U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence

Lecture 3b: Meta-Heuristics – Simulated Annealing

(based on Løkketangen, 2019)

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence

Simulated Annealing (Arrefecimento Simulado)

- A metaheuristic inspired by statistical thermodynamics
  - Based on an analogy with the cooling of material in a heat bath
- Used in optimization for over 30 years
- Very simple to implement
- A lot of literature
- Converges to the global optimum under weak assumptions (- usually slowly)
- Simulated annealing can be used for very hard computational optimization problems where exact algorithms fail
- Even though it usually achieves an approximate solution to the global optima, it could be enough for many practical problems

Simulated Annealing

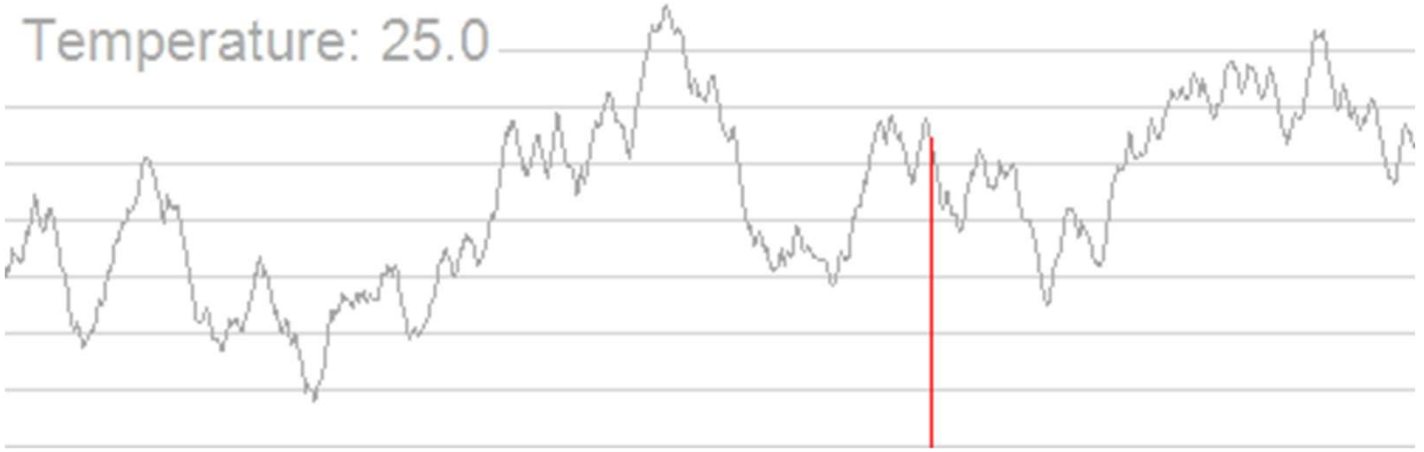
- **Name** of the algorithm comes from **annealing in metallurgy**, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects
- Both are attributes of the material that depend on their thermodynamic free energy
- Heating and cooling the material affects both the temperature and the thermodynamic free energy

Simulated Annealing

- **Metropolis' Algorithm (1953)**
  - Algorithm to simulate energy changes in physical systems when cooling
- **Kirkpatrick, Gelatt and Vecchi (1983)**
  - Suggested to use the same type of simulation to look for good solutions in a COP

Simulated Annealing

- The state of some physical systems, and the function E(s) to be minimized, is analogous to the internal energy of the system in that state.
- The goal is to bring the system, from an arbitrary initial state, to a state with the minimum possible energy.

