# Simulated Annealing

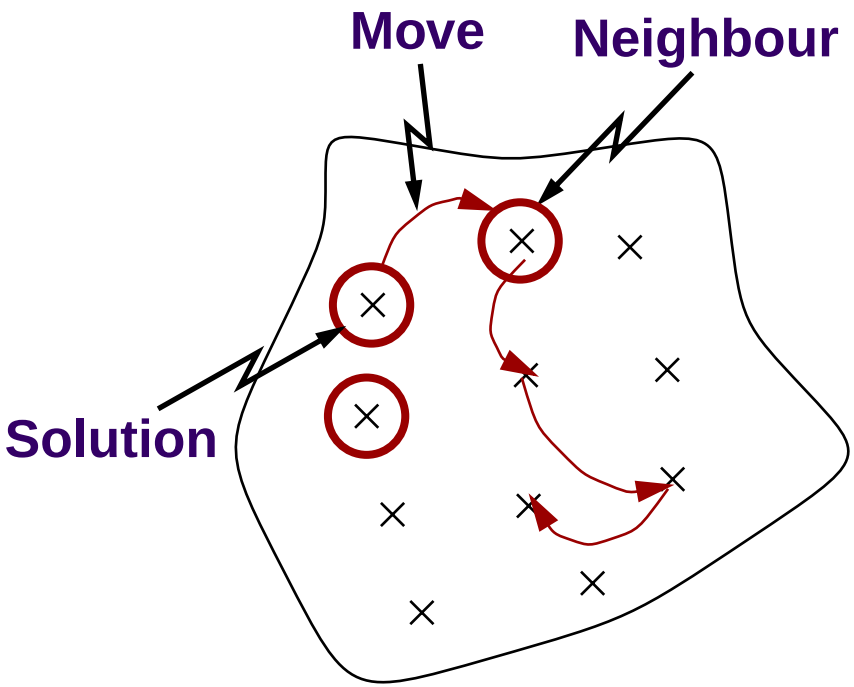
**Petru Eles**

**Department of Computer and Information Science (IDA)**
**Linköpings universitet**
**http://www.ida.liu.se/~petel/**

# Outline

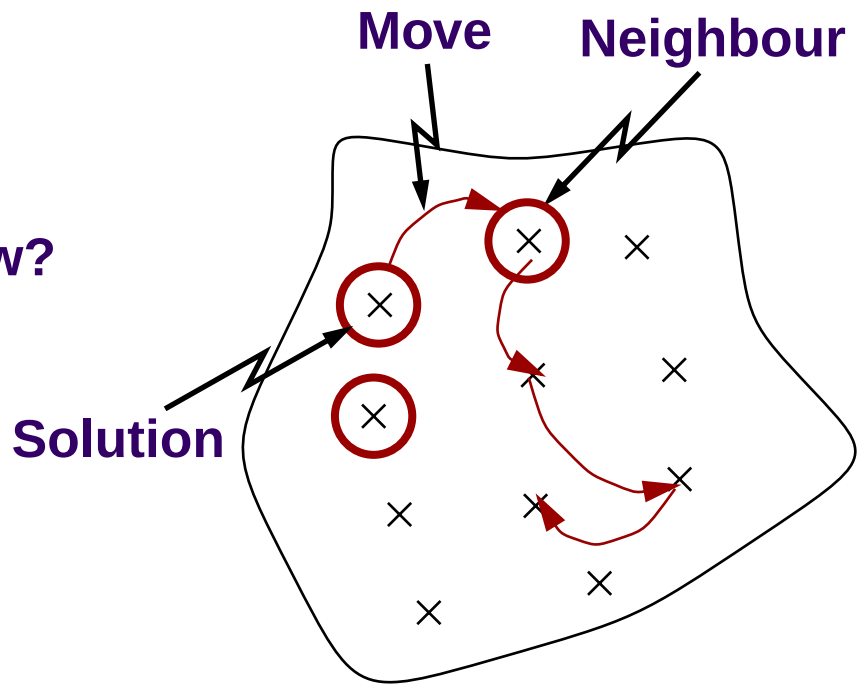
- Neighborhood Search
- Greedy Heuristics
- Simulated Annealing: the Physical Analogy
- Simulated Annealing Algorithm
- Theoretical Foundation
- Simulated Annealing Parameters
- Generic and Problem Specific Decisions
- Simulated annealing Examples
  - Traveling Salesman problem
  - Hardware/Software Partitioning

# Neighborhood Search



# Neighborhood Search

- Problems:
  - Moves
    - How do I get from one Solution to another?
  - Exploration strategy (you cannot try all alternatives!)
    - How many neighbors to try out?
    - Which neighbor to select?
    - What sequence of moves to follow?
  - When to stop?



# General Neighborhood Search Strategy

- ***neighborhood N(x)*** **of a solution *x* is a set of solutions that can be reached from *x* by a simple operation (*move*).**

**construct initial solution $x_0$; $x^{now} = x_0$**
***repeat***
    **Select new, acceptable solution $x' \in N(x^{now})$**
    **$x^{now} = x'$**
***until*** **stopping criterion met**
***return*** **solution corresponding to the minimum cost function**

# Greedy Heuristics

**When is a solution acceptable?**

construct initial solution $x_0$; $x^{now} = x_0$
*repeat*
    Select new, **acceptable** solution $x' \in N(x^{now})$
    $x^{now} = x'$
*until* stopping criterion met
*return* solution corresponding to the minimum cost function

