CS 420-02: Undergraduate Simulation, Modeling and Analysis

Rahul Simha

Department of Computer Science
College of William & Mary
Williamsburg, VA

Chapter 1

Simulated Annealing

CS 420-02: Undergraduate Simulation, Modeling and Analysis

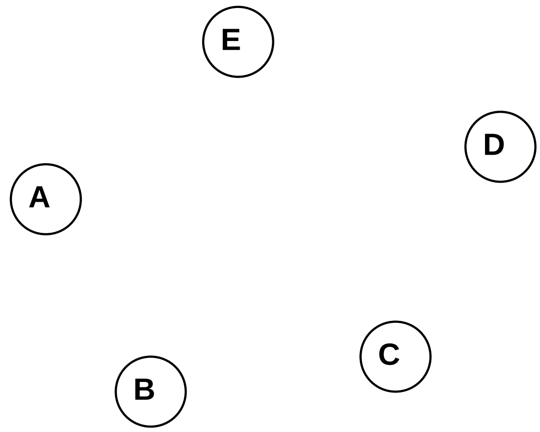
1.1 Introduction

- Multi-part lecture:
  1. Markov chains.
  2. Statistical physics and the Boltzmann distribution.
  3. Annealing in metallurgy.
  4. Combinatorial problems and local search.
  5. Simulated annealing.

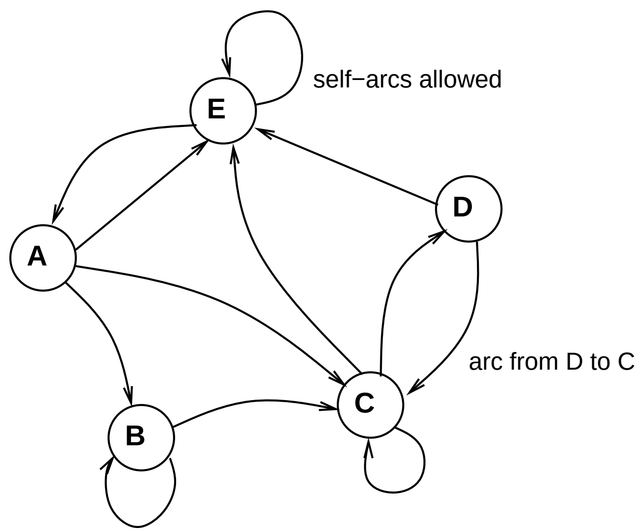
1.2 Markov Chains

- Markov chains via an example: consider the following process:

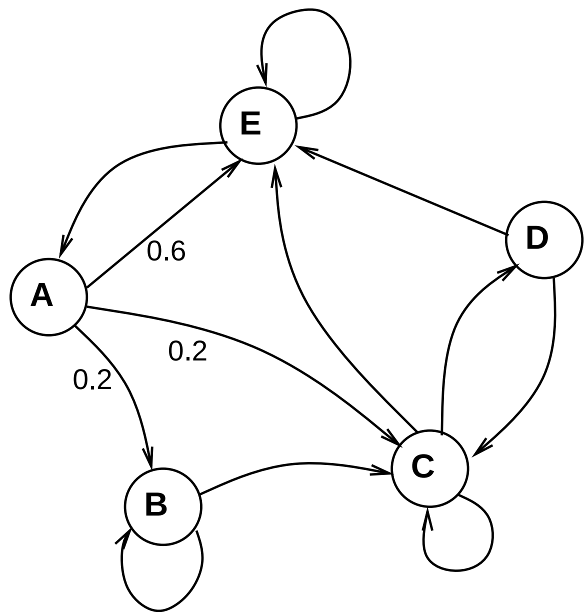
  1. Draw a bunch of "states" (e.g., 5 states):



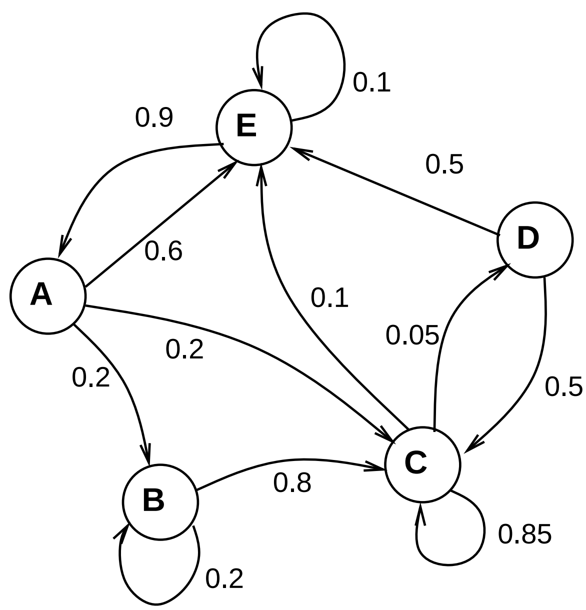
  2. Draw directed arcs between some of the states:



  3. For each state, use a probability distribution over the outgoing arcs:



4. Do this for all states:



5. Pick a start state, e.g. `start = A`.

6. Execute this algorithm:

```
i := 1;
s := start;
repeat
    jump to neighbor of s using arc probabilities of s;
    i := i + 1;
until i > n;
```

Note: jump probabilities are independent of past history

- Questions of interest:
  - Suppose X_n = state you are in after n-th jump.
  - Q: what is $P[X_n = A]$?
  - If I start in A, after how long do I get back to A? (first passage time to A).
- Markov chain theory:

  If these conditions hold:

  1. All states are reachable;
  2. set of states is finite;

  then

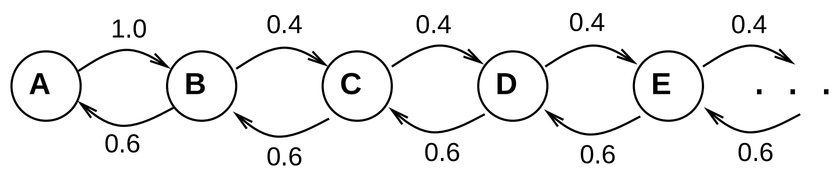
  $$\lim_{n \to \infty} P[X_n = A]$$

  exists and is easy to compute.

  $\lim_{n \to \infty} P[X_n = A]$ = long term probability of being in A.

  Note: limit theorems hold under other conditions as well.
- Simulation:
  - For above example, which state is likely to have the least probability?

- Why Markov chains are useful:
  - Many systems can be modeled as a process evolving on a state space
  - If the "Markov" property holds, these systems can be analyzed quite easily.
  - Many powerful results exist in the theory of Markov chains.
- Why Markov chains are called Markov chains:
  - A.Markov: Russian mathematician who first worked out the mathematics of Markov chains.
  - His examples usually looked like chains:



- Summary:
  - A Markov chain is a process that jumps around from state to state, in a collection of states.
  - The long term probability of being in a state can be computed.
  - First passage time is the average time to return to a start state (hard to compute).

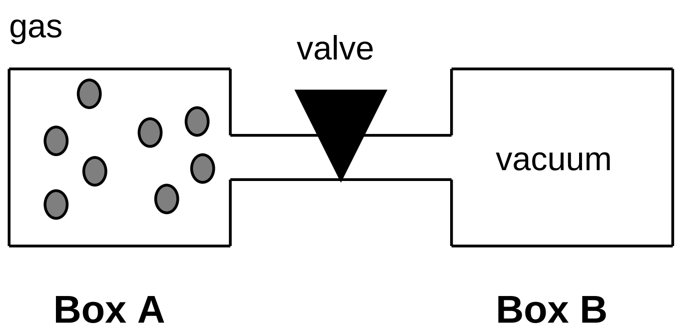
1.3 The Boltzmann Distribution

- Ludwig Boltzmann (Austria, 1844-1906):
  - Prior to Boltzmann, macroscopic laws of gases were discovered and empirically verified, e.g.,

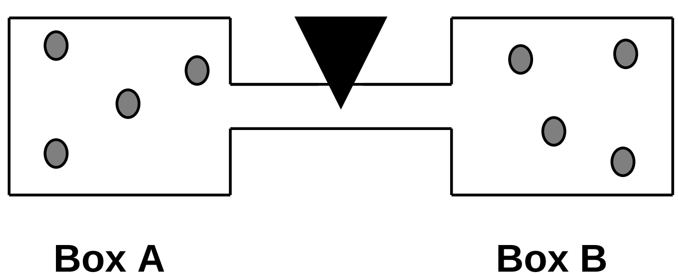
    $$\frac{PV}{T} = \text{constant} \qquad \text{(Boyle's law)}$$

  - Boltzmann was interested in explaining macroscopic properties using microscopic properties.
- Example of a problem Boltzmann was interested in:

  Suppose all molecules are initially in Box A:



Then, the valve is opened and after a while the system is examined:



The molecules appear to be evenly distributed (identitical pressure).

