Optimization. Simulated annealing

Maciej Komosinski

Institute of Computing Science
Poznan University of Technology
www.cs.put.poznan.pl/mkomosinski

Annealing

- increasing the temperature of a hot bath to such a value that the solid melts
- *slowly* decreasing the temperature until the molecules line up and reach zero temperature (ground state)
- the opposite of hardening

The Metropolis algorithm

- Metropolis et al. (1953) – an algorithm of statistical *simulation* (Monte Carlo) of changes of a solid in a hot bath until the state of thermal equilibrium

- random generation of a sequence of states of a solid:
  - a state i of a solid and its energy E_i,
  - perturbation (small change) $\rightarrow$ next state.
    The energy of the next state is E_j.
  - if $E_j - E_i \leq 0$, the state j is accepted as the current state

The Metropolis algorithm

- Metropolis et al. (1953) – an algorithm of statistical *simulation* (Monte Carlo) of changes of a solid in a hot bath until the state of thermal equilibrium

- random generation of a sequence of states of a solid:
  - a state i of a solid and its energy E_i,
  - perturbation (small change) $\rightarrow$ next state.
    The energy of the next state is E_j.
  - if $E_j - E_i \leq 0$, the state j is accepted as the current state
  - otherwise, state j is accepted with some probability:

$$\exp\left(\frac{E_i - E_j}{k_B T}\right)$$

T – bath temperature
k_B – Boltzmann constant

The Metropolis algorithm

- Metropolis et al. (1953) – an algorithm of statistical *simulation* (Monte Carlo) of changes of a solid in a hot bath until the state of thermal equilibrium

- random generation of a sequence of states of a solid:
  - a state i of a solid and its energy E_i,
  - perturbation (small change) $\rightarrow$ next state.
    The energy of the next state is E_j.
  - if $E_j - E_i \leq 0$, the state j is accepted as the current state
  - otherwise, state j is accepted with some probability:

$$\exp\left(\frac{E_i - E_j}{k_B T}\right)$$

T – bath temperature
k_B – Boltzmann constant

- https://www.youtube.com/watch?v=h1NOS_wxgGg, https://www.youtube.com/watch?v=vTUwEu53uzs

Analogies to optimization

Physical system	Optimization problem
state	solution
energy	cost
ground state	optimum
temperature T	parameter c
fast cooling	local optimization
slow cooling	simulating annealing

Simulated annealing

- using the **Metropolis** algorithm for combinatorial optimization
- other names: *Monte Carlo annealing*, *probabilistic hill climbing*, *stochastic relaxation*

Acceptance criterion

- i, j – solutions
- $f(i)$, $f(j)$ – costs
- the acceptance criterion determines whether j obtained from i is accepted

$$\mathrm{P}_c\{\text{accept } j\} = \begin{cases} 1 & \text{if} \quad f(j) \leq f(i) \\ \exp\left(\frac{f(i)-f(j)}{c}\right) & \text{if} \quad f(j) > f(i) \end{cases}$$

Acceptance criterion

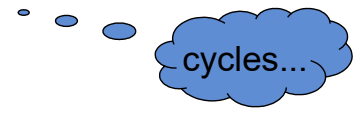
- i, j – solutions
- $f(i)$, $f(j)$ – costs
- the acceptance criterion determines whether j obtained from i is accepted

$$\mathrm{P}_c\{\text{accept } j\} = \begin{cases} 1 & \text{if} \quad f(j) \leq f(i) \\ \exp\left(\frac{f(i)-f(j)}{c}\right) & \text{if} \quad f(j) > f(i) \end{cases}$$