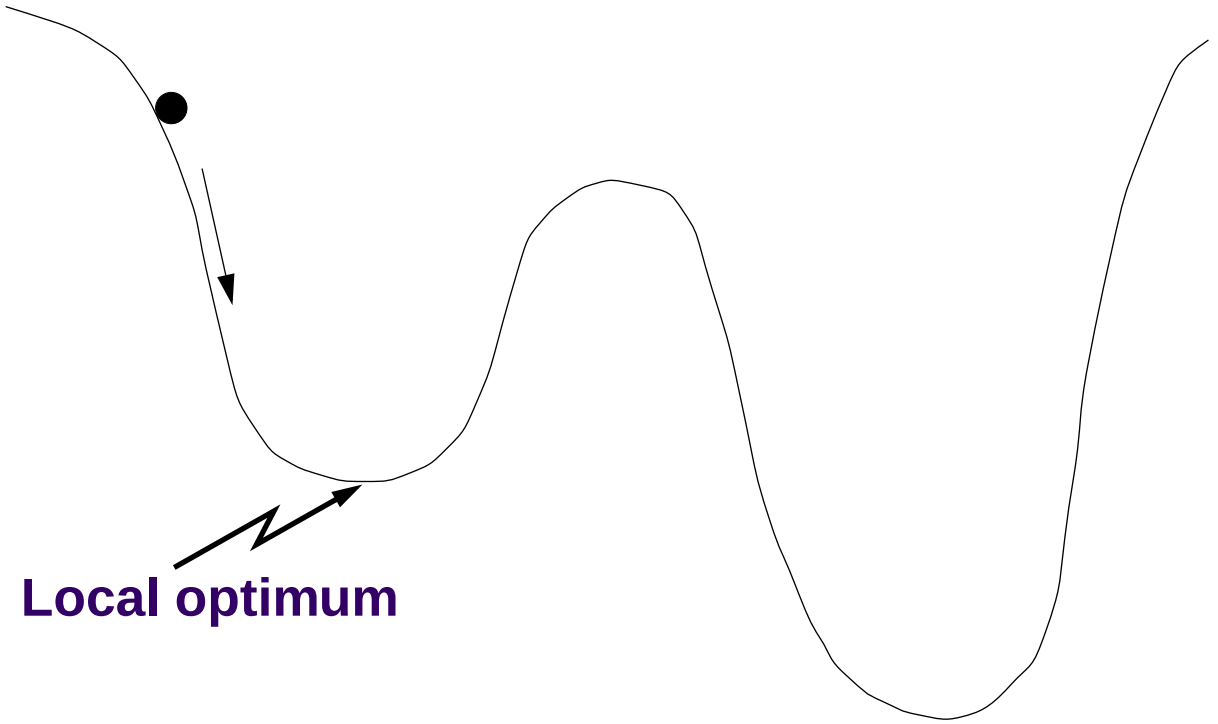
# Greedy Heuristics

**When is a solution acceptable?**

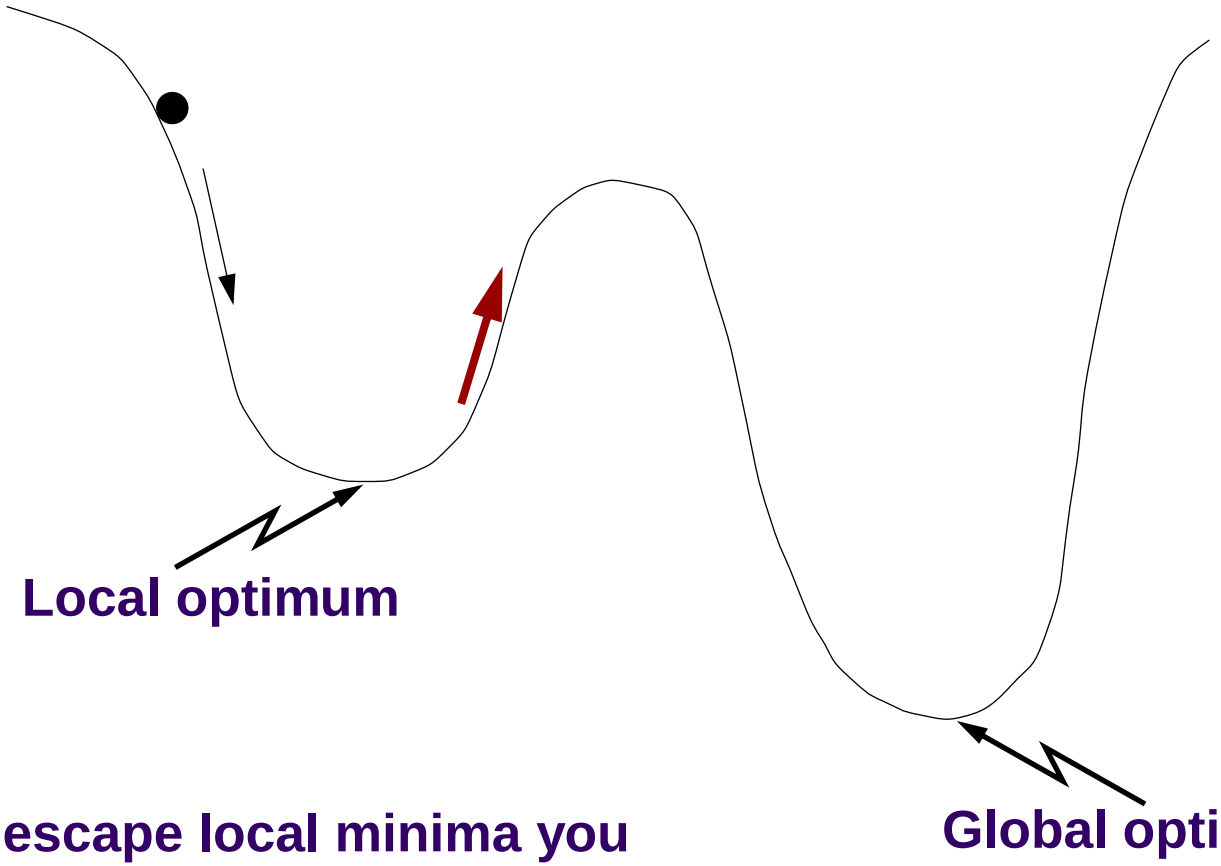
construct initial solution $x_0$; $x^{now} = x_0$
*repeat*
    Select new, **acceptable** solution $x' \in N(x^{now})$
    $x^{now} = x'$
*until* stopping criterion met
*return* solution corresponding to the minimum cost function

- *Greedy heuristics* always move from the current solution to the best neighboring solution.