The system is continuously observed for a long time, yet the initial configuration is never observed again - why?

- Markov chain analogy:

  Let

$$
\begin{aligned}
n &= \text{total \# molecules} \\
n_A &= \text{\# molecules in A} \\
n_B &= \text{\# molecules in B}
\end{aligned}
$$

  State of the system $= (n_A, n_B)$.

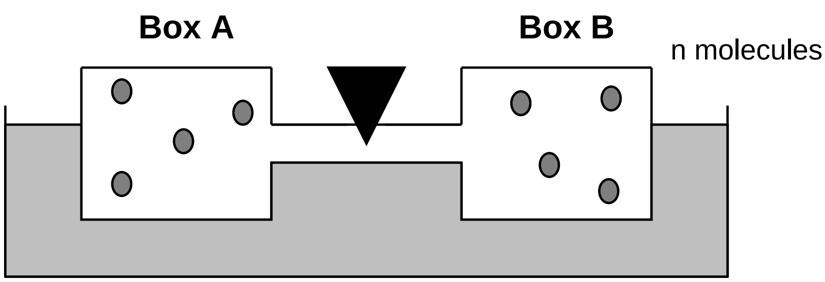
  Initial state $= (n, 0)$.

  Observation: first passage time from $(n, 0)$ to $(n, 0)$ is *very* long. (average time is longer than the life of the universe, for a large system).

- A simple model:
  - Suppose at every step, each molecule selects a Box at random (with equal probability).
  - Then, $P[\text{all molecules in A}] = 0.5^n$.
  - In fact, $P[k \text{ molecules in A}] = \begin{pmatrix} n \\ k \end{pmatrix} 0.5^n 0.5^{n-k}$.
  - Most probable state: $(\frac{n}{2}, \frac{n}{2})$.
  - E.g., n=20:
    * $P[\frac{n}{2} \text{ molecules in } A] = P[10 \text{ in } A] \approx 0.176$.
    * $P[n \text{ molecules in } A] = P[20 \text{ in } A] \approx 10^{-6}$.

- Boltzmann's analysis: key assumptions
  - We cannot account for the behavior of each individual molecule.
  - All configurations with the same energy are equally probable.

- Boltzmann's analysis:
  - System:



– Notation:
  * Each configuration of molecules is a *state*.
  * S = set of states = $\{s_1, s_2, \ldots, s_m\}$.
  * $E(s)$ = energy of state s.
  * $E_1, E_2, \ldots, E_k$ = possible energies.
– Desired: what is P[a state has energy E_i]?
– Analysis:
  Note that

  $P[\text{energy is } E_A + E_B] = P[\text{energy in A is } E_A] \times P[\text{energy in B is } E_B]$.

  Thus, the probability distribution has the form

$$f(x+y) = f(x)f(y).$$

  Note that

$$e^{-\beta(x+y)} = e^{-\beta x}e^{-\beta y}$$

  and thus $f(x) = e^{-\beta x}$ is a candidate function.
  Fact: f is necessarily of the form $f(x) = e^{-\beta x}$.
  Thus,

$$P[\text{a state has energy } E] = (\text{const})e^{-\beta E}.$$

  Recall: we have a finite number of energies. Hence,

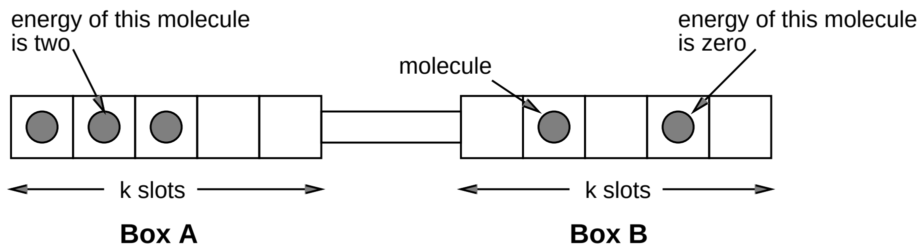
$$P[\text{a state has energy } E_i] = Ze^{-\beta E_i}.$$

  where

$$Z = \frac{1}{\sum_k e^{-\beta E_k}}.$$

  This is called the Boltzmann distribution.