Homework: plot $e^{\frac{-\Delta}{c}}$ for several different c.

```
procedure SIMULATED_ANNEALING
begin
        INITIALIZE(x_start, C_0, L)
        k:=0
        x:=x_start
        repeat
                for l := 1 to L_k do
                begin
                        GENERATE(x' z N(x))
                        if f(x') ≤ f(x) then
                          x := x'
                        else
                          if exp(−(f(x') − f(x))/C_k) > random[0, 1) then
                            x := x'
                end
                k := k + 1
                CALCULATE(C_k)
        until STOPPING_CONDITION
end
```

```
procedure SIMULATED_ANNEALING
begin
        INITIALIZE(x_start, C_0, L)
        k:=0
        x:=x_start
        repeat
                for l := 1 to L_k do
                begin
                        GENERATE(x' z N(x))
                        if f(x') ≤ f(x) then
                           x := x'
                        else
                           if exp(−(f(x') − f(x))/C_k) > random[0, 1) then
                              x := x'
                end
                k := k + 1
                CALCULATE(C_k)
        until STOPPING_CONDITION
end
```



The convergence of the SA algorithm

- it is possible to determine such L_k and c_k that ensure the convergence of SA to the optimum

The convergence of the SA algorithm

- it is possible to determine such L_k and c_k that ensure the convergence of SA to the optimum
- a good approximation of SA: generating homogeneous Markov chains of finite length for a finite sequence of decreasing values of the parameter c

A Markov chain

- a Markov chain is a sequence of trials (events, solutions) in which the probability of the outcome of a given trial depends (only) on the outcome of the previous trial

A Markov chain

- a Markov chain is a sequence of trials (events, solutions) in which the probability of the outcome of a given trial depends (only) on the outcome of the previous trial
- a Markov chain is non-homogeneous if the transition probability depends on the trial number, k. If it does not depend on k, then the Markov chain is (time-)homogeneous.

The cooling process (schedule) is determined by

- a finite sequence of values of the parameter c, i.e.,
  - initial parameter value, c_0
  - parameter decrement function
  - final parameter value

The cooling process (schedule) is determined by

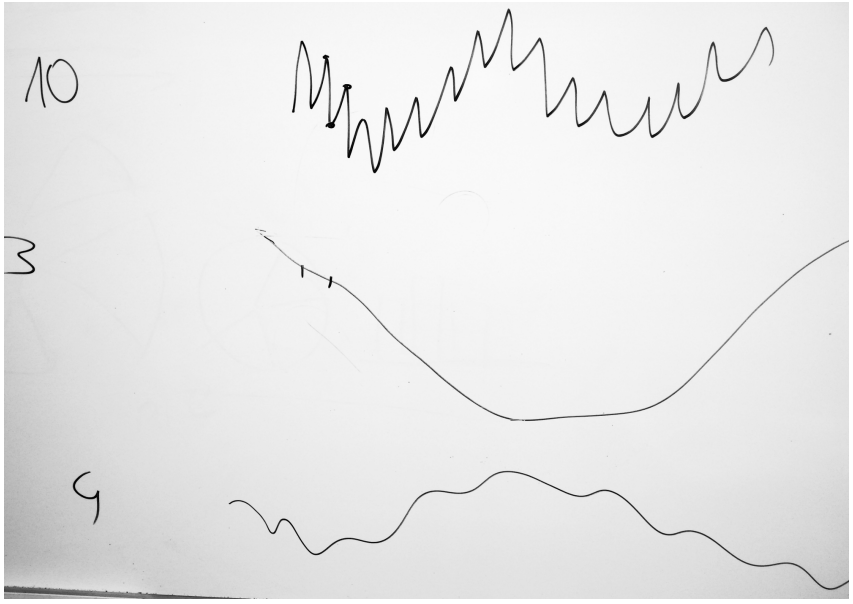
- a finite sequence of values of the parameter c, i.e.,
  - initial parameter value, c_0
  - parameter decrement function
  - final parameter value
- a finite number of moves for each value of the parameter c, i.e.,
  - a finite length of each homogeneous Markov chain

Initial value of the parameter c

- the c_0 value should be high enough to ensure that all moves are accepted (the initial acceptance rate is close to 1).