# Greedy Heuristics



# Hill Climbing




- In order to escape local minima you have to allow uphill moves!

# Simulated Annealing Strategy

- SA is based on neighborhood search
- SA is a strategy which occasionally allows uphill moves.
  - Uphill moves in SA are applied in a controlled manner

# The Physical Analogy

- Metropolis - 1953: simulation of cooling of material in a heath bath;
  - A solid material is heated past its melting point and then cooled back into a solid state (annealing).
  - The final structure depends on how the cooling is performed
    - slow cooling $\rightarrow$ large crystal (low energy)
    - fast cooling $\rightarrow$ imperfections (high energy)

# The Physical Analogy

- Metropolis - 1953: simulation of cooling of material in a heath bath;
  - A solid material is heated past its melting point and then cooled back into a solid state (annealing).
  - The final structure depends on how the cooling is performed
    - slow cooling $\rightarrow$ large crystal (low energy)
    - fast cooling $\rightarrow$ imperfections (high energy)
- Metropolis' algorithm simulates the change in energy of the system when subjected to the cooling process; the system converges to a final "frozen" state of a certain energy.

# The Physical Analogy

- Metropolis regarded the material as a system of particles.
- His simulation follows the energy of the particles with changing temperature
- According to thermodynamics:
  - at temperature T, the probability of an increase in energy of $\Delta$E is:

$$p(\Delta E) = e^{-\Delta E / kT}$$

***k*** **is the Boltzmann constant**

# The Metropolis Simulation

```
set initial temperature
repeat
        for a predetermined number of times do
                generate a perturbation
                if energy decreased then
                        accept new state
                else
                        accept new state with probability p(ΔE)
                end if
        end for
        decrease temperature
until frozen
```

# Simulated Annealing Algorithm

**Kirkpatrick - 1983:** The Metropolis simulation can be used to explore the feasible solutions of a problem with the objective of converging to an optimal solution.

| Thermodynamic simulation | SA Optimization |
|---|---|
| System states | Feasible solutions |
| Energy | Cost |
| Change of state | Neighboring solution |
| Temperature | Control parameter |
| Frozen state | Solution (close to optimal) |

# Simulated Annealing Algorithm

construct initial solution $x_0$; $x^{now} = x_0$
set initial temperature $T = TI$
*repeat*
    *for* $i = 1$ *to* $TL$ *do*
        generate randomly a neighbouring solution $x' \in N(x^{now})$
        compute change of cost $\Delta C = C(x') - C(x^{now})$
        *if* $\Delta C \leq 0$ then
            $x^{now} = x'$ (accept new state)
        *else*
            Generate $q = random(0,1)$
            *if* $q < e^{-\Delta C/T}$ then $x^{now} = x'$ *end if*
        *end if*
    *end for*
    set new temperature $T = f(T)$
*until* stopping criterion
*return* solution corresponding to the minimum cost function

# Theoretical Foundation

- The behaviour of SA can be modeled using Markov chains.

- For a given temperature, one homogeneous chain
  - transition probability $p_{ij}$ between state *i* and state *j* depends only on the two states.

- But we have a sequence of different temperatures

a number of different homogeneous chains

a single non-homogeneous chain

# Theoretical Foundation

**For optimal convergence:**

- **With homogeneous chains:**
  - **the number of iterations at any temperature has to be at least quadratic in the size of the solution space.**

    **Solution space is exponential!**

- **With non-homogeneous chain:**
  - **cooling schedule which guarantees asymptotic convergence:**

    $t_k = c/\log(1+k)$ $c$: depth of the deepest local minimum

    **Number of iterations exponential!**

# Theoretical Foundation

**For optimal convergence:**

- **With homogeneous chains:**
  - **the number of iterations at any temperature has to be at least quadratic in the size of the solution space.**

    **Solution space is exponential!**

- **With non-homogeneous chain:**
  - **cooling schedule which guarantees asymptotic convergence:**

    $t_k = c/\log(1+k)$ $c$: depth of the deepest local minimum

    **Number of iterations exponential!**

- **These results are of no practical importance.**

# SA Parameters

construct initial solution $x_0$; $x^{now} = x_0$
set initial temperature $T = TI$
*repeat*
    *for* $i = 1$ *to* $TL$ *do*
        generate randomly a neighbouring solution $x' \in N(x^{now})$
        compute change of cost $\Delta C = C(x') - C(x^{now})$
        *if* $\Delta C \leq 0$ then
            $x^{now} = x'$ (accept new state)
        *else*
            Generate $q = random(0,1)$
            *if* $q < e^{-\Delta C / T}$ then $x^{now} = x'$ *end if*
        *end if*
    *end for*
    set new temperature $T = f(T)$
*until* stopping criterion
*return* solution corresponding to the minimum cost function

# SA Parameters

Two kinds of decisions have to be taken heuristically:

- Generic decisions
  - Can be taken without a deep insight into the particular problem.
  - Are tuned experimentally.
- Problem specific decisions
  - Are related to the nature of the particular problem.
  - Need a good understanding of the problem

# Generic Decisions

- initial temperature (*TI*)
- temperature length (*TL*)
- cooling ratio (function *f*)

cooling schedule

- stopping criterion

# Problem Specific Decisions

- space of feasible solutions and neighborhood structure
- cost function (*C*)
- starting solution

# Initial Temperature

- *TI* must be high enough - in order the final solution to be independent of the starting one.

**Are there any rules?**

# Initial Temperature

- *TI* must be high enough - in order the final solution to be independent of the starting one.

**Are there any rules?**

- If maximal difference in cost between neighboring solutions is known, *TI* can be calculated so that increases of that magnitude are initially accepted with sufficiently large probability:

$$p_{in} = e^{-\Delta C_{max}/T}$$

- Before starting the effective algorithm a heating procedure is run:
  - the temperature is increased until the proportion of accepted moves to total number of moves reaches a required value.

# Initial Temperature

- ***TI*** **must be high enough - in order the final solution to be independent of the starting one.**

**Are there any rules?**

- **If maximal difference in cost between neighboring solutions is known, *TI* can be calculated so that increases of that magnitude are initially accepted with sufficiently large probability:**

$$p_{in} = e^{-\Delta C_{max}/T}$$

- **Before starting the effective algorithm a heating procedure is run:**
  - **the temperature is increased until the proportion of accepted moves to total number of moves reaches a required value.**

**But, in any case, experimental tuning is needed!**

# Temperature Length and Cooling Ratio

The rate at which temperature is reduced is governed by:

- Temperature length (*TL*): number of iterations at a given temperature
- Cooling ratio (*f*): rate at which temperature is reduced

Alternatives
- large number of iterations at few temperatures
- small number of iterations at many temperatures

# Temperature Length and Cooling Ratio

- In practice, very often:
  - $f(T) = aT$, where $a$ is a constant, $0.8 \leq a \leq 0.99$

    (most often closer to 0.99)

usually, cooling is slow

# Temperature Length and Cooling Ratio

- How long to stay at a temperature?
  - The number of iteration at each temperature depends on:
    - size of the neighborhood
    - size of the solution space.
  - The number of iterations may vary from temperature to temperature:
    - It is important to spend sufficiently long time at lower temperatures.

increase the *TL* as you go down with *T*

# Temperature Length and Cooling Ratio

- TL can be also determined using feedback from the SA process:

  - Accept a certain number of moves before decreasing temperature.

small number of iterations at high temperature
large number of iterations at small temperatures

  - Impose a maximum number of iterations at a temperature!

# Temperature Length and Cooling Ratio

- An extreme approach:

  - Execute one single (!) iteration at a temperature.
  - Reduce temperature extremely slowly:

$$f(T) = T/(1 + \beta), \qquad \text{with } \beta \text{ suitably small.}$$

# Stopping Criterion

- In theory temperature decreases to zero.
- Practically, at very small temperatures the probability to accept uphill moves is almost zero.
- Criteria for stopping:
  - A given minimum value of the temperature has been reached.
  - A certain number of iterations (or temperatures) has passed without acceptance of a new solution.
  - The proportion of accepted moves relative to attempted moves drops below a given limit.
  - A specified number of total iterations has been executed

# Problem Specific Decisions

- Neighborhood structure
  - The neighborhood structure depends on the solution space and on the selected moves.
    - Every solution should be *reachable* from every other.
    - Keep the neighborhood small:

      Can be adequately explored in few iterations. :)
      *but*
      No big improvements can be expected from one move. :(

# Problem Specific Decisions

- Cost function
  - Should be calculated quickly - possibly incrementally.

- The starting solution
  - Generated randomly.
  - Good solution (possibly produced by another heuristics);
    in this case the starting temperature should be lower.
  - Starting solution shouldn't be "too good" because it's difficult to escape from its neighborhood.

# Postprocessing

- You keep the best ever result as the “final” solution.

- Make sure that the local minimum close to the “final” solution is reached: run a small, quick greedy optimization.

# Simulated Annealing Examples

- Travelling Salesman

- Hardware/Software Partitioning

# SA Examples: Travelling Salesman Problem

**A salesman has to travel to a number of cities and then return to the initial city; each city has to be visited once. The objective is to find the tour with minimum distance.**

**<u>In graph theoretical formulation</u>:**

**Find the shortest Hamiltonian circuit in a complete graph where the nodes represent cities. The weights on the edges represent the distance between cities. The cost of the tour is the total distance covered in traversing all cities.**

# TSP: Cost Function

- If the problem consists of *n* cities $c_i$, i = 1, .., n, any tour can be represented as a permutation of numbers 1 to *n*.

  $d(c_i,c_j) = d(c_j,c_i)$ is the distance between $c_i$ and $c_j$.

- Given a permutation $\pi$ of the *n* cities, $v_i$ and $v_{i+1}$ are adjacent cities in the permutation. The permutation $\pi$ has to be found that minimizes:

$$\sum_{i=1}^{n-1} d(v_i,v_{i+1}) + d(v_n,v_1)$$

- The size of the solution space is (*n*-1)!/2

# TSP: Moves&Neighborhood

- *k-neighborhood* of a given tour is defined by those tours obtained by removing *k* links and replacing them by a different set of *k* links, in a way that maintains feasibility.

- If $k > 2$, there are several ways of reconnecting after the *k* links have been removed.

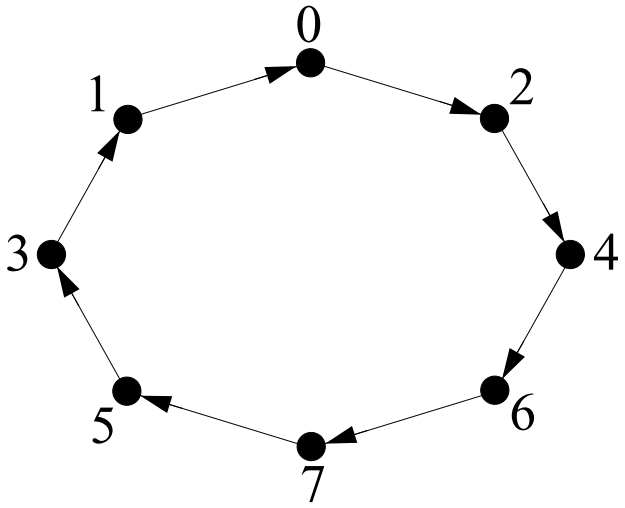
- For $k = 2$, there is only one way of reconnecting the tour after two links have been removed.

# TSP: Moves&Neighborhood

- With $k = 2$:
  - Size of the neighborhood: $n(n - 1)/2$
  - Any tour can be obtained from any other by a sequence of such moves.

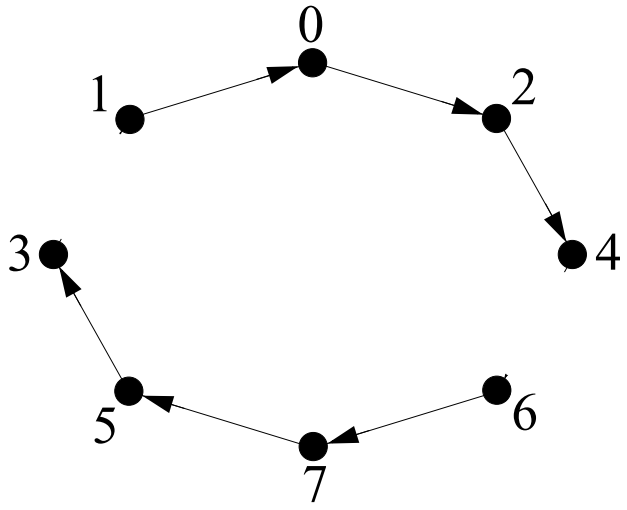
# TSP: Moves&Neighborhood

Permutation:
[0 2 4 6 7 5 3 1]



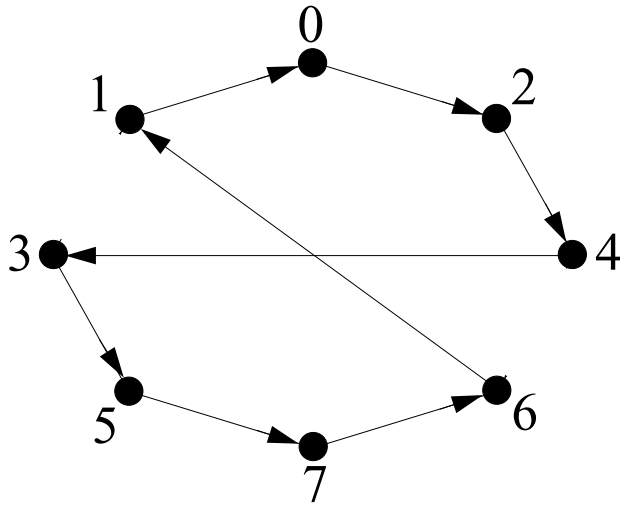
# TSP: Moves&Neighborhood

links $(v_3,v_1)$, $(v_4,v_6)$ are removed



# TSP: Moves&Neighborhood

Permutation:
[0 2 4 3 5 7 6 1]



# TSP: Moves&Neighborhood

- $v_i$ is the city in position $i$ of the tour ($i^{th}$ position in the permutation):

  *remove* $(v_i, v_{i+1})$ and $(v_j, v_{j+1})$

  ↓

  *connect* $v_i$ to $v_j$ and $v_{i+1}$ to $v_{j+1}$

- All 2-neighbors of a certain solution are defined by the pair $i$, $j$ so that $i < j$.

- A neighboring solution is generated by randomly generating $i$ and $j$.

- The change of the cost function can be computed incrementally:

  $\Delta C = d(v_i,v_j) + d(v_{i+1},v_{j+1}) - d(v_i,v_{i+1}) - d(v_j,v_{j+1})$

# TSP: Generic Parameters and Results

- 100 city problem; optimal solution: *C* = 21247.
  - Best solution for *Tl* = 1500, $\alpha$=0.63: *C* = 21331
    - Time = 310 s (Sun4/75)
    - Standard deviation over 10 trials: 30.3;
    - Average cost: 21372
  - Best solution for *Tl* = 1500, $\alpha$=0.90: *C* = 21255.
    - Time = 1340 s (Sun4/75)
    - Standard deviation over 10 trials: 27.5;
    - Average cost: 21284

# TSP: Generic Parameters and Results

- 57 city problem; optimal solution: $C = 12955$
  - Optimal solution for 15% of runs.
  - Time 673 s (Sequent Balance 8000)
  - All non-optimal results within less than 1% of optimum.

# SA Examples: Hardware/Software Partitioning

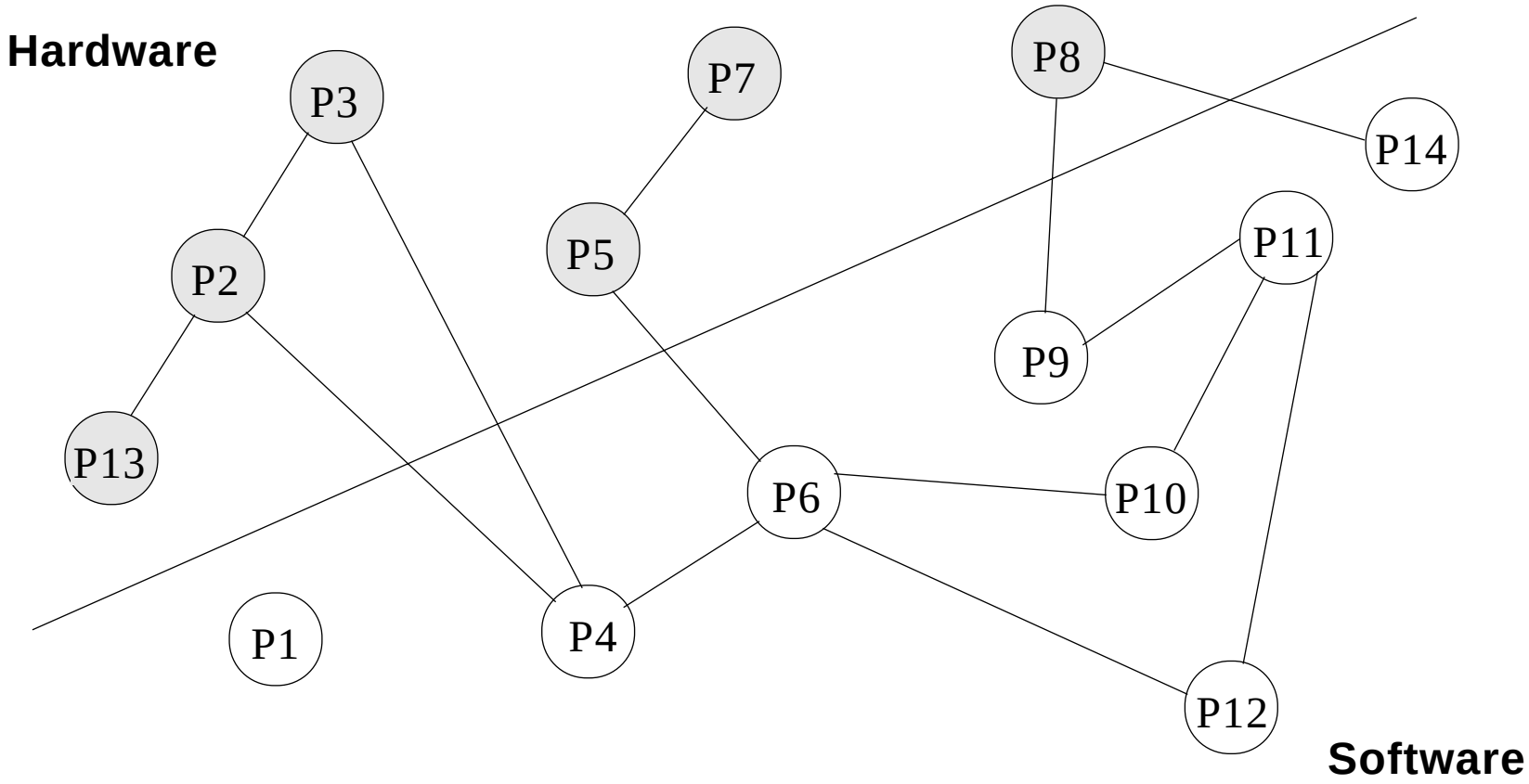
**Input:**

- The *process graph*: an abstract model of a system:
  - Each node corresponds to a process.
  - An edge connects two nodes if and only if there exists a direct communication channel between the corresponding processes
  - Weights are associated to each node and edge:
    - Node weights reflect the degree of suitability for hardware implementation of the corresponding process.
    - Edge weights measure the amount of communication between processes

**Output:**

- Two subgraphs containing nodes assigned to hardware and software respectively.

# SA Examples: Hardware/Software Partitioning



# SA Examples: Hardware/Software Partitioning

**Weight assigned to nodes:**

$$W2_i^N = M^{CL} \times K_i^{CL} + M^U \times K_i^U + M^P \times K_i^P - M^{SO} \times K_i^{SO}$$

$K_i^{CL}$ **is equal to the RCL (relative computation load) of process *i*, and thus is a measure of the computation load of that process;**

$K_i^U = \frac{Nr\_op_i}{Nr\_kind\_op_i}$ ; $K_i^U$ **is a measure of the uniformity of operations in process *i*;**

$K_i^P = \frac{Nr\_op_i}{L\_path_i}$ ; $K_i^P$ **is a measure of potential parallelism inside process *i*;**

$K_i^{SO} = \frac{\sum_{op_j \in SP_i} w_{op_j}}{Nr\_op_i}$ ; $K_i^{SO}$ **captures suitability for software implementation;**

# Hw/Sw Partitioning: Cost Function

## The cost function:

$$\underbrace{Q1 \times \sum_{(ij) \in cut} W1_{ij}^{E}}_{\text{amount of Hw-Sw comm.}} + \underbrace{Q2 \times \frac{\sum_{(i) \in Hw} \frac{\sum_{\exists (ij)} W2_{ij}^{E}}{W1_i^N}}{N_H}}_{\text{Ratio com/cmp of Hw part.}} - \underbrace{Q3 \times \left( \frac{\sum_{(i) \in Hw} W2_i^N}{N_H} - \frac{\sum_{(i) \in Sw} W2_i^N}{N_S} \right)}_{\text{Difference of average weights}}$$

## Restrictions:

$$\sum_{i \in H} H\_cost_i \leq Max^H$$

$$\sum_{i \in H} S\_cost_i \leq Max^S$$

$$W_i^N \geq \text{Lim1} \Rightarrow i \in Hw$$

$$W_i^N \leq \text{Lim1} \Rightarrow i \in Sw$$

# Hw/Sw Partitioning: Moves&Neighborhood

- Simple moves:
  - A node is randomly selected for being moved to the other partition.

- Improved move:
  - Together with the randomly selected node also some of its direct neighbors are moved; a direct neighbor is moved together with the selected node if its movement improves the cost function and does not violate any constraint.

# Hw/Sw Partitioning: Moves&Neighborhood

- A negative side effect of the improved move (revealed by experiences):
  - repeated move of the same or similar node groups from one partition to the other $\Rightarrow$ a reduction of the spectrum of visited solutions.

  - Movement of node groups is combined with that of individual nodes:

    Nodes are moved in groups with a certain probability *p*;

    experimentally: *p* = 0.75.

# Hw/Sw Partitioning: Generic Parameters and Results

## Cooling schedules

| number of nodes | TI | | TL | | a | |
|---|---|---|---|---|---|---|
| | SM | IM | SM | IM | SM | IM |
| 20 | 400 | 400 | 90 | 75 | 0.96 | 0.95 |
| 40 | 500 | 450 | 200 | 150 | 0.98 | 0.97 |
| 100 | 500 | 450 | 500 | 200 | 0.98 | 0.97 |
| 400 | 1400 | 1200 | 7500 | 2750 | 0.998 | 0.995 |

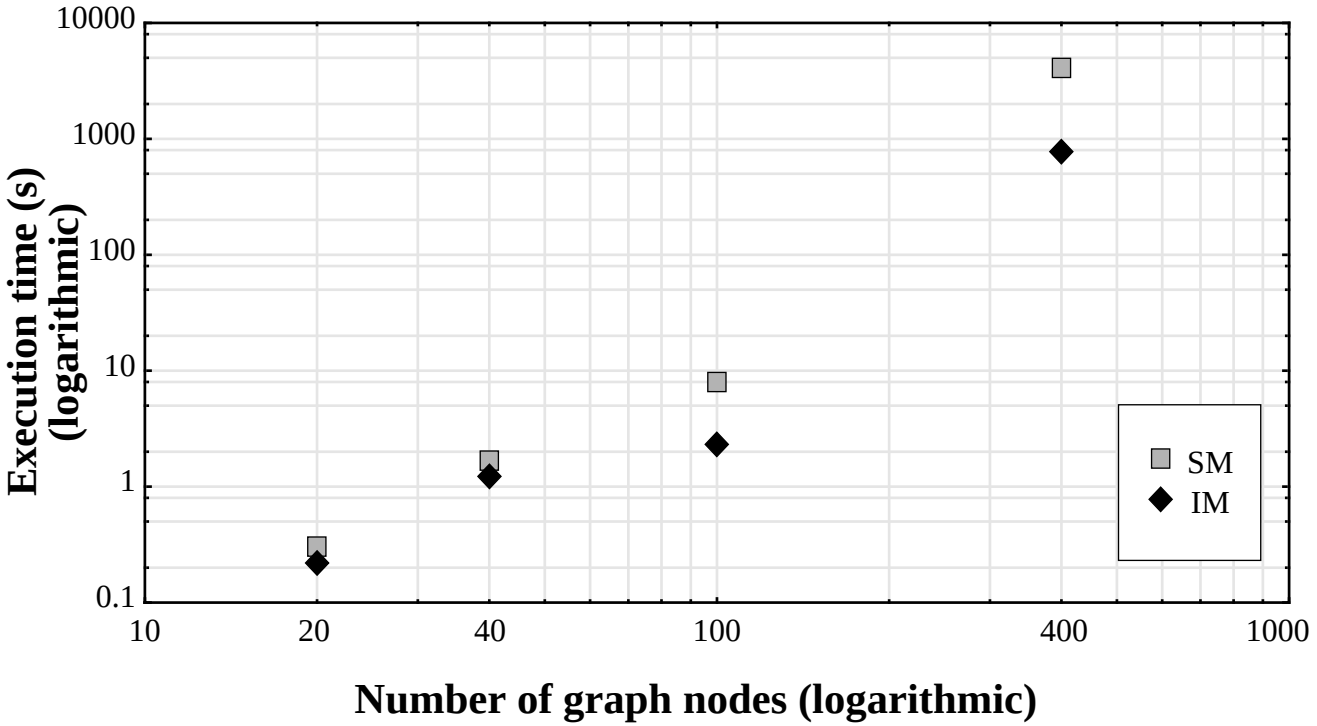
# Hw/Sw Partitioning: Generic Parameters and Results

**Partitioning time with SA**

(on SPARCstation 10)

