

Traveling Salesman Problem

Given a collection of cities and cost of travel between each pair, the TSP is to find the cheapest route to visiting all of cities and returning to your starting point.

- Richard Karp at UC Berkeley proved that the TSP is NP-Complete (exponential or factorial complexity with increasing #cities)

- The TSP was featured in a contest run by Proctor and Gamble in 1962. The challenge problem had 33 cities. There was a tie between many people who found the optimum. An early TSP researcher, Professor Gerald Thompson of Carnegie Mellon University, was one of the winners.
- Groetschel (1977) found the optimal tour of 120 cities from what was then West Germany.
- Padberg and Rinaldi (1987) found the optimal tour of 532 AT&T switch locations in the USA.
- Groetschel and Holland (1987) found the optimal tour of 666 interesting places in world.
- Padberg and Rinaldi (1987) found the optimal tour through a layout of 2,392 points obtained from Tektronics Incorporated.

Traveling Salesman Problem

- Applegate, Bixby, Chvátal, Cook (1998) found the optimal tour of the 13,509 cities in the USA with populations greater than 500.
- Applegate, Bixby, Chvátal, and Cook (2001) found optimal tour of 15,112 cities in Germany
- Applegate, Bixby, Chvátal, Cook, Helsgaun (2004) found the optimal tour of 24,978 cities in Sweden.
- In April 2006, Applegate et al solved and then published in 2008 "Certification of an optimal TSP tour through 85,900 cities", and currently stands as verified record for solved TSP!

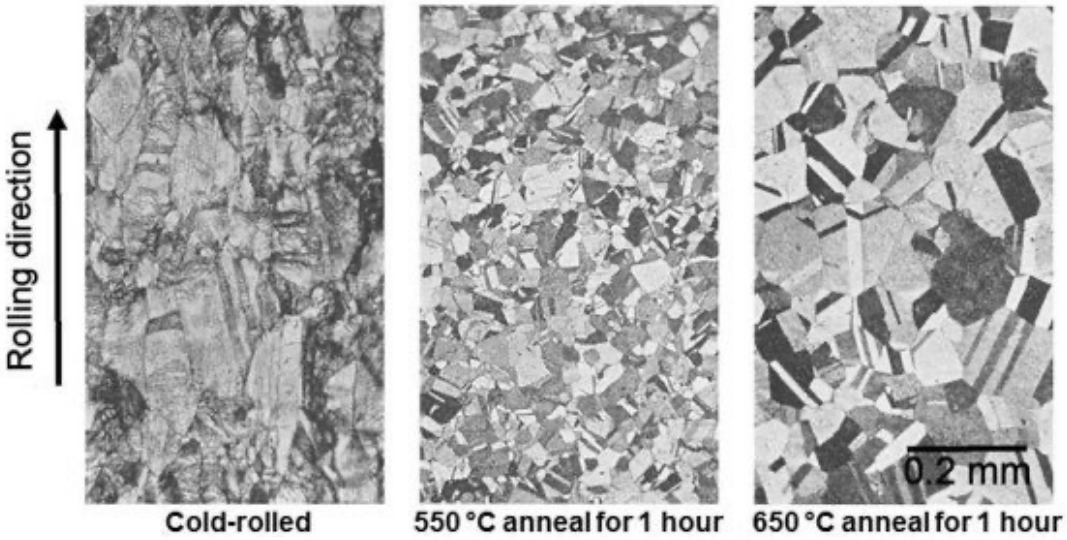
The current challenge is a 1,904,711-city tour throughout the world- not provable as deterministic Turing machine (any computer) can't determine optimal value! Each city is specified by latitude and longitude; cost of travel between cities is an approximation to great circle distance on Earth.

- Tour of length 7,515,772,107 was found in 2018 by Helsgaun, and improves upon previous records. An earlier record tour of length 7,518,425,642 was found by Nguyen, Yoshihara, Yamamori, and Yasunaga in 2003, which used, in part, a parallel hybrid genetic algorithm.

Simulated Annealing

Simulated annealing is motivated by the way molten metals crystallize through annealing. The physical process of annealing starts with the material at an initially high temperature and therefore disordered with large entropy, and then it is slowly cooled so that the system at any time is close to thermodynamic equilibrium.

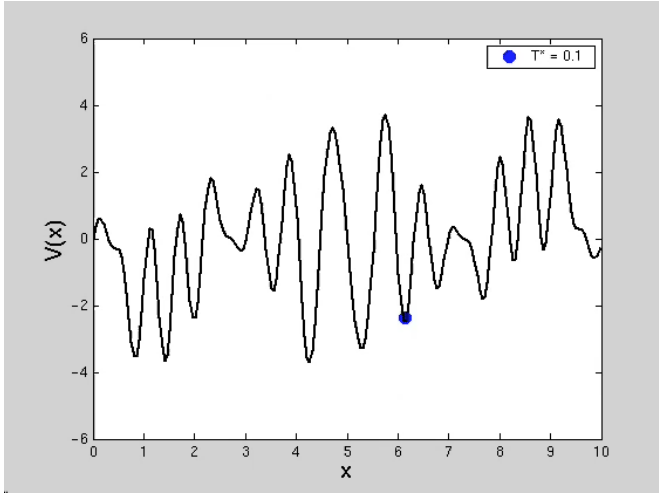
- As cooling proceeds, the system becomes more and more ordered and eventually approaches a perfect crystalline ground state at T=0 which corresponds to the lowest potential energy state (global minimum) with zero entropy.
- But if initial temperature of system is too low or cooling is done insufficiently slowly, the system may become quenched, forming defects in the crystal.

