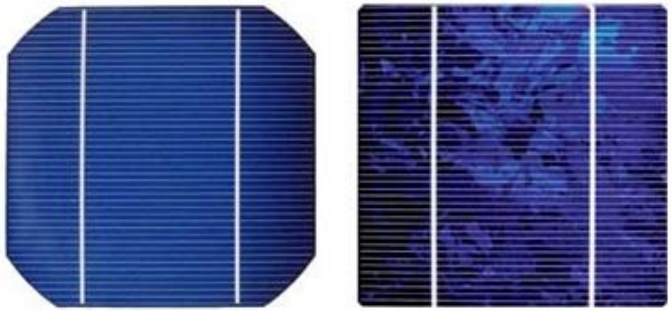
Iterative Improvement
Simulated Annealing

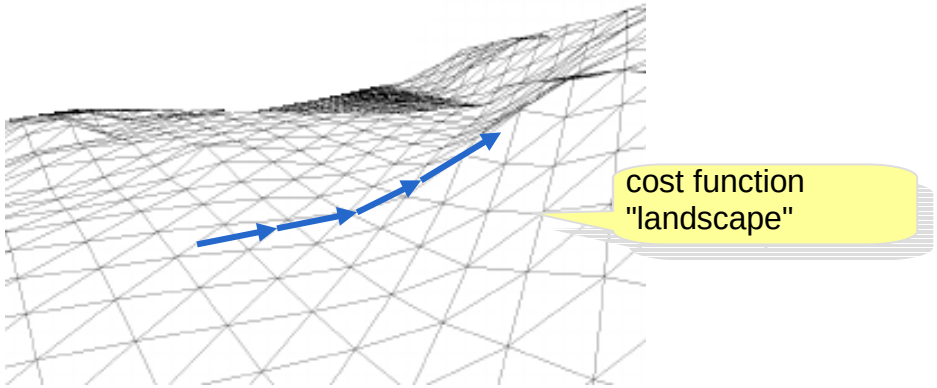


Slides based on lecture by Van Larhoven

Iterative Improvement

Iterative Improvement

- General method to combinatorial optimization problems



Principle:

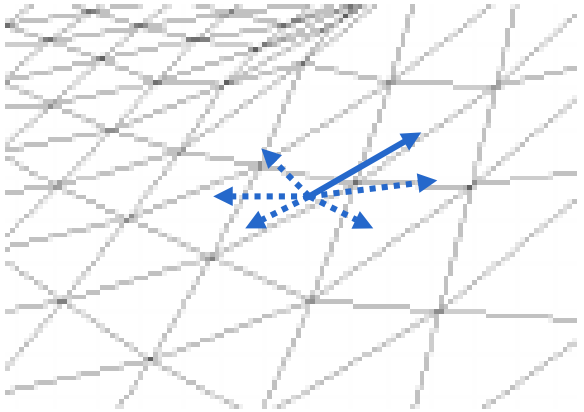
- Start with initial configuration
- Evaluate Gradient of some cost function (or fitness function)
- Go to direction of Gradient
- Stop if quality is sufficiently high, if no improvement can be found or after some fixed time

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}.$$

Iterative Improvement - without Gradient

School of Engineering

- General method to solve combinatorial optimization problems



Principle:

- Start with initial configuration
- Randomly or systematically search neighborhood and select a most promising neighbor as candidate
- Evaluate some cost function (or fitness function) and accept candidate if "better"; if not, select another neighbor
- Stop if quality is sufficiently high, if no improvement can be found or after some fixed time

Iterative Improvement Prerequisites

Needed are:

- A method to generate initial configuration
- A transition or generation function to find a neighbor as next candidate
- A cost function
- An Evaluation Criterion
- A Stop Criterion

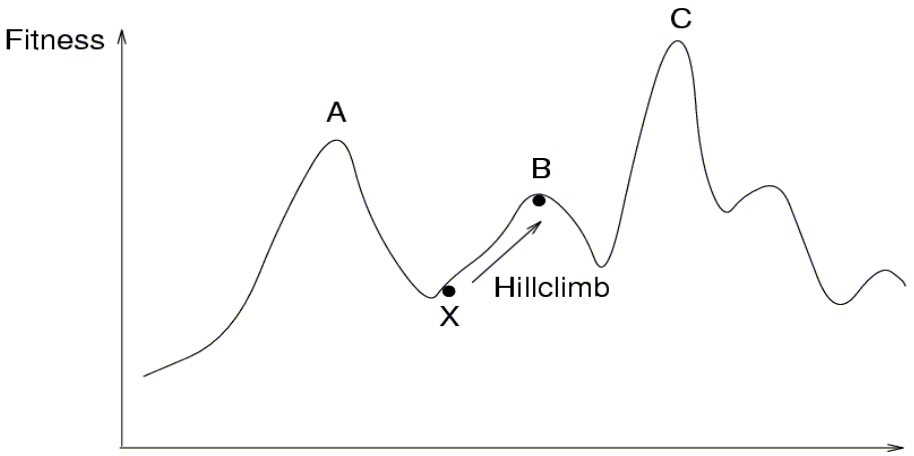
Iterative Improvement Pros- and Cons

Simple Iterative Improvement or Hill Climbing:

- Candidate is always and only accepted if cost is lower (or fitness is higher) than current configuration
- Stop when no neighbor with lower cost (higher fitness) can be found

Disadvantages:

- Local optimum as best result
- Local optimum depends on initial configuration
- Generally no upper bound on iteration length



How to cope with disadvantages

School of Engineering

- Repeat algorithm many times with different initial configurations
- Use information gathered in previous runs
- Use a more complex Generation Function to jump out of local optimum

- Use a more complex Evaluation Function:
  - Do sometimes accept candidates with higher cost to escape from local optimum
  - Adapt the parameters of this Evaluation Function during execution
  - Based upon the analogy with the simulation of the annealing of solids

Simulated Annealing

Simulated Annealing

School of Engineering

Use a more complex Evaluation Function:

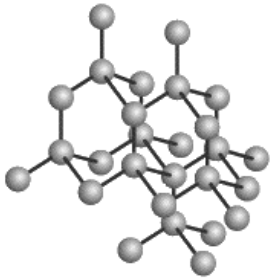
- Do sometimes accept candidates with higher cost to escape from local optimum
- Adapt the parameters of this Evaluation Function during execution
- Based upon the analogy with the simulation of the annealing of solids

- Other Names
  - Monte Carlo Annealing
  - Statistical Cooling
  - Probabilistic Hill Climbing
  - Stochastic Relaxation
  - Probabilistic Exchange Algorithm

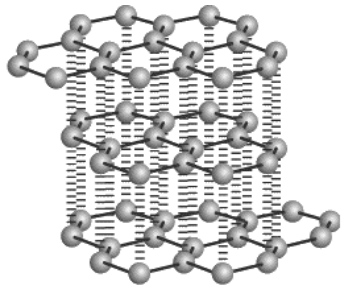
Analogy

- Slowly cool down a heated solid, so that all particles arrange in the ground energy state -> e.g. crystal
- At each temperature wait until the solid reaches its thermal equilibrium
- Probability of being in a state with energy *E* :

$$P(\mathbf{E} = E_i) = exp(-E / k_B * T*Z_{T0})$$



Diamant



Graphite

E	Energy/Cost
T	Temperature
k_B	Boltzmann constant
Z_{T0}	Normalization factor (ev. temperature dependant)

Simulation of cooling (Metropolis 1953)

- At temperature T :
- Perturb (randomly) the current state to a new state
- ΔE is the difference in energy between current and new state
- If $\Delta E < 0$ (new state is lower), accept new state as current state
- If $\Delta E > 0$, accept new state with probability

$$P\ (accept) = exp\ (-\ \Delta E\ /\ k_B * Z_{T0} / T)$$

- Eventually the systems evolves into thermal equilibrium at temperature T ; then the formula mentioned before holds
- When equilibrium is reached, temperature T can be lowered and the process can be repeated

Simulated Annealing

- Same algorithm can be used for combinatorial optimization problems:
- Energy *E* corresponds to the Cost function *C*
- Temperature *T* corresponds to control parameter *t*

$$P(\text{config} = \text{config}_i) = exp(-\Delta C(i) * K_{c0} / c)$$

C	Cost
c	Temperature of Control parameter (slowly decreasing -> 0)
K_{c0}	Normalization factor
	chose so that there is a 50% initial chance

Algorithm

```
initialize;
count = 0;
bolzCount = 0;
acceptCount = 0;
LOOP
    count++;
    perturb ( config.i → config.j, ΔC_ij);
    IF ΔC_ij < 0 THEN accept
    ELSE IF ΔC_ij > 0 AND exp(-ΔC_ij/c) > random[0,1] THEN
       accept; bolzCount++;
    IF accept THEN update(config.j); acceptCount++;
     measureProgress
    next_lower (c)
END
```

change configuration randomly

change in cost

Algorithm Termination/Acceptance

- For termination condition measure progress
- e.g. count accepted transitions in 1000 iterations

```
measureProgress:
   IF count % 1000 == 0 THEN
        IF acceptCount == 0 BREAK
         ELSE bolzCount = 0; acceptCount = 0;
  END
```

Parameters

School of Engineering

- Choose the start value of *c* so that in the beginning nearly all perturbations are accepted (exploration), but not too big to avoid long run times
- The function *next_lower* in the homogeneous variant is generally a simple function to decrease *c*, e.g. a fixed part (80%) of current *c*
- At the end *c* is so small that only a very small number of the perturbations is accepted (exploitation)

- If possible, always try to remember explicitly the best solution found so far; the algorithm itself can leave its best solution and not find it again

Performance

School of Engineering

- SA is a general solution method that is easily applicable to a large number of problems
- "Tuning" of the parameters (initial c, decrement of c, stop criterion) is relatively easy
- Generally the quality of the results of SA is good, although it can take a lot of time
- Results are generally not reproducible: another run can give a different result
- SA can leave an optimal solution and not find it again
  (so try to remember the best solution found so far)
- Proven to find the optimum under certain conditions; one of these conditions is that you must run forever

All the other Algorithms

All the other Algorithms

- There are Thousands of other interesting algorithms:

  https://en.wikipedia.org/wiki/Category:Algorithms

- This was only an introduction


Merry Christmas!
www.wallpapers.com