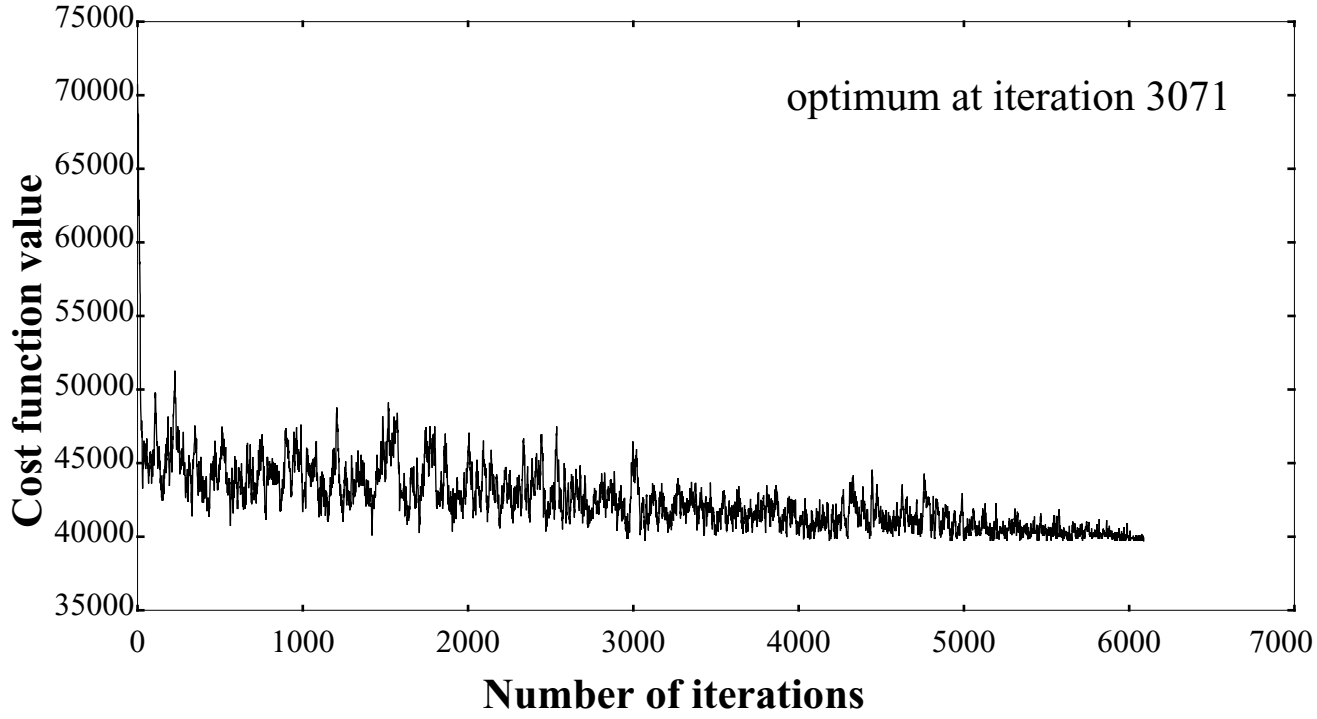
| number of nodes | CPU time (s) | | speedup |
|---|---|---|---|
| | SM | IM | |
| 20 | 0.28 | 0.23 | 22% |
| 40 | 1.57 | 1.27 | 24% |
| 100 | 7.88 | 2.33 | 238% |
| 400 | 4036 | 769 | 425% |

# Hw/Sw Partitioning: Generic Parameters and Results



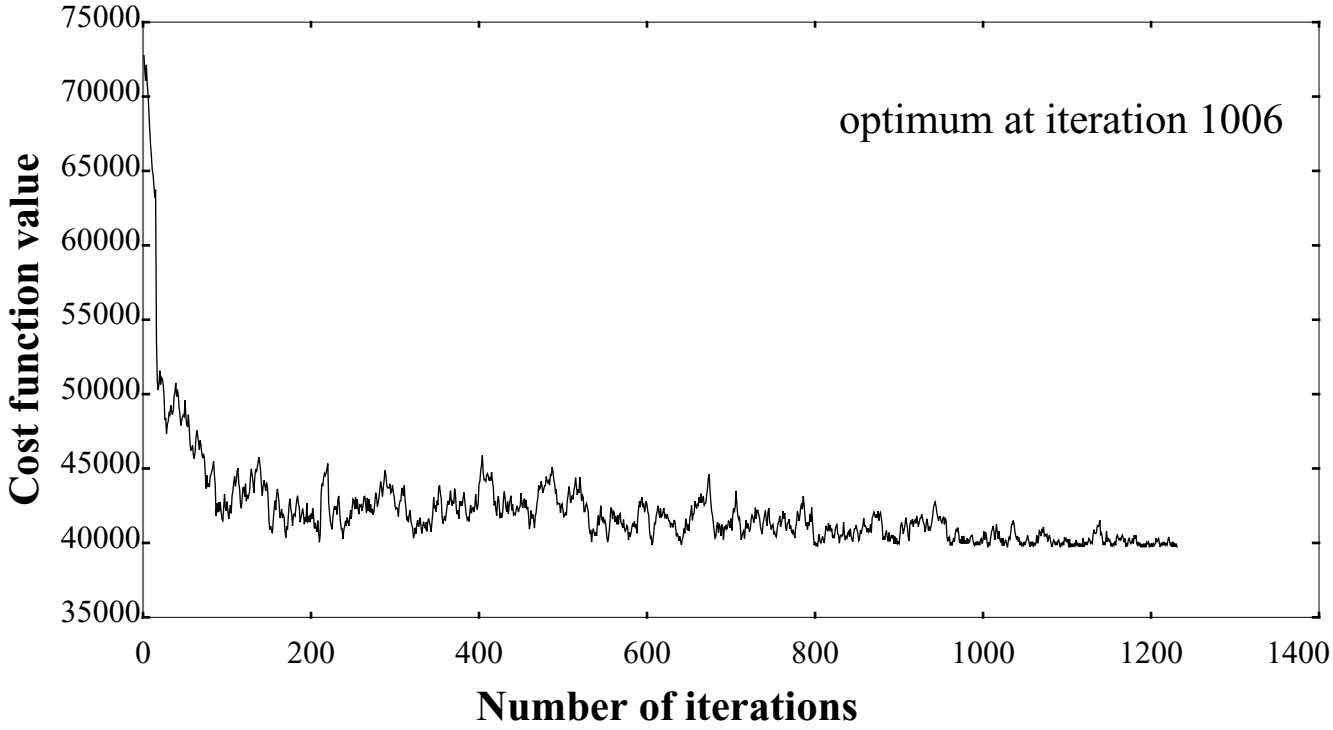
# Hw/Sw Partitioning: Generic Parameters and Results

- Variation of cost function during SA with simple moves for 100 nodes



# Hw/Sw Partitioning: Generic Parameters and Results

- Variation of cost function during SA with improved moves for 100 nodes



# Conclusions

- SA is based on neighborhood search and allows uphill moves.
- It has a strong analogy to the simulation of cooling of material.
- Uphill moves are allowed with a temperature dependent probability.
- Generic and problem-specific decisions have to be taken at implementation.
- Experimental tuning is very important!