- The probability of finding the system in energy E.
  - Let $P[E] = P[\text{a state has energy } E] = Ze^{-\beta E}$.
  - Note: $P[E]$ is a decreasing function of E.
  - Let $\Omega(E) = \#$ states with energy E.
  - Note: $\Omega(E)$ is an increasing function of E.
  - Let $P_{sys}[E] = P[\text{system has energy } E]$.
    Then,
    $$P_{sys}[E] = \Omega(E)P[E].$$
    Example: a plot of $\Omega(E)$, $P[E]$ and $P_{sys}[E]$
  - Q: why does $\Omega(E)$ increase?
- A simple simulation experiment:
  - System (1-dimensional example):



- n molecules.
- Each molecule selects a slot randomly in either Box.
- The energy of a molecule = # neighbors.
- Energy of a configuration = sum of energies of molecules.

• The effect of temperature:

- Recall Boltzmann distribution: $P[E] = Ze^{-\beta E}$.
- By computing macro properties (e.g., pressure), it turns out:

$$\beta \propto \frac{1}{T}.$$

  This is usually written as

$$\beta = \frac{1}{\kappa T}$$

  where κ is Boltzmann's constant. Thus,

$$P[E] = Ze^{-E/\kappa T}.$$

- Next, consider two states s_1 and s_2 with energies $E(s_2) > E(s_1)$. Then,

$$r = \frac{P[E(s_1)]}{P[E(s_2)]} = \frac{Ze^{-E(s_1)/\kappa T}}{Ze^{-E(s_2)/\kappa T}} = e^{[E(s_2)-E(s_1)]/\kappa T}.$$

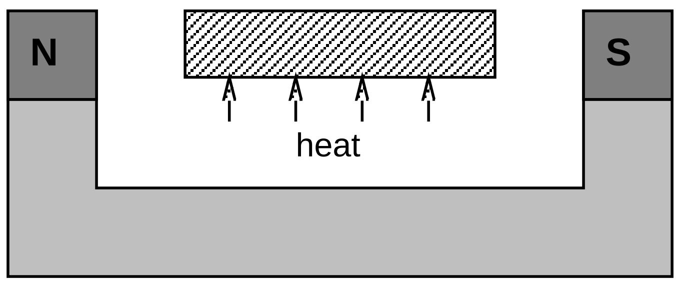
- Q: What happens to r as $T \to \infty$?
- Q: What happens to r as $T \to 0$?
- Thus, low energy states are more probable at low temperatures.

- Simulation example:

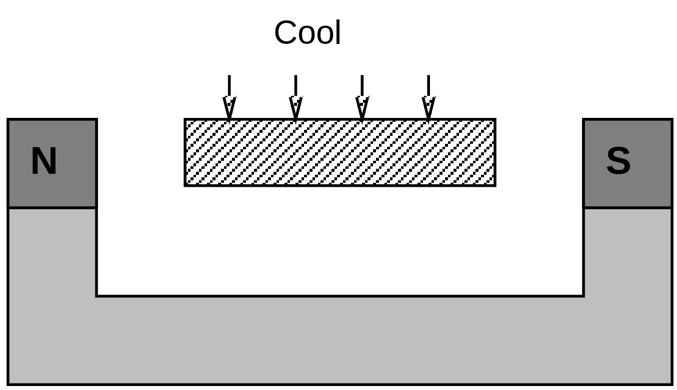
- Summary:
  - P[a state has energy E] $\propto e^{-E/\kappa T}$.
  - Low energy states are favored at low temperatures.

1.4 Annealing

- *Annealing* is a process discovered centuries ago as a technique for improving the strength of metals.
- Key idea: cool metal slowly during the forging process.
- Example: making bar magnets
  - Wrong way to make a magnet:
    1. Heat metal bar to high temperature in a magnetic field:

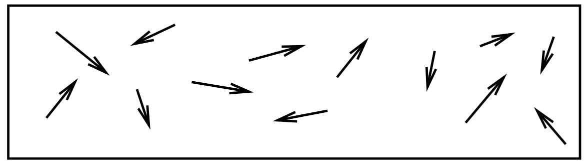


    2. Cool rapidly (quench):

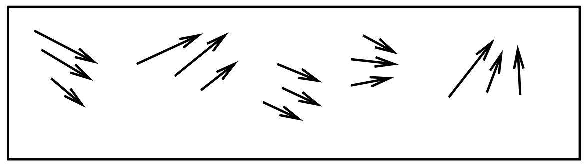


  - Right way: cool slowly.