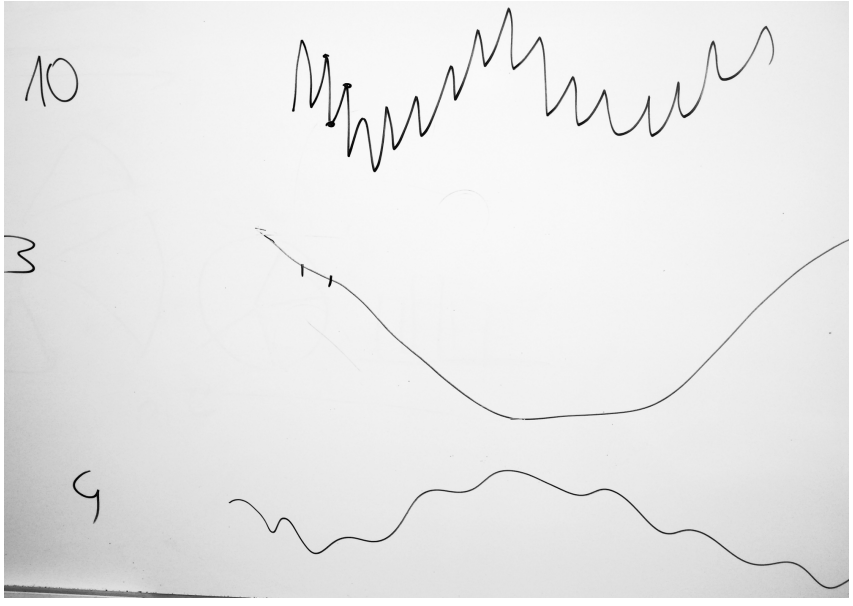
Initial value of the parameter c

- the c_0 value should be high enough to ensure that all moves are accepted (the initial acceptance rate is close to 1).

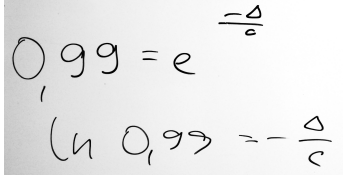


Initial value of the parameter c

- the c_0 value should be high enough to ensure that all moves are accepted (the initial acceptance rate is close to 1).



- depends on the problem (see the formula with Δf and P_c)



- for example: $p \approx 0.98$ and average $\Delta f = 1000 \Rightarrow c_0 \approx 49\ 500$
- or we can simulate heating...