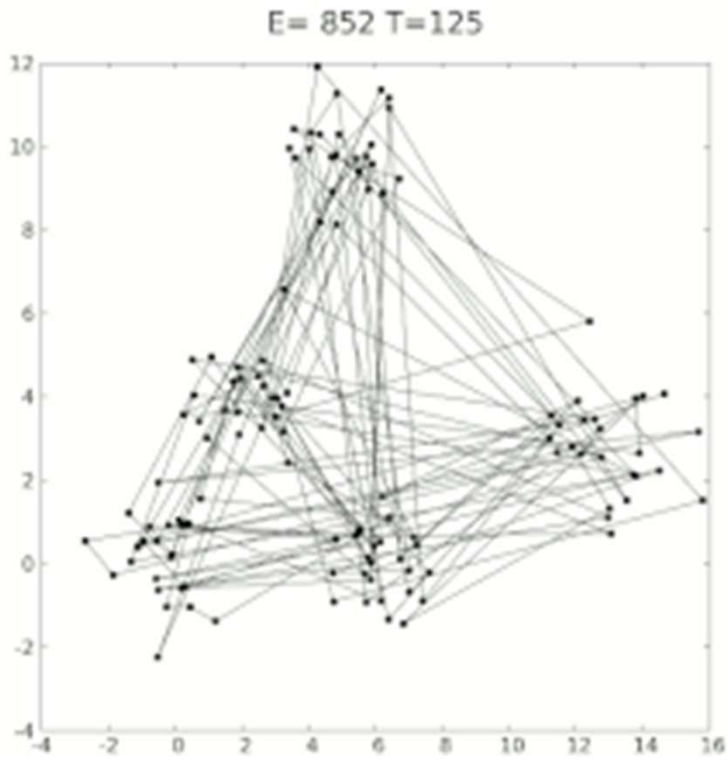


Simulated Annealing

Thermodynamics	Discrete optimization
1. Configuration of particles	1. Solution
2. System state	2. Feasible solution
3. Energy	3. Objective Function
4. State change	4. Move to neighboring solution
5. Temperature	5. Control Parameter
6. Final state	6. Final Solution

Simulated Annealing

- Can be interpreted as a modified random descent in the space of solutions
  - Choose a random neighbor
  - Improving moves are always accepted
  - Deteriorating moves are accepted with a probability that depends on:
    - the amount of the deterioration and on
    - the *temperature* (a parameter that decreases with time)
- Can escape local optima



TSP (125 points) Solved with SA

Based on Hill-Climbing (Subir a Colina)

- **Basic "Hill climbing":** Generates one by one the successors of the current state and if founds one better evaluated than the Current State, selects it and applies it
- **Basic "Hill climbing" (random):** Generates a random successor of the current state and if it is better evaluated than the Current State, selects it and applies it
- **"Steepest ascent":** Generates all possible successors and selects the best evaluated one

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

Simulated Annealing – Basic Algorithm

- **Basic Algorithm for Simulated Annealing:**
  - Start with an initial solution (random or other method)
  - For a given number of cycles or until achieving a given stopping criteria
  - Change the temperature, reducing it following the schedule

$s = s_0$; $T = T_0$

For $Iter = 0$ To $Iter_{max}$

  $T \leftarrow$ temp_sched(T)

  $s_{new} \leftarrow$ random_neighb(s)

  If $Prob(Eval(s), Eval(s_{new}), T) \geq$ rand(0, 1)

  Then $s \leftarrow s_{new}$

  - Calculate a possible new solution neighbour from the current one
  - If the probability of accepting the new solution given its performance change from the current one and the current temperature is greater or equal to a random value between o and 1, accept the new solution as the current solution

Choice of Move in Simulated Annealing

- **Modified ”Random Descent”**
- **Select a random solution in the neighbourhood**
- **Accept this**
  - Unconditionally if better than current
  - With a certain, finite probability if worse than current
- **The probability is controlled by a parameter called the *temperature***
- **Can escape from local optima**

Move Acceptance in Simulated Annealing

Assuming a maximization problem:

- Set: Δ = Eval(random neighbour) – Eval(current solution)
- If Δ > 0 → accept (we have an improving neighbour)
- Else accept it with probability $e^{\frac{\Delta}{t}}$
- If the move is not accepted: Try another random neighbor

```
function SIMULATED-ANNEALING( problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"
  local variables: current, a node
                   next, a node
                   T, a "temperature" controlling prob. of downward steps
  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
      T ← schedule[t]
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← VALUE[next] – VALUE[current]
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(Δ E/T)
```

Simulated Annealing - Structure

- **Initial temperature t_0 high**
  - (if $\infty \rightarrow$ random walk)
- **Reduce temperature *t* regularly**
  - Need a *cooling schedule*
  - If too fast $\rightarrow$ stop in some local optimum too early
  - If too slow $\rightarrow$ too slow convergence
- **Might restart**
- **Choice of neighborhood structure is very important**

Simulated Annealing

```
1: input: starting solution, s_0
2: input: neighborhood operator, N
3: input: evaluation function, f
4: input: the cooling schedule, t_k
5: input: the number of iterations for each temperature, M_k
6: current ⇐ s_0
7: k ⇐ 0
8: while stopping criterion not met do
9:    m ⇐ 0
10:   while m < M_k do
11:      s ⇐ randomly selected solution from N(current)
12:      if f(s) ≤ f(current) then
13:         current ⇐ s
14:      else
15:         Δ ⇐ f(s) − f(current)
16:         ξ ⇐ a random number, uniformly drawn from [0, 1]
17:         if ξ ≤ e^{−Δ/t_k} then
18:            current ⇐ s
19:         end if
20:      end if
21:      m ⇐ m + 1
22:   end while
23:   k ⇐ k + 1
24: end while
```