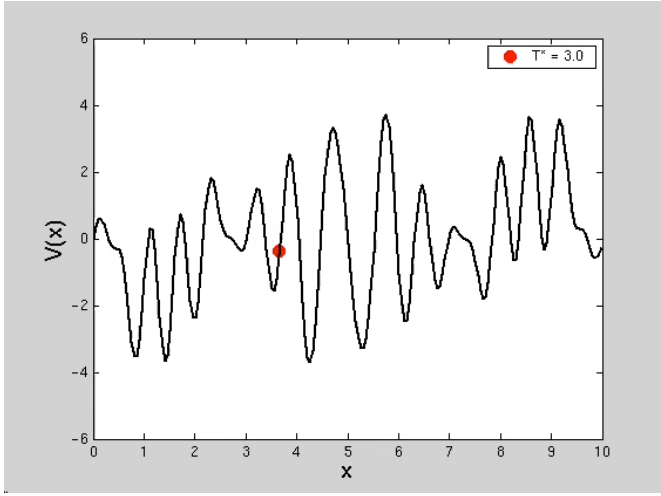


- The metaphor is to cool T the quickest we can but still having the guarantee that no trapping in any local minimum will occur.

Simulated Annealing

"Search directions" are determined by a hyperparameter T, as a metaphor to temperature. T acts as a source of stochasticity, convenient for detrapping from local minima through hill climbing. Simulated Annealing involves the following general procedure





Program Simulated Annealing

Initialize search variable(s)

Do I=1,N_{temp}

  Cooling schedule (changes T)

  Do J=1,Nequil

    Update search variable(s)

    Global criteria (global opt?)