Cooling schedule by Kirkpatrick, Gelatt and Vecchi

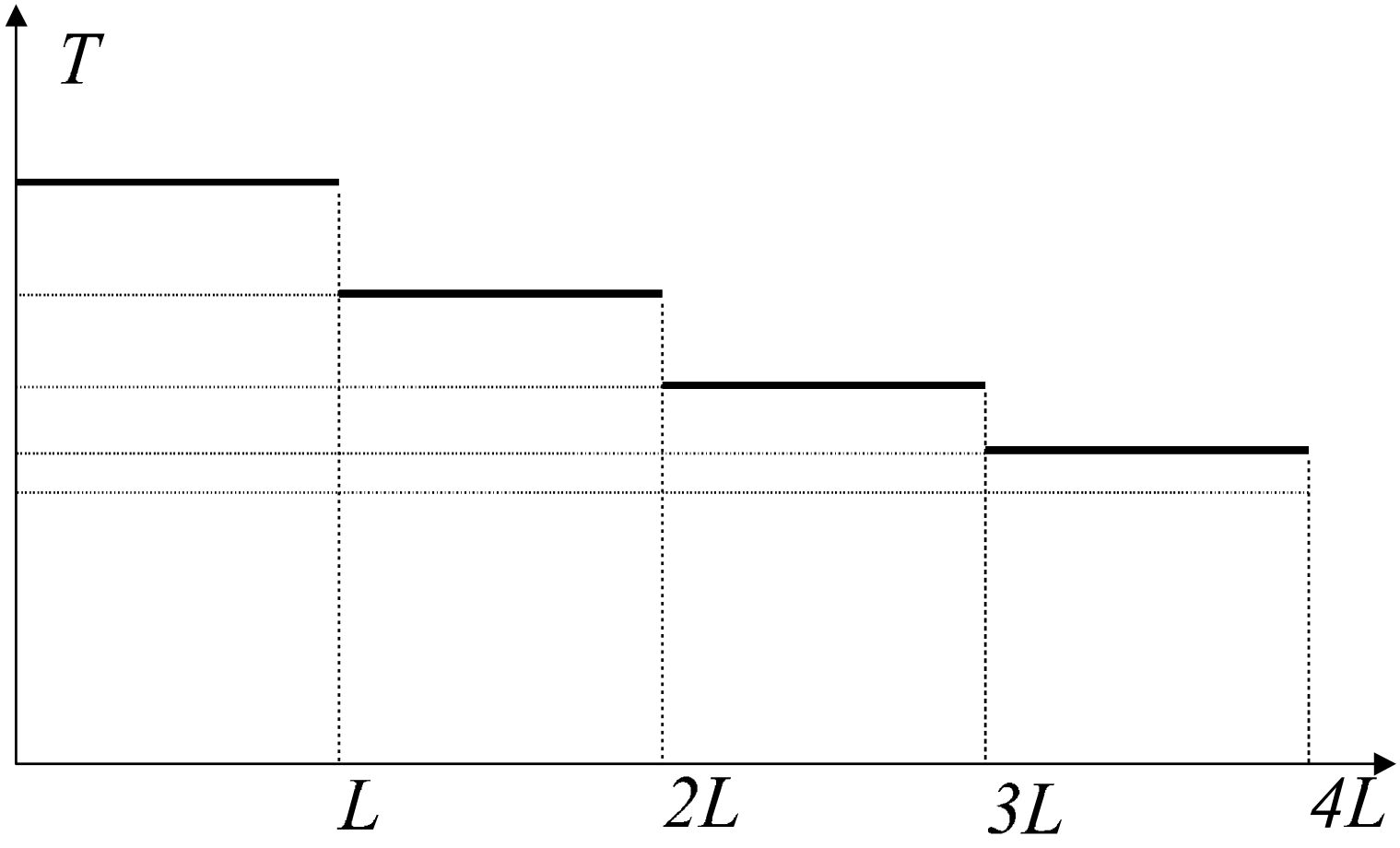
- Decreasing the value of the parameter c:

  $c_{k+1} = \alpha\ c_k, \qquad k = 1, 2, \ldots$

  $c_{k+1} = \alpha^k\ c_0$

  α is a constant smaller than 1 (for example $0.8 - 0.99$)

A simple cooling schedule: for example, $L_k = L$



Markov chain length

- should be sufficient for the algorithm to "visit" all neighbors at least once (i.e., to reach thermodynamic equilibrium at each temperature level)

Markov chain length

- should be sufficient for the algorithm to "visit" all neighbors at least once (i.e., to reach thermodynamic equilibrium at each temperature level)
- since the probability of acceptance decreases over time, one would expect $L_k \to \infty$ for $c_k \to 0$. Therefore, the length of the Markov chain must be limited for small c_k

Markov chain length

- should be sufficient for the algorithm to "visit" all neighbors at least once (i.e., to reach thermodynamic equilibrium at each temperature level)
- since the probability of acceptance decreases over time, one would expect $L_k \to \infty$ for $c_k \to 0$. Therefore, the length of the Markov chain must be limited for small c_k
- in practice – proportional to the average neighborhood size

The final value of the parameter c

The algorithm terminates when, for example, the current solution does not change for several consecutive Markov chains.

Other cooling schedules