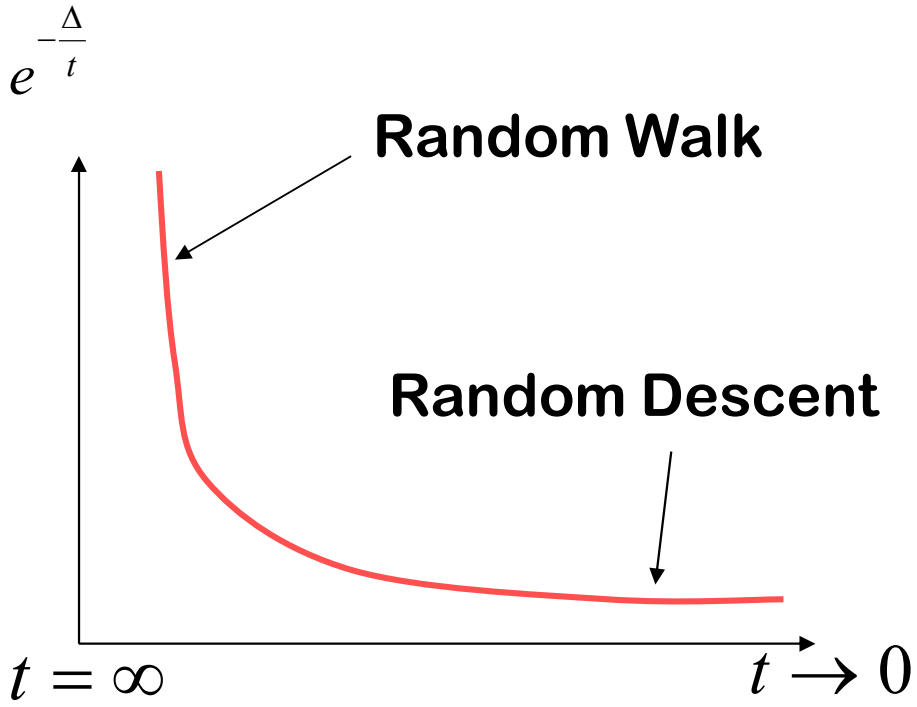
Simulated Anealling – Overall Structure

- Set the initial value of the control variable t (t_0) to a high value
- Do a certain number of iterations with the same temperature
- Then reduce the temperature $t_{i+1} = \alpha(t_i)$
- Need a ”cooling schedule”
- Stopping criterion – e.g. ”minimum temperature”
  - Repetition is possible
- Solution quality and speed are dependent on the choices made
- Choice of neighborhood structure is important

Simulated Annealing – Cooling Schedule

Cooling schedule is vitally important

- Much research on this
- Static schedules:
  - specified in advance
- Adaptive schedules:
  - react to information from the search
- Depending on the cooling schedule it may be good to perform some iterations at low (or zero) temperature to converge to local optima although not necessary if algorithm well applied



Statistical Analysis of SA

- Model: State transitions in the search space
- Transition probabilities [p_{ij}] (i,j are solutions)
- Only dependent on i and j: homogenous Markov chain
- If all the transition probabilities are finite, then the SA search will converge towards a stationary distribution, independent of the starting solution.
  - When the temperature approaches zero, this distribution will approach a uniform distribution over the global optima
- Statistical guarantee that SA finds a global optimum
- But: exponential (or infinite) search time to guarantee finding the optimum

Simulated Annealing in Practice

- **Heuristic algorithm with behaviour strongly dependent on the cooling schedule**
- **Theory:**
  - An exponential number of iterations at each temperature
- **Practice:**
  - A large number of iterations at each temperature, few temperatures
  - A small number of iterations at each temperature, many temperatures
- **Geometric chain:**
  - $t_{i+1} = \alpha(t_i)$, i = 0,…,Iter
  - α <1 (0.8 - 0.99)
- **Number of repetitions can be varied**
- **Adaptivity:**
  - Variable number of moves before the temperature reduction
- **Necessary to experiment**

General Decisions

- **Cooling Schedule**
  - Based on maximum difference in the objective function value of solutions, given a neighborhood
  - Number of repetitions at each temperature
  - Reduction rate, α
- **Adaptive number of repetitions**
  - More repetitions at lower temperatures
  - Number of accepted moves, but a maximum limit
- **Very low temperatures are not necessary**
- **Cooling rate most important**