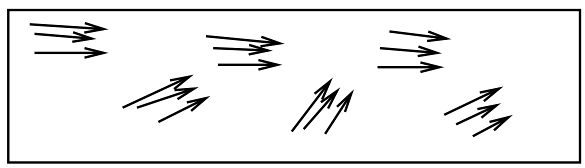
- Why slow-cooling works:
  - At high heat, magnetic dipoles are agitated and move around:



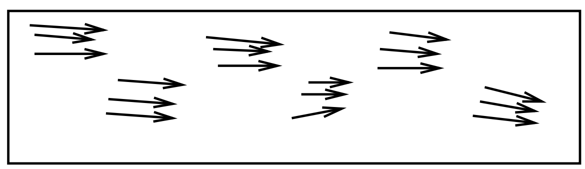
  - The magnetic field tries to force alignment:



  - If cooled rapidly, alignments tend to be less than optimal (local alignments):



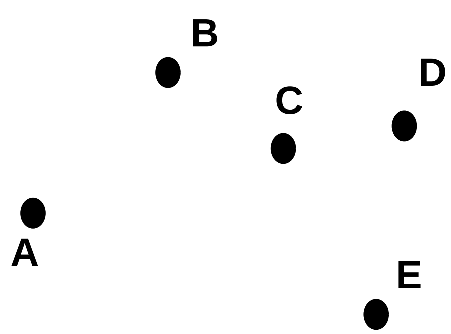
  - With slow cooling, alignments are closer to optimal (global alignment):



- Summary: slow cooling helps because it gives molecules more time to "settle" into an optimal configuration.

1.5 Combinatorial Optimization Problems

- A *combinatorial optimization problem* is:
  - S = set of states (potential solutions).
  - C, a cost function over the states:
    $C(s)$ = cost of state s.
  - Goal: find state with least cost.
  - Usually S is too large for exhaustive search.
- Example: the Traveling Salesman problem
  - Informal description:
    We are given a bunch of cities:



and the distance between each pair of cities (matrix D):

	A	B	C	D	E
A	0	2.7	3.1	3.6	2.9
B		0	1.8	2.1	3.4
C			0	0.8	1.2
D				0	1.1
E					0

We wish to find a tour through the cities (each city occurs only once in a tour) of minimal total length.

– Why is this a combinatorial optimization problem?

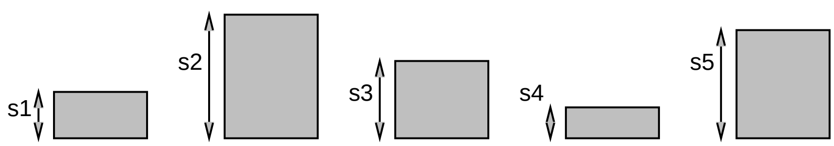
  * Does it have a set of states?

$$
\begin{aligned}
S &= \{\text{ all possible tours}\} \\
&= \{ABCDE, ABCED, ABECD, \ldots, EDCAB\} \qquad \surd
\end{aligned}
$$

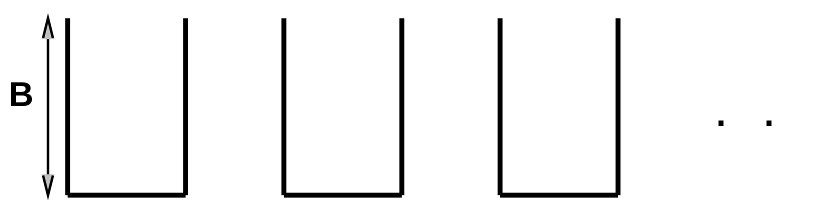
  * Does it have a cost function on the states?
    $C(ABCDE) = D(A,B) + D(B,C) + D(C,D) + D(D,E)$. $\surd$
  * Is the goal to find the minimal cost state?
    Goal: find an ordering of cities $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$ such that $C(\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5)$ is minimal. $\surd$

- Example: the Bin Packing problem

  – Informal description:
  Given a collection of items of sizes $s_1, \ldots, s_n$



  and an unlimited supply of bins each of size B:



  pack the items into as few bins as possible.

  – Formal description:

  * Item sizes: $s_1, s_2, \ldots, s_n$.
  * Assignment function:

$$
\delta_{ij} = \begin{cases} 1, & \text{if item } i \text{ is put into bin } j \\ 0, & \text{otherwise} \end{cases}
$$

  * B = bin size.

* Goal: minimize k, the number of bins such that

$$\sum_{i=1}^{n} s_i \delta_{ij} \leq B \tag{1}$$

$$\sum_{i=1}^{k} \delta_{ij} = 1 \tag{2}$$

$$\tag{3}$$

- Why is this a combinatorial optimization problem?
  * Set of states: all possible assigments of 0-1 values to the matrix δ.
  * Cost function: number of bins used.

- Example: the Satisfiability problem
  - U is a collection of Boolean variables $\{x_1, x_2, \ldots, x_n\}$.
  - O is a collection of Boolean operators: $\wedge$ (**and**), $\vee$ (**or**) and $'$ (**not**).
  - B is a Boolean expression using variables in U and operators in O, e.g.,

$$B = (x_1 \vee x_2) \wedge (x_1' \vee x_3 \wedge x_2)$$

  - Is there an assignment of T and F values to the x_i's such that B is true?

- Summary: a combinatorial optimization problem is:
  - S = set of states = $\{s_1, s_2, \ldots, s_m\}$.
  - A cost function $C : S \rightarrow R$
    $C(s_i)$ = cost of state s_i.
  - Goal: find least-cost state.

- Note:
  - Let $S^* = \{s : C(s) \leq C(s') \text{ for every } s' \in S\}$.
  - Need to find any element in S^*.

  - Usually size of problem is n (number of cities).
  - Size of state space is large (all possible tours).

- Fact: A large class of problems (NP-complete problems) are polynomially equivalent to each other.
  (If you can solve one efficiently, you can solve every one of them).

1.6 Local Search

- *Local search* is a general-purpose algorithm to solve any combinatorial optimization problem.
- Algorithm:

Algorithm: GREEDY-LOCAL-SEARCH

```
s := initial_state; // e.g., initial tour
repeat
    s' := GENERATE-NEW-STATE(s); // new tour
    if C(s') < C(s) // new tour has less cost
        s := s';
        changed := true;
    else
        changed := false;
    endif;
until not changed;
return s, C(s);
```

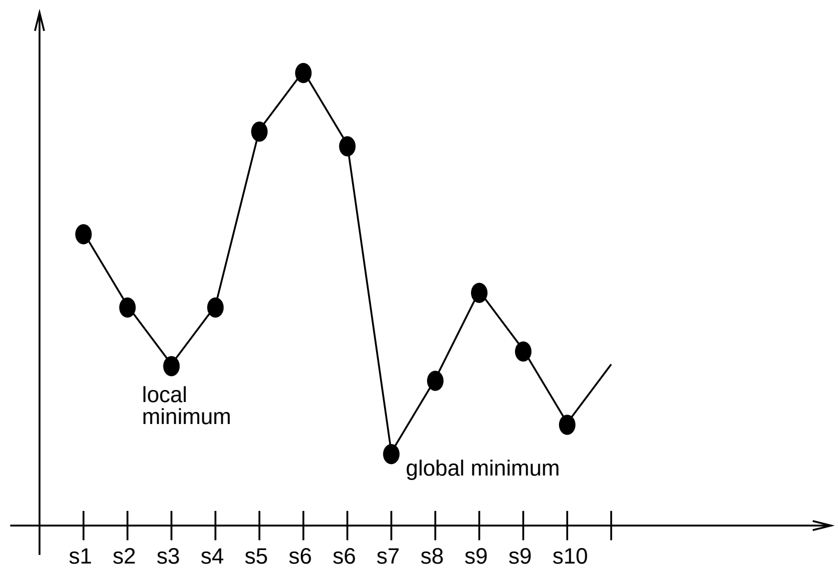
- How to generate new states?
  e.g., Traveling Salesman problem:
  - Suppose current tour is $s = \alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$.
  - Pick two cities at random, e.g.

$$\begin{array}{ccccc} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 \\ & \uparrow & & \uparrow & \end{array}$$

  - Swap the two cities: $s' = \alpha_1\alpha_4\alpha_3\alpha_2\alpha_5$.

- How well does GREEDY-LOCAL-SEARCH work?
  Ans: not very well on most problems.

  Why?
  Ans: The local structure of the cost *landscape* reveals little about the global structure.



- Observation: perhaps we should allow an algorithm to "climb" out of local minima?

1.7 Summary So Far

- Markov chains:
  - A process that jumps from state to state.
  - Long-term probabilities can be computed.
- Boltzmann distribution:
  - Consider a system that can be in one of many states, and where each state has an energy level.
  - Suppose energy values are: $E_1, E_2, \ldots, E_m$.
  - The Boltzmann distribution:

    $$P[\text{a state has energy } E_i] = Ze^{-\beta E_i}.$$

    where

    $$Z = \frac{1}{\sum_k e^{-\beta E_k}}.$$
  - Small $T \Rightarrow$ low-energy states have higher probability.
- Annealing:
  - Slow cooling (after heating) helps improve properties of materials.
- Combinatorial optimization problem:
  - Set of states and a cost function over the states.
  - Goal: find minimum cost state.
- Local search:
  - Start in any state.
  - Jump to a neighboring state if it's cheaper.
  - Stop when you can't go anywhere.

1.8 Simulated Annealing

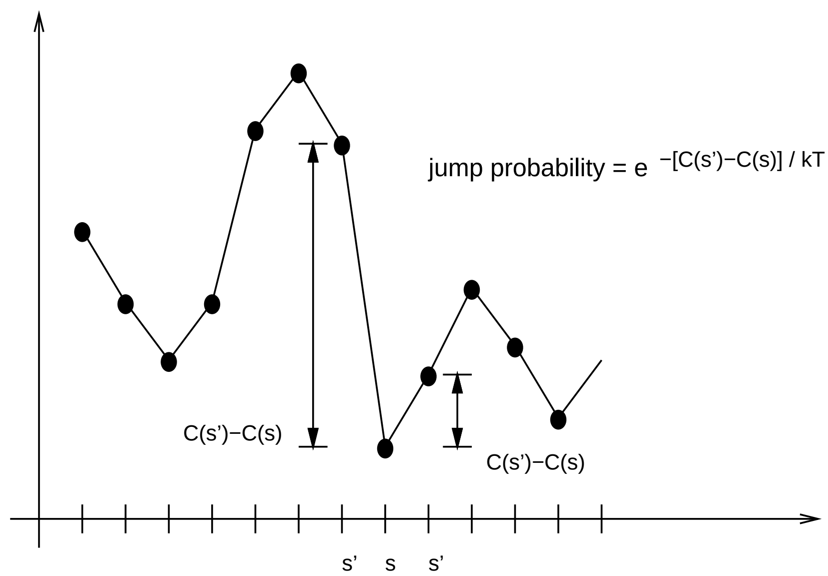
- Key ideas:
  - Simulated annealing = local search with modifications.
  - Allow jumps to higher cost states.
  - Use a coin flip to determine whether you should jump to a higher cost state
    (with probability $e^{-[C(s')-C(s)]/\kappa T}$.)
  - Decrease the probability as time goes on.
    (By decreasing the temperature).
  - The hope is:
    * Initially, higher-cost jumps occur with high probability
      $\Rightarrow$ allows exploration of state space.
    * Later, higher-cost jumps occur with low probability
      $\Rightarrow$ decrease the chances of jumping out of low cost states.

- Algorithm:

Algorithm: SIMULATED-ANNEALING

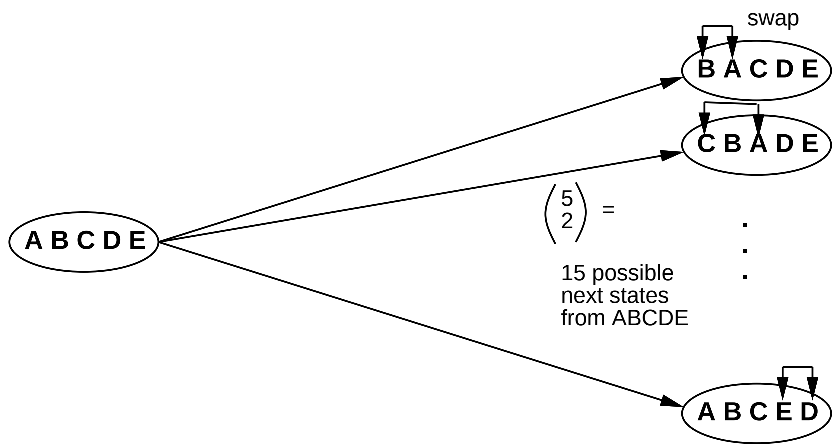
```
s := initial_state;
min_s := s;
T := initial_temperature;
repeat
   s' := GENERATE-NEW-STATE(s);
   if C(s') < C(s)
      s := s';
   else if uniform_random() < e^(-[C(s')-C(s)]/κT)
      s := s'; // even though C(s') > C(s)
   else
      stay in same state;
   endif;
   if C(s) < C(min_s)
      min_s := s;
   endif;
   T := NEW-TEMPERATURE(T);
until tired;
output min_s, C(min_s);
```

• Note: probability of jump depends on cost difference.

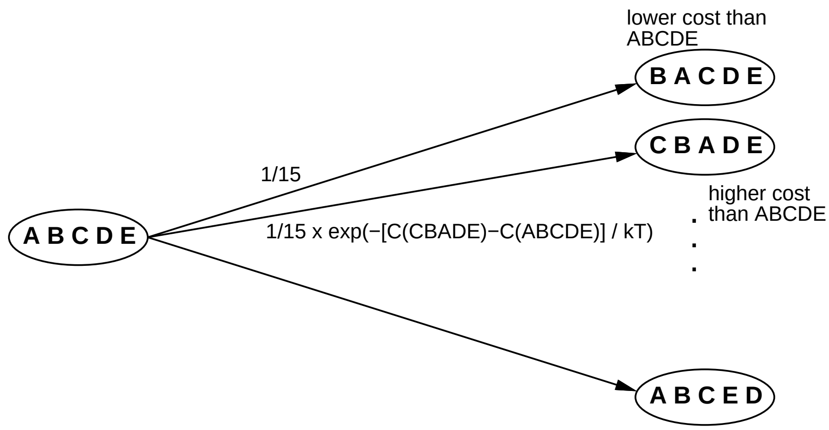


1.9 Mathematics of Simulated Annealing

- Example: Traveling Salesman over 5 cities.
- Consider the state $ABCDE$: where can we jump to from here?



What are the jump probabilities?



Next, let X_n = state after n-th jump.

Then, X_n is a Markov chain!

- Fixed-temperature mathematics:
  - Suppose T is constant throughout the execution of the algorithm.
  - It turns out the Markov chain can be solved easily to give:

$$\lim_{n \to \infty} P[X_n = s] \propto e^{-C(s)/\kappa T}$$

    – the state distribution is the Boltzmann distribution.
  - Consider states s_1 and s_2 such that $C(s_2) > C(s_1)$.
    For large n,

$$\begin{aligned} r &= \frac{P[X_n = s_1]}{P[X_n = s_2]} \\ &= \frac{e^{-C(s_1)/\kappa T}}{e^{-C(s_2)/\kappa T}} \\ &= e^{-[C(s_1) - C(s_2)]/\kappa T} \end{aligned}$$

    Note:
    * For large T, $r \approx 1$.
    * For small T, $r \approx \infty$.
  - Theoretical result:

$$\lim_{T \to 0} \lim_{n \to \infty} P[X_n \in S^*] = 1.$$

    (Recall: S^* = set of optimal states.)

- Decreasing-temperature mathematics:
  - As $n \to \infty$, $T \to 0$.
  - The process is still a Markov chain, but a non-standard Markov chain (non-homogeneous)
    $\Rightarrow$ difficult to analyze.
  - Theoretical result: If $T_n \to 0$ slowly (e.g., $T_n \geq \frac{\gamma}{\log_e n}$) then

$$\lim_{n\to\infty} P[X_n \in S^*] = 1.$$

  - Key idea in proof:
    Let $c = \max_{i,j}[C(s_i) - C(s_j)]$.
    Then,

$$\lim_{n\to\infty} P[X_n \in S^*] = 1$$

    if

$$P[\text{stuck in a well}] = 0$$

    which is true if

$$\sum_n e^{-c/\kappa T_n} = \infty \qquad \text{(Borel-Cantelli lemma)}$$

    which is true if

$$\sum_n e^{-(c/\kappa\gamma)\log n} = \infty$$

    which is true if

$$\sum_n \frac{1}{n} = \infty$$

    which is true.

1.10 Simulated Annealing: Summary

- A metaphor from the physics of metals was used to create an algorithm.
- Simulated Annealing is a general purpose algorithm to solve combinatorial optimization problems.
- To solve a particular problem, you need to define a Generate-New-State(s) fucntion for that problem.
- The initial temperature will have to be selected depending on the particular instance of the problem.
- The mathematics of simulated annealing involve Markov chains (a construct in probability theory).
- In practice:
  - Simulated annealing is easy to implement.
  - Simulated annealing has been found to work well for approximately bowl-like landscapes.
  - Performance is strongly dependent on good neighborhood functions.
  - Performance can be enhanced if supplemented with other strategies (e.g., use multiple starting points).
  - The theoretical temperature schedule is too slow.
  - Newer algorithms (e.g., TABU search) build on and are better than simulated annealing.