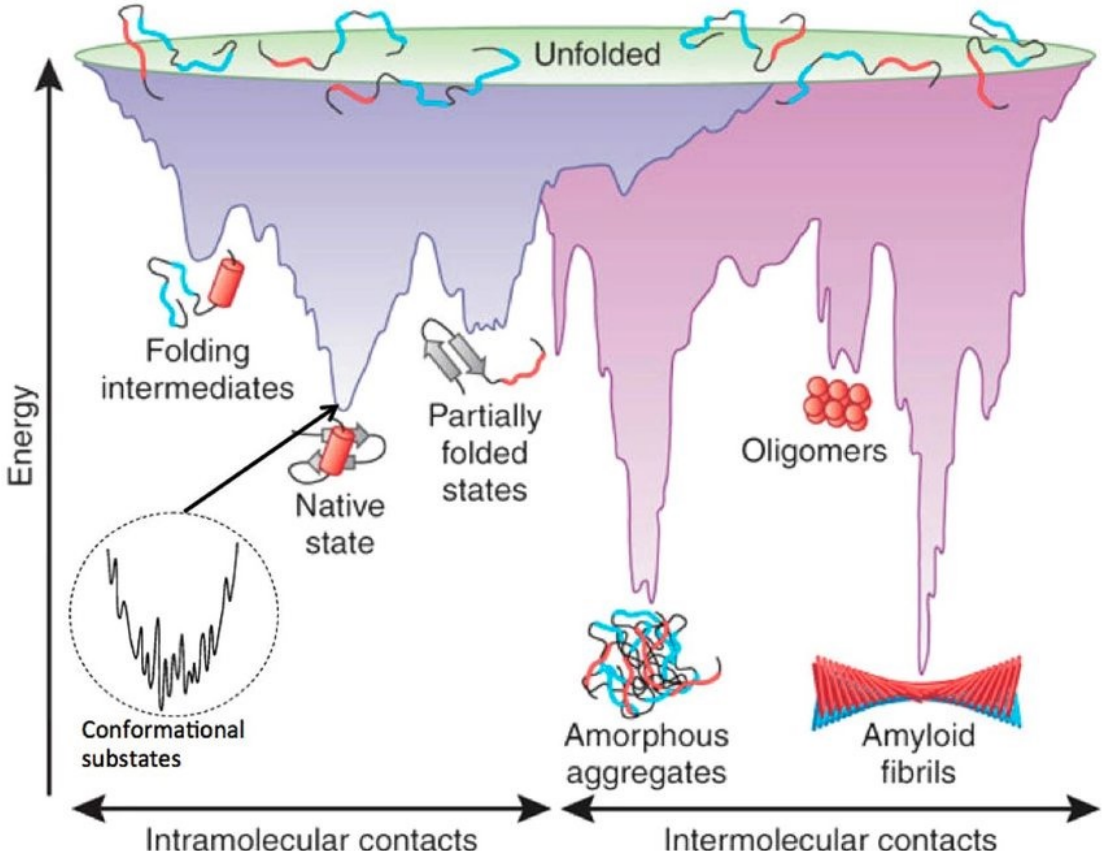
  End do

End Do

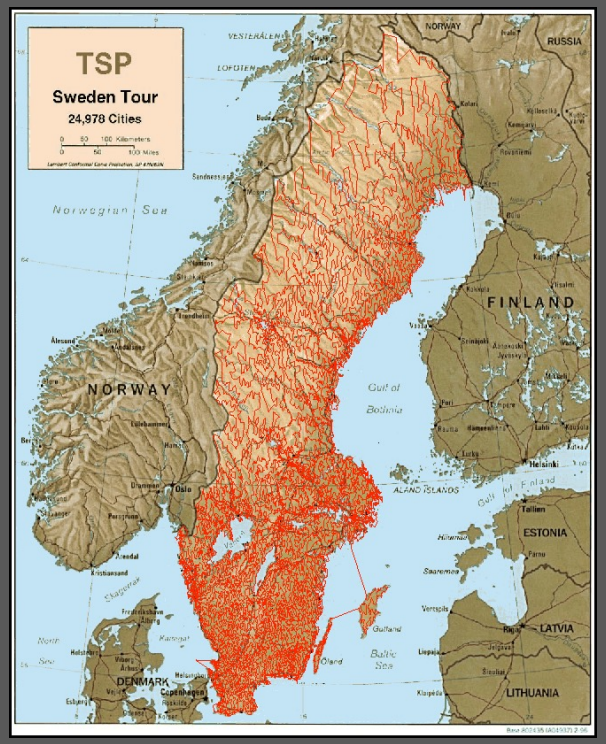
Stop

Lets break these down in sequence

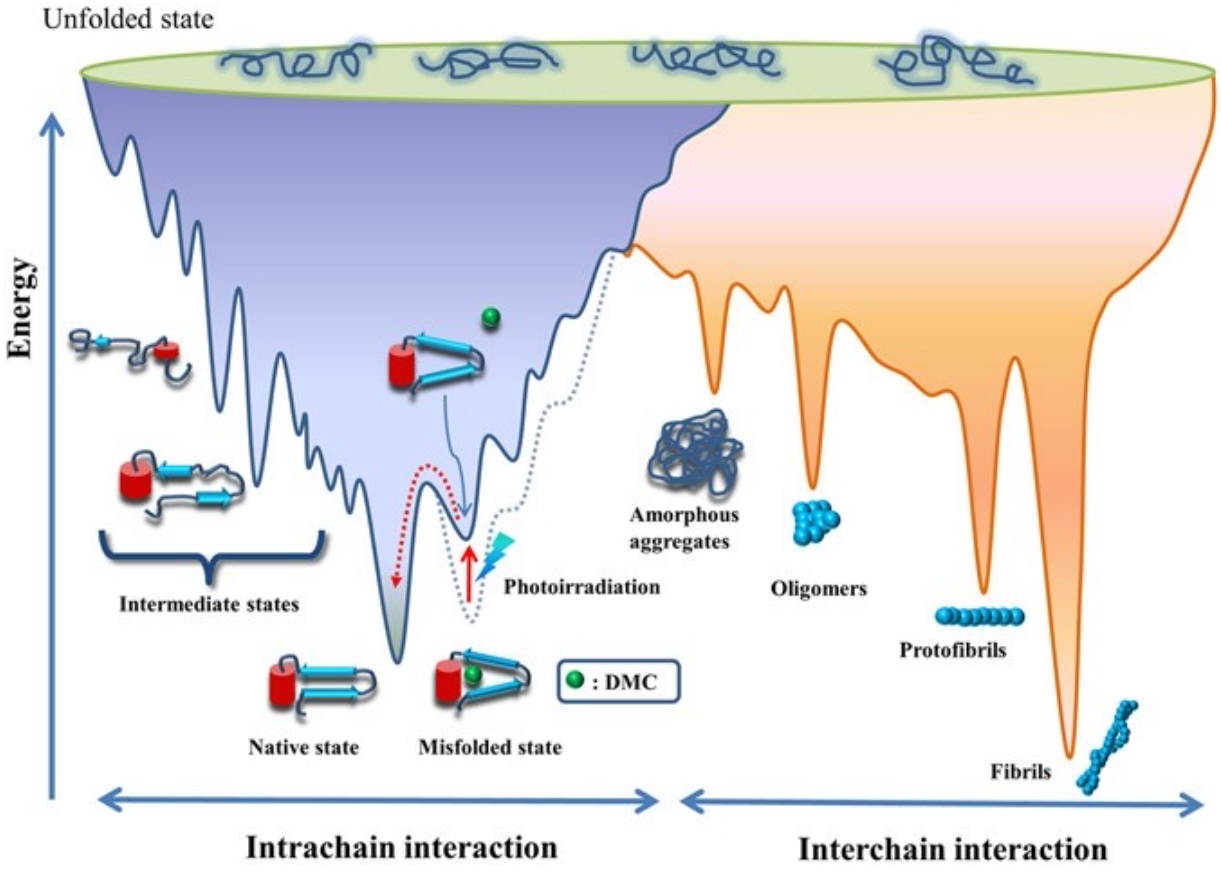
The Components of Simulated Annealing



One of the primary strengths of a meta-heuristic algorithm like SA is that it can be applied to arbitrary GO problem.



Search Directions and Global Minimum Criteria



In protein folding, the search variables are the cartesian positions of atoms in the protein structure.

- I need a set of moves that generate trial "configurations"

What is the cost function?

(1) Global free energy minimum, Gmin, is functional native state:

- Gmin= H-TS

(2) # conformational states at free energy minimum ~1

- S=klnW=0

(3) Enthalpy is largely potential energy

- H=E+pV pV~0

Therefore searching for global *potential energy* E minimum

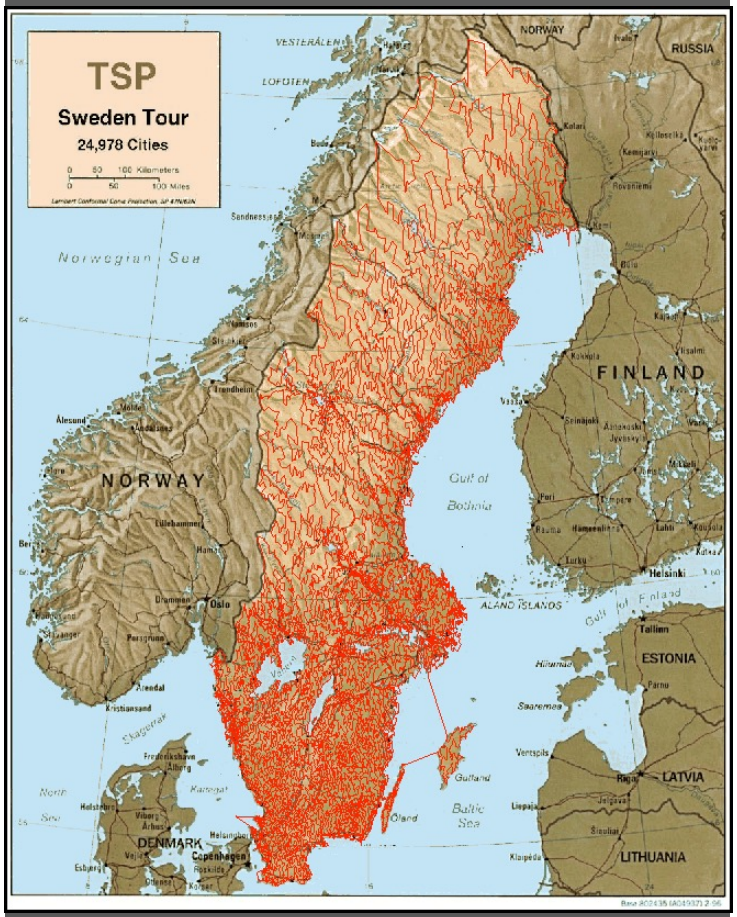
Search Directions and Global Minimum Criteria

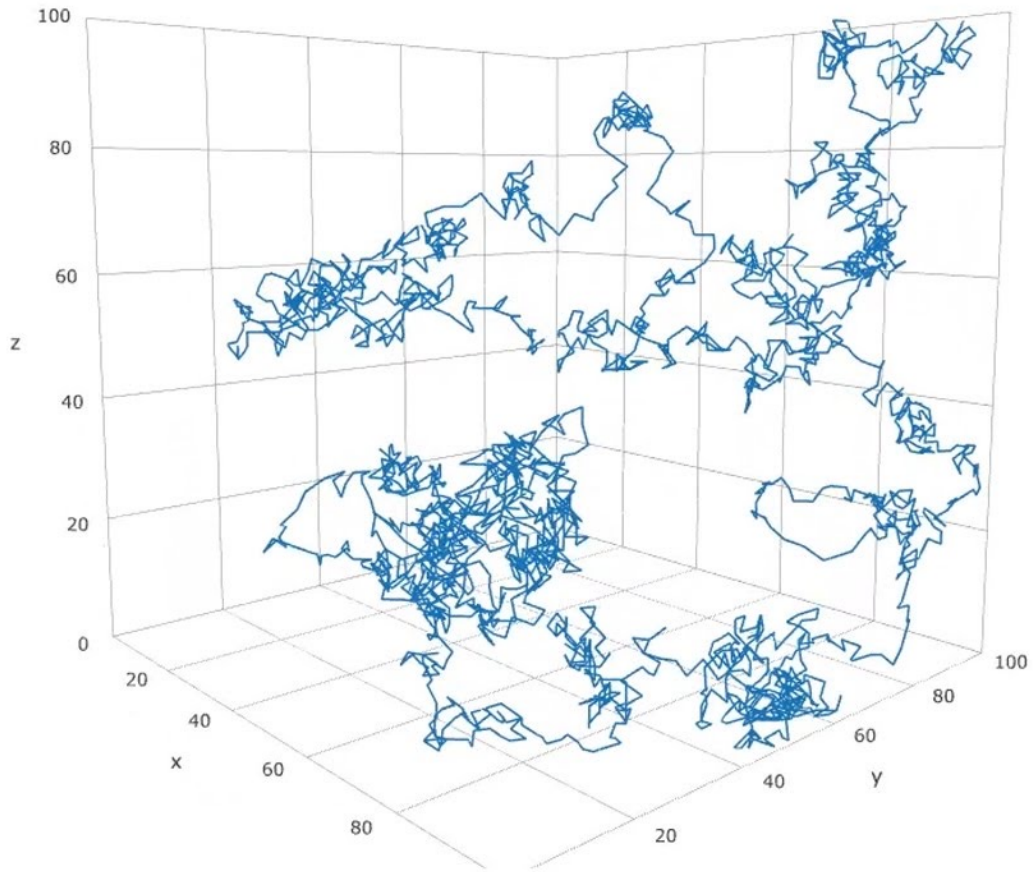
1. start with a path which lists all of the cities in random order selected. The length of the path is then calculated (can be a function of distance and time).
2. a smaller segment of the total path is selected, with its start and end cities chosen at random. Then, we perform a "move":

reverse: an alternative path is generated in which the cities in the chosen segment are reversed in order of visit.

transport, the segment is clipped out of its current position in the path and spliced in at a randomly chosen point in the remainder of the path.

3. now calculate the length of the trial path and compare to the original path to estimate the *cost difference*.





Update Search Variables

We update variables such that we are finding better solutions according to our cost function.

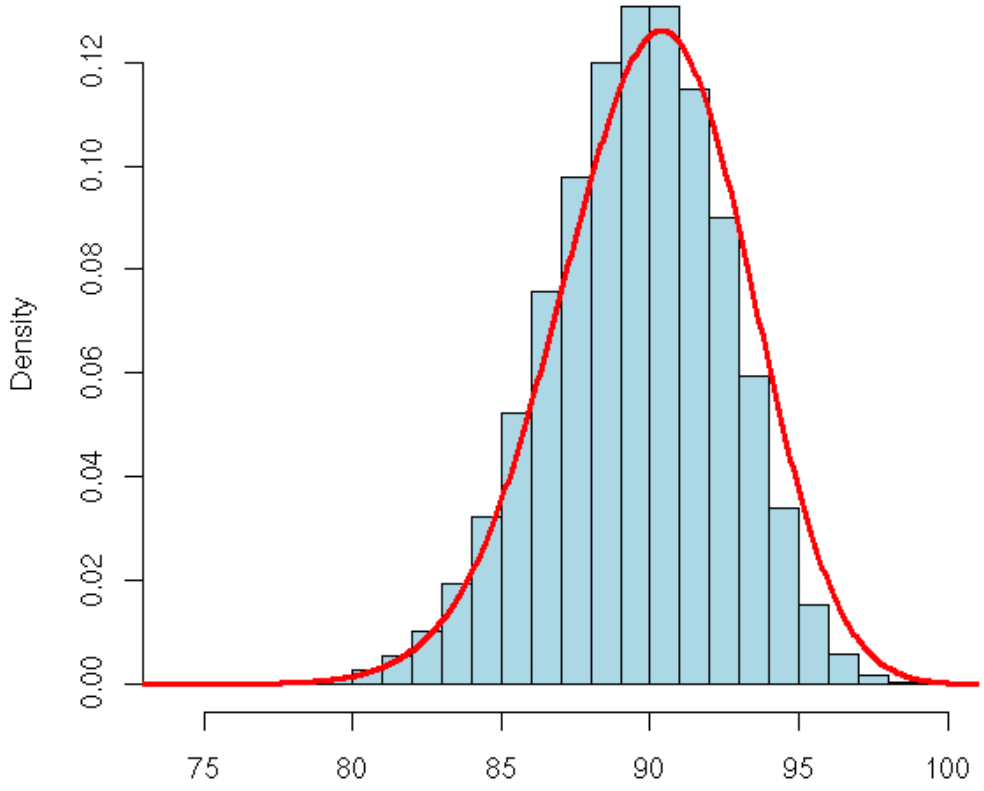
In the classical SA algorithm, we propose stochastic trial moves drawn from a Gaussian (i.e., a random walk) in the neighborhood of current state.

- Trial move is always accepted if it decreases the cost function, and we update the search variables to their new values. We can also hill climb due to available kinetic energy, and it is conditionally accepted based on canonical-ensemble Boltzmann.

- If the T is fixed, this procedure is the well known Metropolis Monte Carlo algorithm for simulating thermostatistical equilibrium

Gaussian Generating Function

$$P(m,N,p) = \frac{N!}{m!(N-m)!} p^m (1-p)^{N-m}$$



Inebriated person starts at lamp post and takes N unit length steps in random directions. Let's restrict this to 1D so that s/he takes *m* steps to right and *N-m* steps to the left. What is the probability that the walker has made *m* steps to the right after N steps?

(a) For each step there are only two outcomes: R or L (assume equal probability, p=0.5)

(b) The next step is completely independent of all previous steps:

(c) Limiting Binomial for large N → Gaussian.

- Search variables are explored with random steps

$$\vec{w}_{i+1}^{N} = \vec{w}_{i}^{N} + random\ vector$$

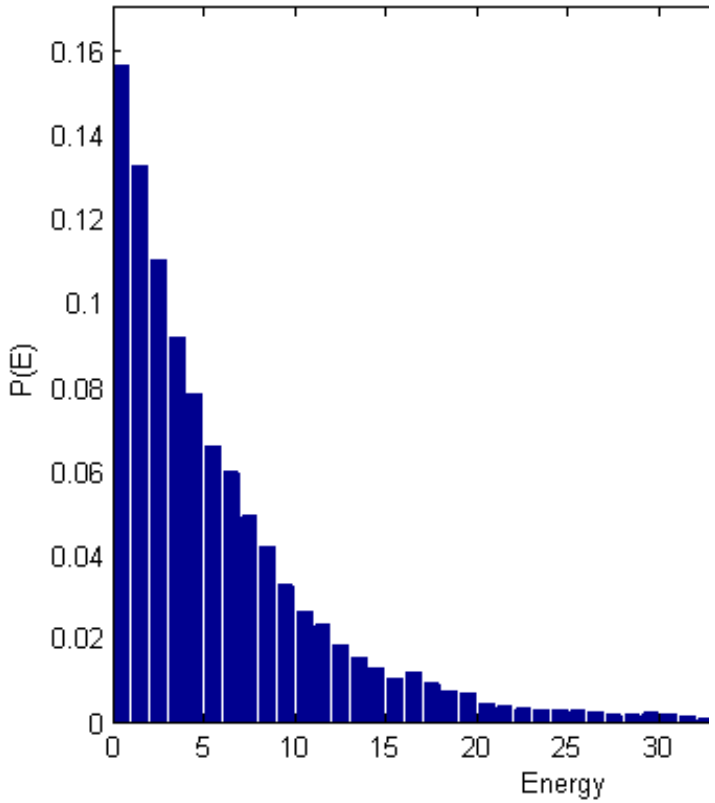
Boltzmann Probability

Imagine we have constructed a large number of M trials of generating many such vectors with a cost function that we can evaluate. Since the limiting distribution, the Boltzmann probability, corresponds to equilibrium under constant N, V, and T conditions, we need to accept these trial search vectors under this condition

$$p_{NVT}(i) = exp(-\beta E(\vec{R}_i^N))/Z$$

And hence this transition probability from old to new configuration $\pi(i \rightarrow j)$ must be constructed so it can't destroy this equilibrium (condition of microscopic reversibility)

$$p_{NVT}(i)\pi(i \rightarrow j) = p_{NVT}(j)\pi(j \rightarrow i)$$



Update Search Variables: Metropolis Algorithm

This means that random steps to left are equally probable as random steps to the right, or $\pi(i \to j) = \pi(j \to i)$ so that we accept new state as

$$acc(\mathrm{i} \to \mathrm{j}) = \min\left[1, \frac{p_{NVT}(j)}{p_{NVT}(i)}\right]$$

- Accept move if probability of j state is greater than i state

$$p_{NVT}(j) > p_{NVT}(i) \qquad acc(i \to j) = 1$$

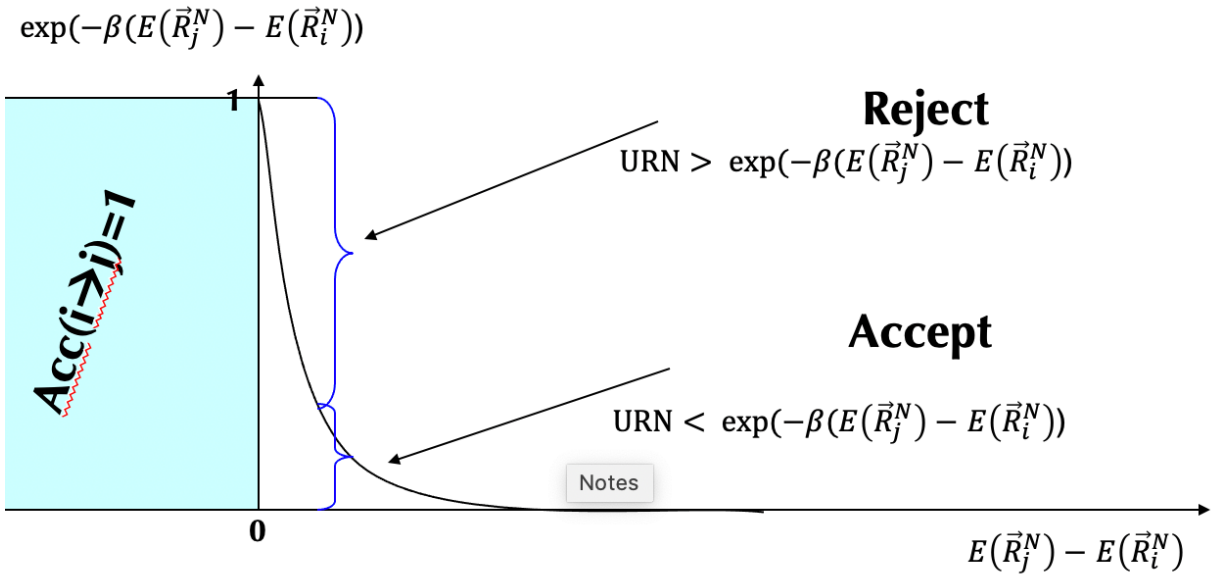
- Otherwise, accept move conditionally with relative probability

$$p_{NVT}(j) \leq p_{NVT}(i) \qquad acc(i \to j) = \frac{p_{NVT}(j)}{p_{NVT}(i)}$$

$$acc(i \to j) = \frac{exp(-\beta E\left(\vec{R}_j^N\right))/Z}{exp(-\beta E\left(\vec{R}_i^N\right))/Z} = exp(-\beta(E\left(\vec{R}_j^N\right) - E\left(\vec{R}_i^N\right))$$

- Note $\beta = 1/k_bT$ means that we can tolerate higher cost function moves ($\Delta E > 0$) up to ~k_bT

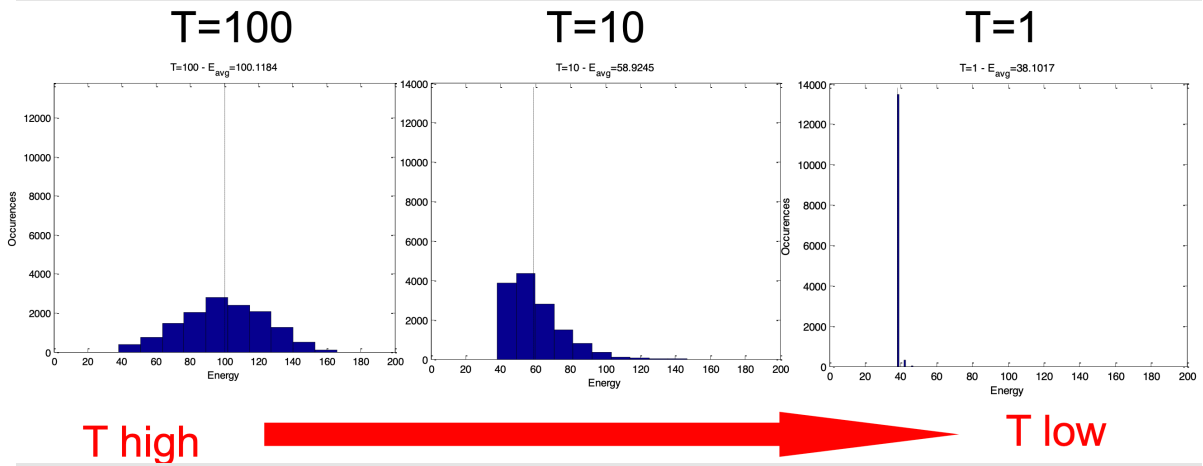
Metropolis Monte Carlo Acceptance



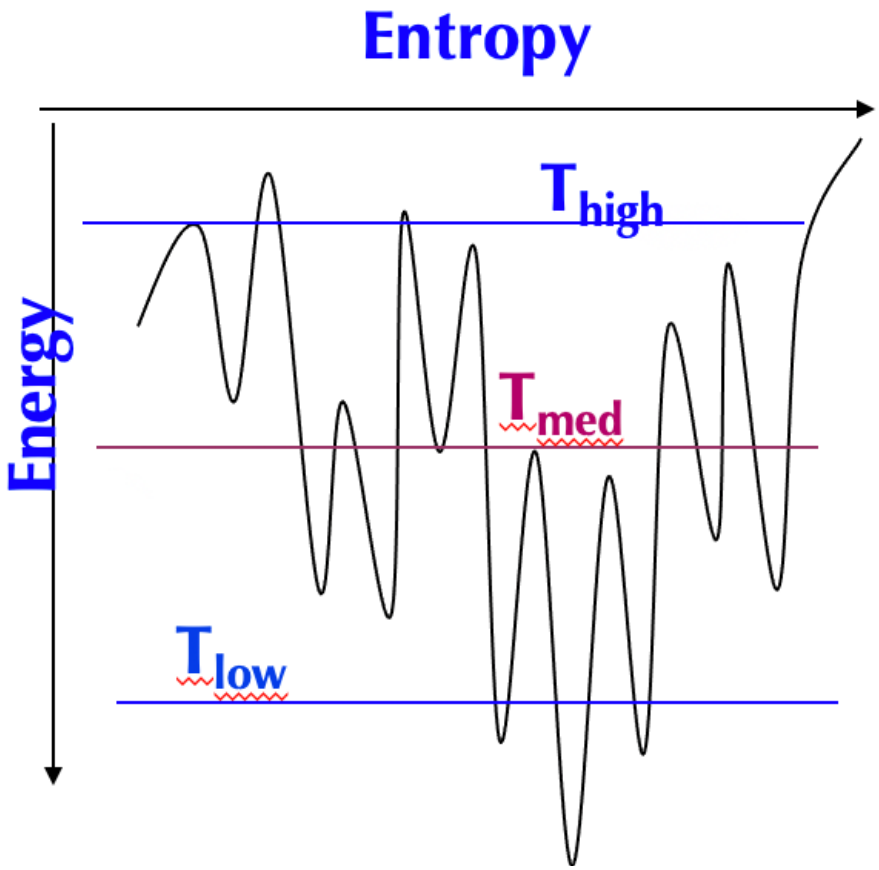
Geman and Geman showed that for CSA, a necessary and sufficient condition for having probability 1 of ending at global minimum is that the temperature decreases logarithmically with the simulation time.

Metropolis Monte Carlo and Simulated Annealing

Simulated annealing uses the Metropolis MC algorithm with slowly decreasing temperature in order that system relaxes into its ″ground state″.



Cooling Schedule in SA



Performance of SA as a global optimization algorithm involves choice of hyperparameters to define a cooling schedule

- initial temperature T_{high}
- # iterations N_{temp} at each T_{SA}
- how, and how much, is T_{SA} decremented (incremented) during cooling (heating)

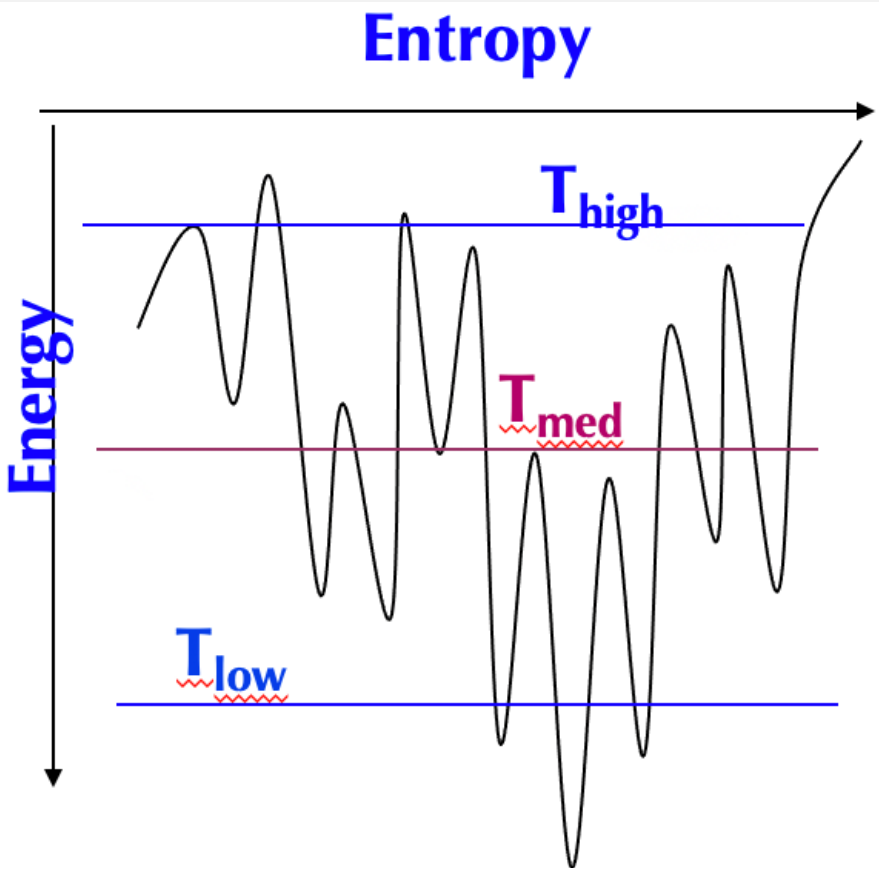
Linear cooling: $T_{SA}=T_{SA}-\alpha T$

- Typically spend as much time at T_{high} as at T_{low}
- can still get trapped in local minimum at T_{low} where need to spend more time annealing

Geometric cooling: $T_{SA}=\alpha\, T_{SA}$

- Spends more time at T_{low}
- Can be effective for some applications

Cooling Schedule in SA



In the special case that the cooling schedule has parametric form $T_{SA} \propto c/\log(I)$, where I is your iteration over N_{temp}, the condition for convergence is that c be greater than the depth of the deepest local minimum which is not a global minimum state.

Logarithmic cooling:

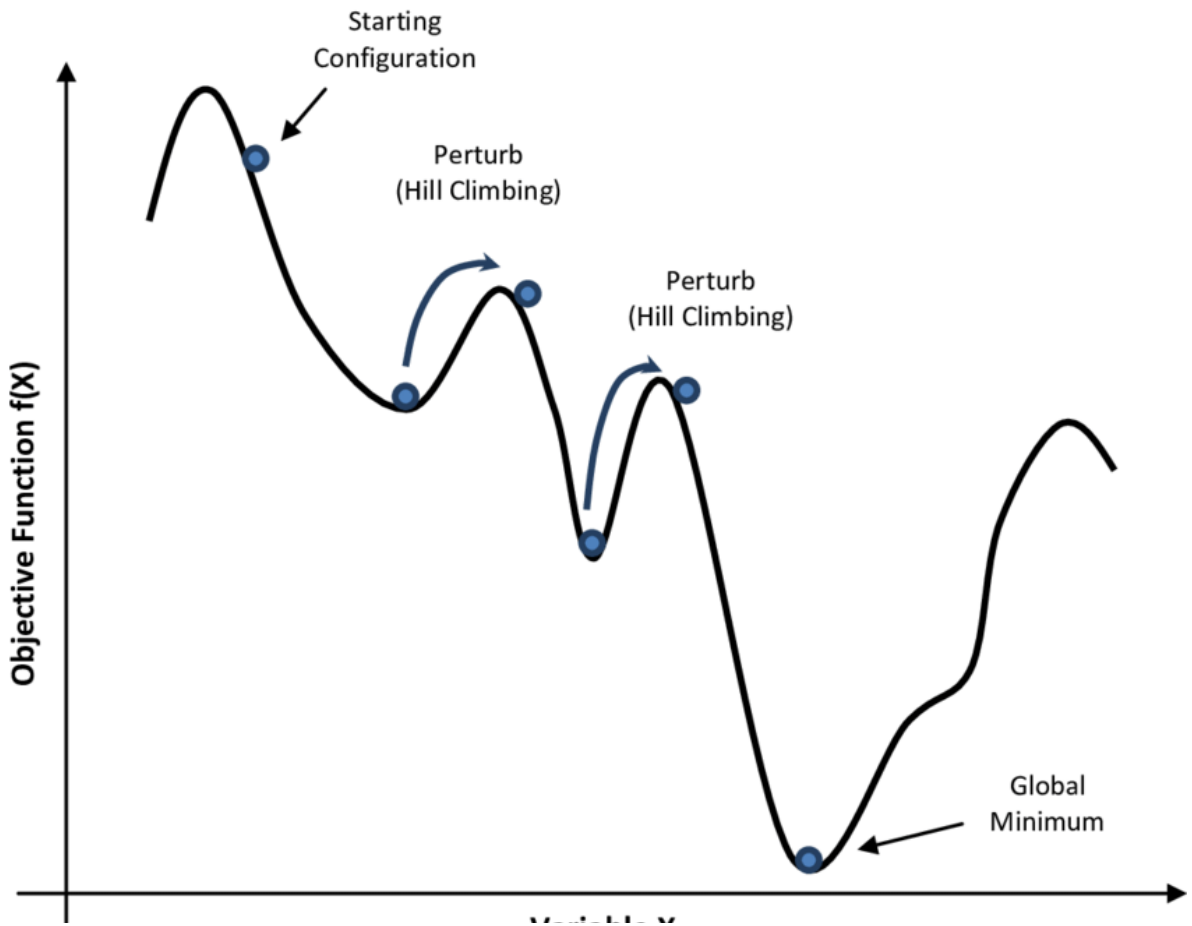
$$T_{SA} = \frac{T_{SA}}{1 + \frac{T_{SA}\log(1+\delta)}{3\sigma_{curr}}}$$

δ

σ_{curr}

New hyperparameters

Better optimization via system- dependent parameters, but need expert knowledge of your system!



Simulated Annealing: Pros and Cons

PROS:

- Easy to implement
- Certain cooling schedules have been proven to asymptotically reach a global minimum with P→1
- Has been successfully applied to chip circuit design, TSP, etc

CONS:

- Picking T_{high} is a black art: when too high SA amounts to a completely random search (at least until cooled to effective T_{SA})
- The cooling schedules that asymptotically reach a global minimum with P→1 do so in non-polynomial time!
- later stages of SA are largely gradient searches