Problem Specific Decisons

- **Important goals**
  - Time to get the solution
  - Quality of the solution
- **Important choices**
  - Search space
    - Infeasible solutions – should they be included?
  - Neighborhood structure
  - Move evaluation function
    - Use of penalty for violated constraints
    - Approximation – if expensive to evaluate
  - Cooling schedule

Choice of Neighbourhoods

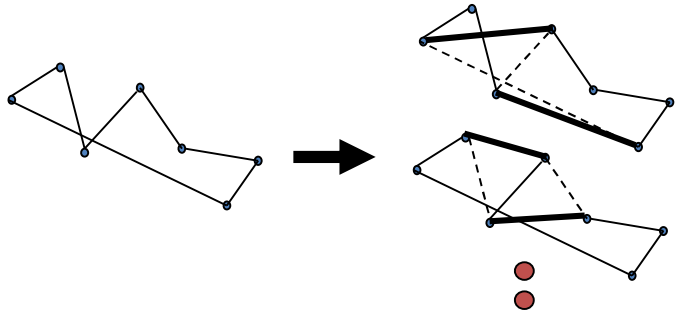
- **Size**
- **Variation in size**
- **Topologi**
  - Symmetry
  - Connectivity
    - Every solution can be reached from all the others
- **Topography**
  - Spikes, Plateaus, Deep local optima
- **Move evaluation function**
  - How expensive is it to calculate ?

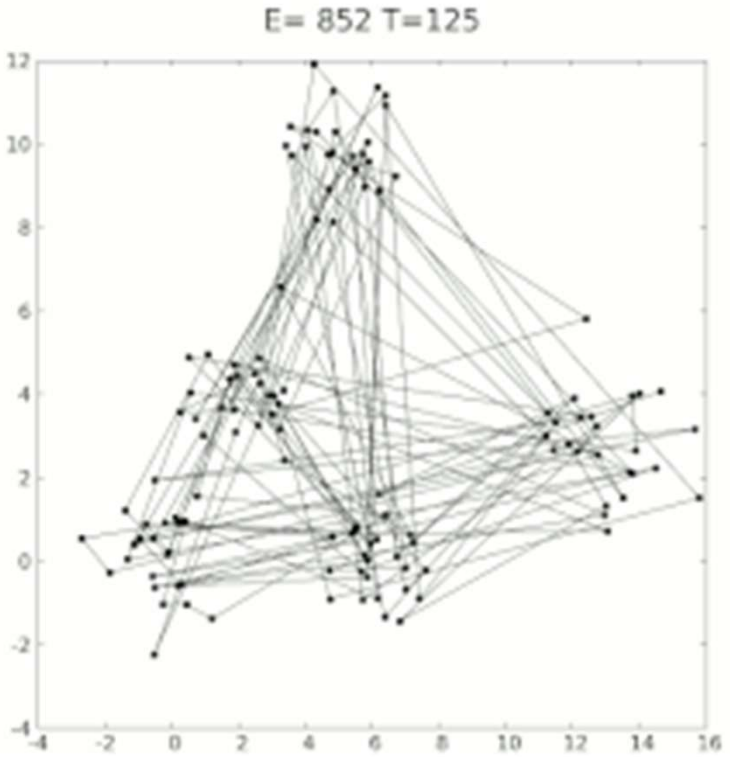
Simulated Annealing – Speed

- **Random choice of neighbour**
  - Reduction of the neighbourhood
  - Does not search through all the neighbours
- **Cost of new candidate solution**
  - Difference without full evaluation
  - Approximation (using surrogate functions)
- **Move acceptance criterion**
  - Simplify

Simulated Annealing – Example: TSP

- Search space: (n-1)!/2
- Neighborhood size:
  - 2-opt: n(n-1)/2
- Connected
- Simple representation of moves
- Natural cost function
- Difference in cost between solutions is easy to calculate
- Generalization: k-Opt





Simulated Annealing – Fine Tuning

- **Test problems**
- **Test bench**
- **Visualization of solutions**
- **Values for**
  - cost / penalties
  - temperature
  - number / proportion of accepted move
  - iterations / CPU time
- **Depencies between the SA-parameters**
- **The danger of overfitting**

Simulated Annealing – Summary

- **Inspired by statistical mechanics - cooling**
- **Metaheuristic**
  - Local search
  - Random descent
  - Use randomness to escape local optima
- **Simple and robust method**
  - Easy to get started
- **In practice:**
  - Computationally expensive
  - Needs well devised neighboorhoods and cooling schedules
  - Fine tuning can give good results
  - SA can be good where robust heuristics based on problem structure are difficult to make

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence

Lecture 3b: Meta-Heuristics – Simulated Annealing

(based on Løkketangen, 2019)

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence