Random Search Algorithms. Simulated Annealing

- Motivation
- Simple Random Search Algorithms
- Simulated Annealing Algorithms

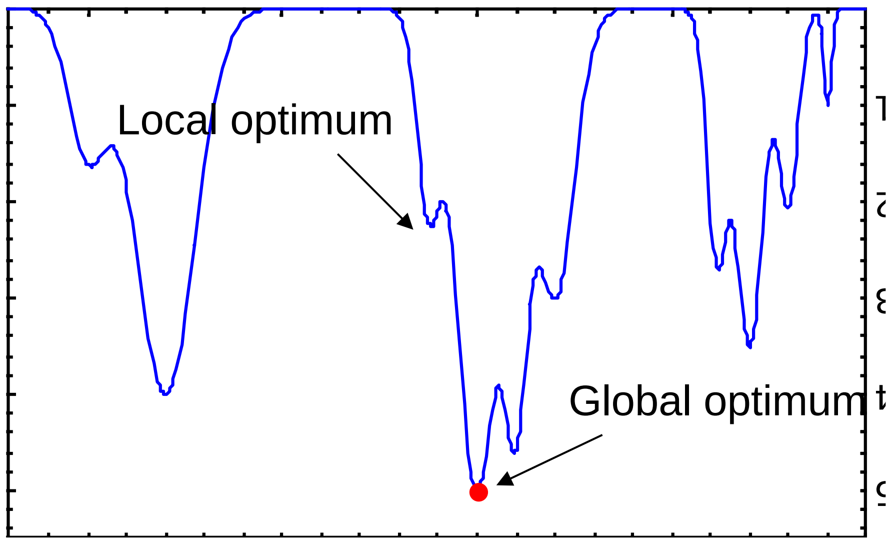
Motivation

Global optimization:

- Identify the global optimum of a function : f(x*)>=f(x), for all x
- If the objective function has also local optima then the local search methods (e.g. gradient-like methods as hill climbing) can be trapped in such local optima

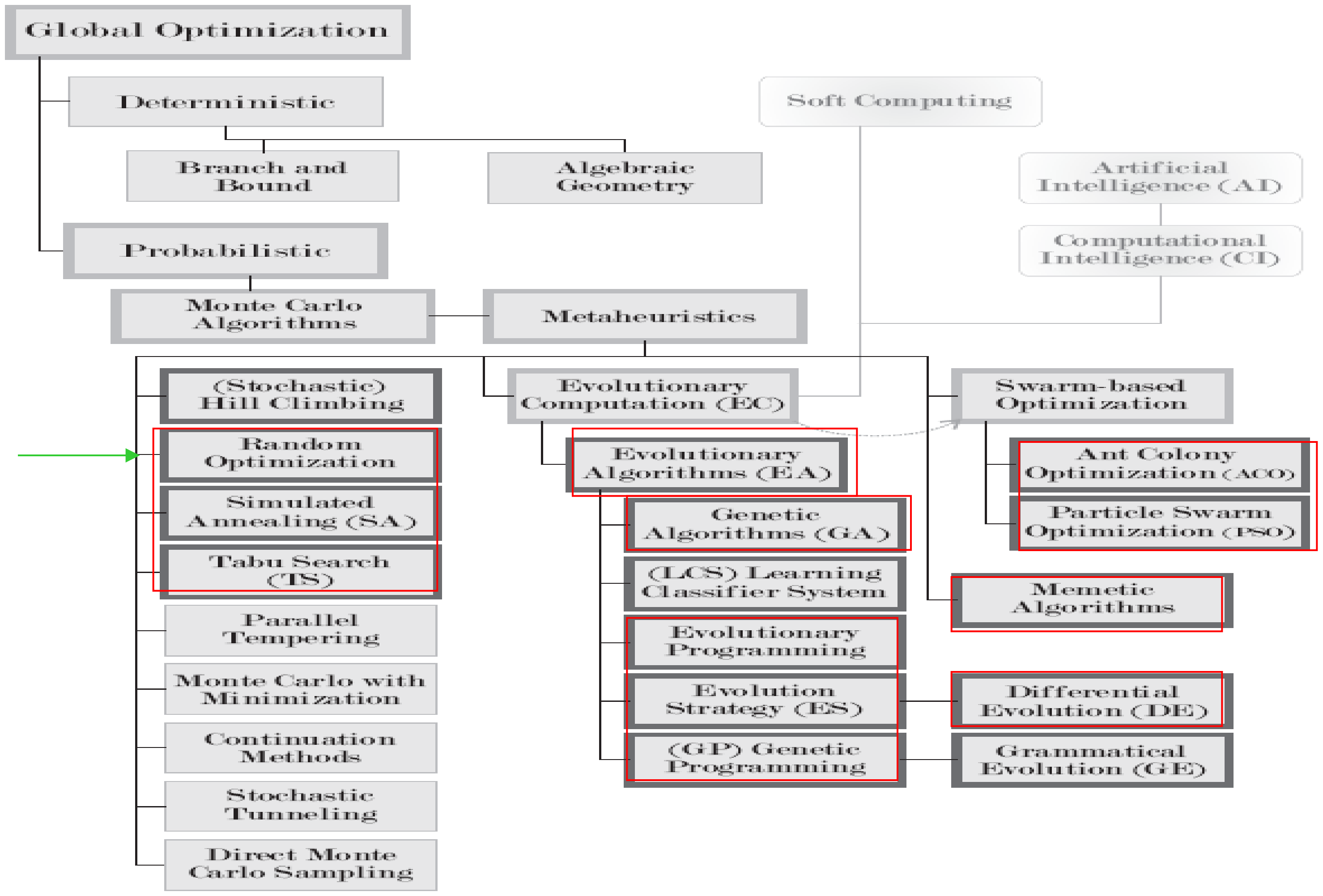
Example:

- Neural network training



Motivation

A taxonomy of global optimization methods



Random search algorithms

The local optima problem can be avoided by using random perturbations in the search process.

Example:

The simplest idea is to replace the search direction corresponding to the gradient with a random direction.

Advantages: the random search algorithms are very easy to be implemented; the objective function does not have to be smooth; it is enough if the function can be evaluated (even by simulations)

Disadvantages: the random search algorithms are not necessarily convergent in the usual sense; in most cases they are only convergent in a probabilistic sense.

Random search algorithms

Idea:

- the current approximation of the solution is randomly perturbed
- if the perturbed configuration is better than the current approximation then the perturbation is accepted

Problem: find x* which minimizes the function f

General structure:

```
Initialize x
Repeat
    If f(x+z)<f(x) then x:=x+z
Until " a stopping condition is satisfied "
```

Ex: Matyas algorithm (1960)

```
Init x(0)
k:=0
e:=0
REPEAT
    generate a perturbation vector
     $(z_1,\ldots z_n)$
    IF  f(x(k)+z)<f(x(k))
    THEN  x(k+1):=x(k)+z
              e:=0
     ELSE x(k+1):=x(k)
              e:=e+1
    k:=k+1
UNTIL (k=kmax) OR (e=emax)
```

Obs. The components of the random vector have the normal distribution

Problem: how to choose the parameters of the perturbation ?

Example: N(0,s)

Random values simulating

A variant to generate random values distributed according to the standard normal distribution is based on the Box-Muller algorithm

```
u:=random(0,1)   // value uniformly distributed in (0,1)
v:=random(0,1)
r:=sqrt(-2*ln(u));
z1:=r*cos(2*PI*v)
z2:=r*sin(2*PI*v)
RETURN z1,z2
```

Remark:

1. each call of this algorithm will produce two independent values
2. to obtain a value distributed according to N(m,s) use the property: N(m,s)=m+s*N(0,1)

Solis-Wets Algorithm (1981)

```
Init x(0)
k:=0; e:=0; m(0):=0
REPEAT
    generate a random vector $(z_1,\ldots z_n)$ having components with the
     distribution N(m(k),1)
    IF  f(x(k)+z)<f(x(k)) THEN x(k+1):=x(k)+z;   m(k+1):=0.4*z+0.2*m(k)
    ELSE
    IF f(x(k)-z)<f(x(k)) THEN x(k+1):=x(k)-z; m(k+1):=m(k)-0.4*z
    ELSE
    IF f(x(k)-z)>f(x(k)) AND f(x(k)+z)>f(x(k)) THEN
                                                        x(k+1):=x(k)
                                                        m(k+1):=0.5*m(k)
    k:=k+1
UNTIL (k=kmax)
```

Applications

Neural networks: the Matyas and Solis-Wets algorithms were used in training feedforward neural networks

Idea: the BackPropagation is replaced with a random search algorithm

Impact: the problem of local minima is partially solved in this way (by random perturbation the algorithm can escape from the neighborhood of a local minimum)

Remarks:

1. The evaluation of random algorithms can be conducted only in a statistical framework (the algorithm should be executed for several times and then the average and standard deviation of its results are computed)

2. These algorithms belong to the class of Monte-Carlo algorithms

Simulated Annealing

Idea:

- accept, with some probability, also perturbations which lead to an increase of the objective function (in the case of minimization problems)

Inspiration:

- SA algorithms are inspired by the process of restructuring the internal configuration in a solid which is annealed (e.g. crystallization process):
    - The solid is heated (up to the melting point): its particles are randomly distributed.
    - The material is the slowly cooled down: its particles are reorganized in order to reach a low energy state

Contributors: Metropolis(1953), Kirkpatrick, Gelatt, Vecchi (1983), Cerny (1985)

Simulated Annealing

Analogy:

Physical process:		Minimization problem:
• System energy	→	Objective function
• System state	→	Configuration (candidate solution)
• Change of the system state	→	Perturbation of the current configuration
• Temperature	→	Parameter which controls the optimization process

SA= (meta) heuristic method inspired by physical processes

Simulated Annealing

Some physics:

- Each state of the system has a corresponding probability
- The probability corresponding to a given state depends on the energy of the state and on the system temperature (Boltzmann distribution)

$$P_T(s) = \frac{1}{Z(T)} \exp\left(-\frac{E(s)}{k_B T}\right)$$

$$Z(T) = \sum_{s \in S} \exp\left(-\frac{E(s)}{k_B T}\right)$$

E(s) = energy of state s
T = temperature
Z(T)=partition function
(normalization factor)
k_B = Boltzmann constant

Simulated Annealing

Some physics:

- Large values of T (T goes to infinity): the argument of exp is almost 0 => the states have all the same probability

- Small values of T (T goes to 0): only the states with non-zero energy will have non-zero probabilities

$$P_T(s) = \frac{1}{Z(T)} \exp\left(-\frac{E(s)}{k_B T}\right)$$

$$Z(T) = \sum_{s \in S} \exp\left(-\frac{E(s)}{k_B T}\right)$$

$E(s)$ = energy of state s
T = temperature
Z(T)=partition function
(normalization factor)
k_B = Boltzmann constant

Simulated Annealing

How can we use these results from physics to solve an optimization problem ?

- It would be enough to generate configurations according to the Boltzmann distribution for smaller and smaller values of the temperature.

- **Problem:** it is difficult to compute the partition function Z(T) (it means to compute a sum over all possible configurations in the search space which is practically impossible for real-world problems – it would correspond to an exhaustive search)
- **Solution:** the distribution is approximated by simulating the evolution of a stochastic process (Markov chain) having as stationary distribution the Boltzmann distribution => Metropolis algorithm

Simulated Annealing

Metropolis algorithm (1953)

```
Init x(0)
k:=0
REPEAT
  x':=perturb(x(k))
   IF  f(x')<f(x(k)) THEN x(k+1):=x'  (unconditionally)
                        ELSE x(k+1):=x'
                                 with probability min{1,exp(-(f(x')-f(x(k))/T)}

   k:=k+1
UNTIL "a stopping condition is satisfied"
```

Simulated Annealing

Properties of the Metropolis algorithm

- Another acceptance probability:

  $$P(x(k+1)=x') = 1/(1+\exp((f(x')-f(x(k))/T))$$

- Implementation issue: assigning a value with a given probability is based on generating a random value in (0,1)

```
u:=Random(0,1)
IF u<P(x(k+1)=x') THEN x(k+1)=x'
                  ELSE x(k+1)=x(k)
```

- Large values for T -> high acceptance probability for any configuration (pure random search)

  Small values for T -> High acceptance probabilities only for the states with low energy values (greedy search - similar to a gradient descent method)

Simulated Annealing

Properties of the Metropolis algorithm

- The rules used to generate new configurations depend on the problem to be solved

Optimization in continuous domains

$x' = x + z$

$z = (z_1, \ldots, z_n)$

z_i : generated according to the distribution:

- $N(0,T)$
- Cauchy(T) (Fast SA)
- etc

Combinatorial optimization

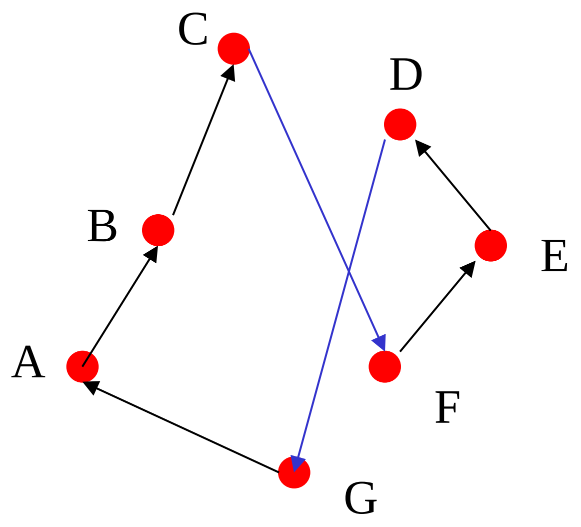
The new configuration is selected deterministically or randomly from the neighborhood of the current configuration

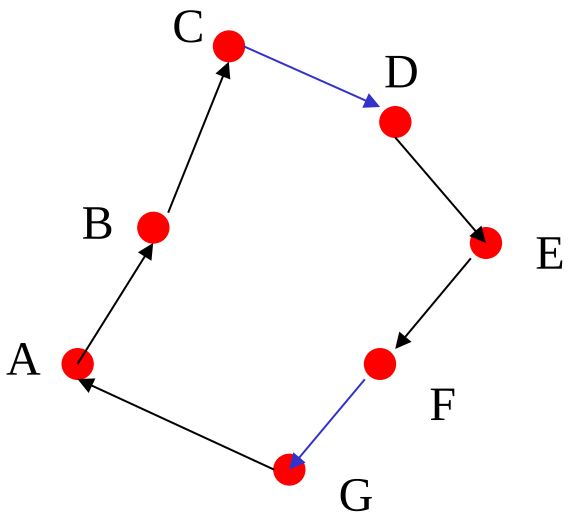
Example: TSP – 2-opt transformation

Simulated Annealing

TSP (Travelling Salesman Problem)

- Generating a new configuration from an existing one (2-opt transformation)





$$\text{ABCFEDG} \longrightarrow \text{ABCFEDG} \longrightarrow \text{ABCDEFG}$$

Simulated Annealing

Simulated Annealing = repeated application of the Metropolis algorithm for decreasing values of the temperature

General structure

```
Init x(0), T(0)
i:=0
REPEAT
    apply Metropolis  (for kmax iterations)
    compute T(i+1)
    i:=i+1
UNTIL T(i)<eps
```

Problem: How to choose the cooling scheme ?

Simulated Annealing

Cooling schemes:

$T(k)=T(0)/(k+1)$

$T(k)=T(0)/\ln(k+c)$

$T(k)=aT(k-1)$ ($a<1$, ex: $a=0.995$)

Remark. T(0) should be chosen such that during the first iterations almost all new configurations are accepted (this ensures a good exploration of the search space)

Simulated Annealing

Convergence properties:

If the following properties are satisfied:

- $Pg(x(k+1)=x'|x(k)=x)>0$ for any x and x' (the transition probability between any two configurations is non-zero)
- $Pa(x(k+1)=x'|x(k)=x)=\min\{1,\exp(-(f(x')-f(x))/T)\}$ (Metropolis acceptance probability)
- $T(k)=C/\lg(k+c)$ (logarithmic cooling schedule)

then $P(f(x(k))=f(x^*)) \to 1$ (x(k) is convergent in probability to the global minimum x^*)

Simulated Annealing

Variant: another acceptance probability (Tsallis)

$$P_a(x') = \begin{cases} 1, & \Delta f \le 0 \\ (1-(1-q)\Delta f / T)^{1/(1-q)}, & \Delta f > 0,\ (1\text{-}q)\Delta f \le 1 \\ 0, & \Delta f > 0,\ (1\text{-}q)\Delta f > 1 \end{cases}$$

$$\Delta f = f(x') - f(x)$$

$$q \in (0,1)$$

Simulated Annealing

Example: Travelling Salesman Problem (TSP)

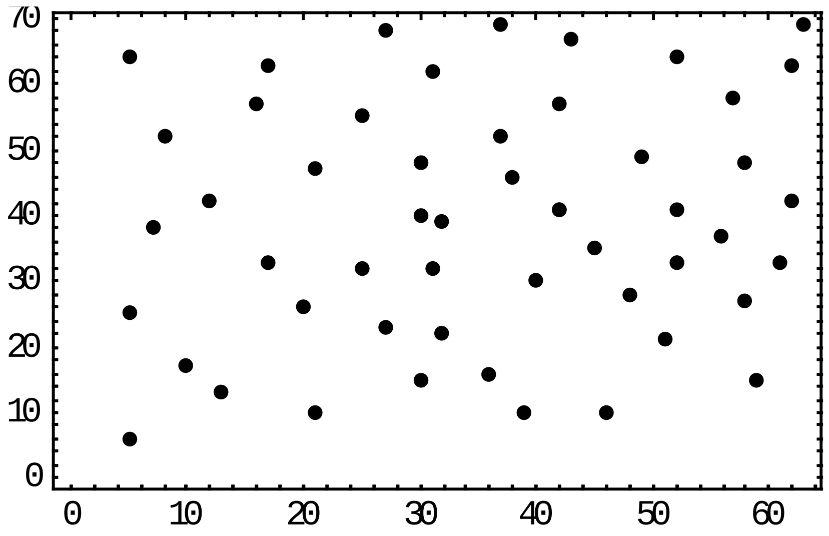
(TSPLib: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95)

Test instance: eil51 – 51 towns

Parameters:

- 5000 iterations; T is changed at each 100 iterations
- T(k)=T(0)/(1+log(k))

Location of towns

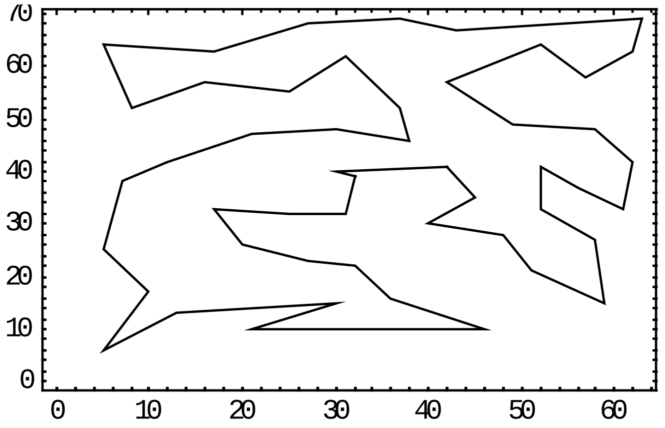


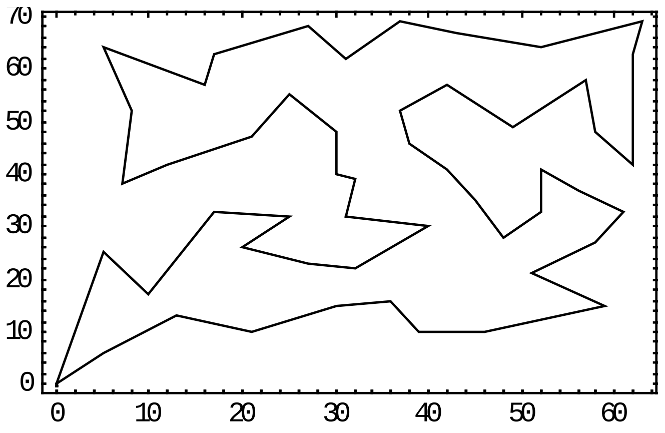
Simulated Annealing

Example: TSP

Test instance: eil51 (TSPlib)

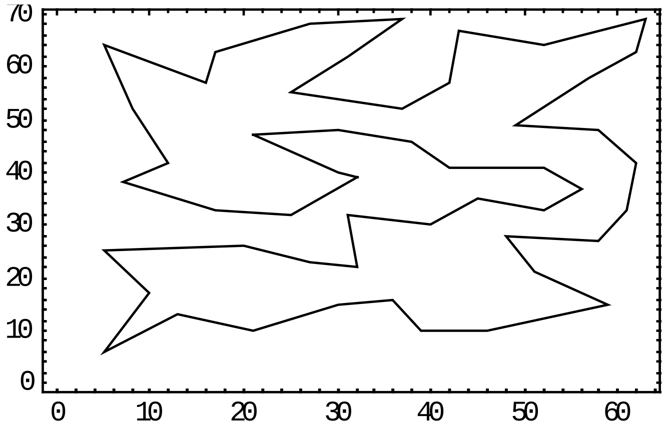
T(0)=5, cost=474.178





T(0)=10,cost=478.384

T(0)=1, cost=481.32



Minimal Cost: 426

Simulated Annealing

Example: timetabling

Problem: Let us consider a set of events/ activities (ex: lectures, exams), a set of rooms and some time slots.

Assign each event to a room and a time slot such that some constraints are satisfied

The constraints could be

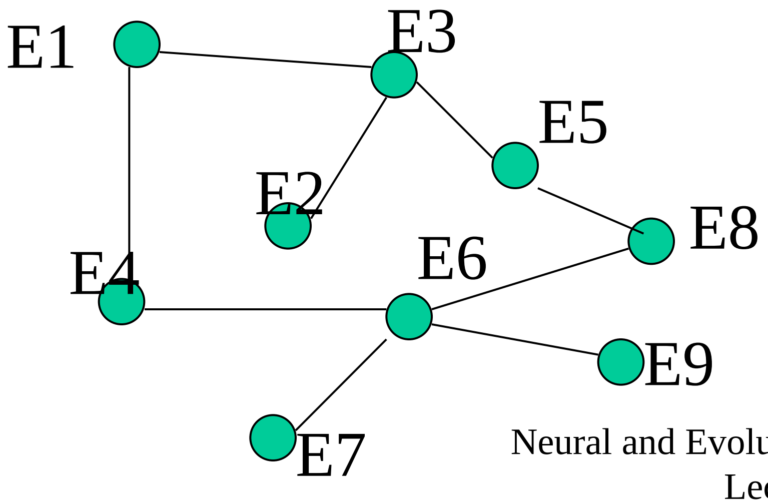
- Hard (strong)
- Soft (weak)

	T1	T2	T3
S1	E1	E3	E9
S2	E5		E8
S3	E6	E4	
S4	E2		E7

Simulated Annealing

- Hard constraints (the solution is acceptable only if they are satisfied):
  - Each event is scheduled only once
  - In each room there is only one event at a given time moment
  - There are no simultaneous events involving the same participants

Constraints graph: two nodes are connected if the corresponding events cannot be scheduled in the same time

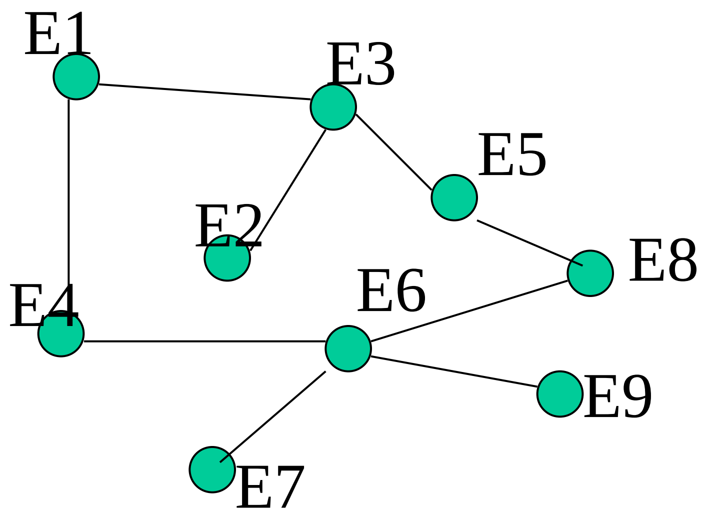


	T1	T2	T3
S1	E1	E3	E9
S2	E5		E8
S3	E6	E4	
S4	E2		E7

Simulated Annealing

- Soft constraints (the solution is better if they are satisfied):
  - There are no more than k events per day involving the same participant
  - There are no participants involved in only one event in a day

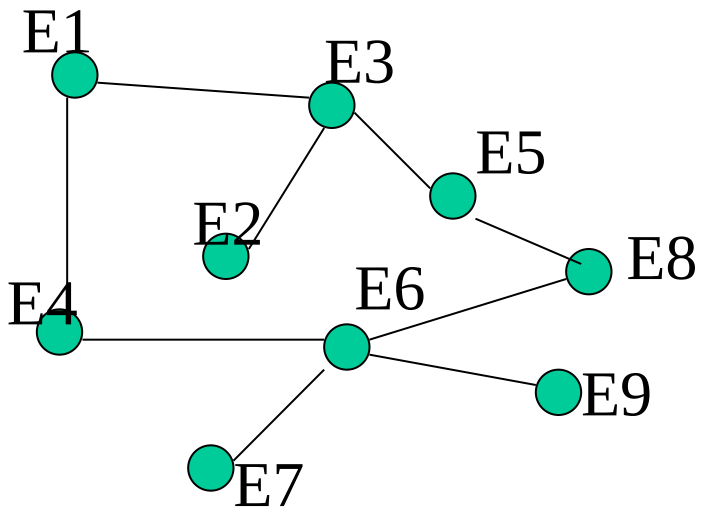
Idea: the soft constraints are usually included in the objective function by using the penalty method, e.g the number of participants involved in more than k events/day is as small as possible



	T1	T2	T3
S1	E1	E3	E9
S2	E5		E8
S3	E6	E4	
S4	E2		E7

Simulated Annealing

- Perturbing the current configuration:
  - The transfer of an event which does not satisfy a strong constraint

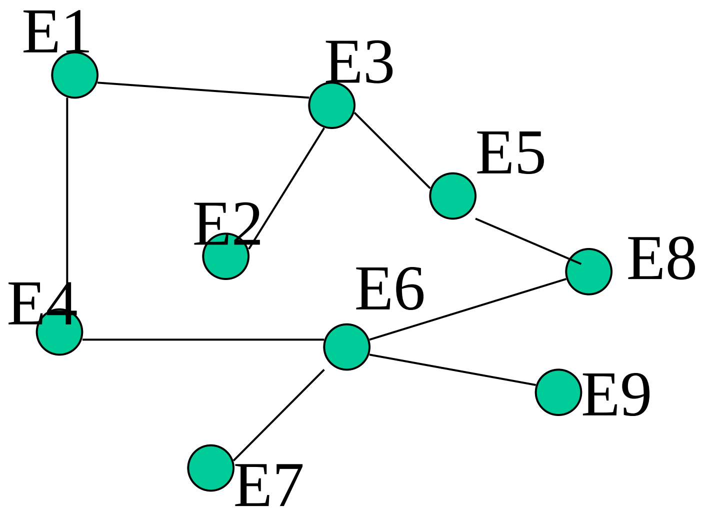


	T1	T2	T3
S1	E1	E3	E9
S2	E5	E6	E8
S3		E4	
S4	E2		E7

	T1	T2	T3
S1	E1	E3	E9
S2	E5		E8
S3	E6	E4	
S4	E2		E7

Simulated Annealing

- Perturbing the current configuration:
  - Swapping two events



	T1	T2	T3
S1	E1		E9
S2	E5	E6	E8
S3	E3	E4	
S4	E2		E7

	T1	T2	T3
S1	E1		E9
S2	E5	E3	E8
S3	E6	E4	
S4	E2		E7

Related techniques

Tabu Search

Creator: Fred Glover (1986)

Aim : combinatorial optimization method

Particularity:

- It is an iterative local search technique based on the exploration of the neighborhood of the current element (the neighborhood is defined as the set of all configurations which can be reached from the current configuration by applying once the search operator); the search operators are specific to the problem (e.g. 2-opt for TSP)
- It uses a list of prohibited configurations (called tabu list) which contains the configurations which are not acceptable in the following iterations (usually the tabu list is implemented as a circular list)

Related techniques – Tabu Search

Tabu Search - General Structure:

```
generate an initial configuration
REPEAT
    –  Select the best element in the neigborhood of the current
       configuration which is not included in the tabu list
    –  Update the tabu list
UNTIL <stopping condition>
```

Remarks:

1. If the neighborhood is too large then it is possible to evaluate only a sample from the neighborhood.
2. In order to improve the behavior one can apply periodically steps of intensification and diversification

Related techniques – Tabu Search

Intensification:

Aim: exploitation of promising regions

Implementation:

- Count the number of iterations when a component remains unchanged – the good components are those with large values of the corresponding counter
- Restart the search process from the best configuration by keeping frozen the good components

Related techniques – Tabu Search

Diversification:

Aim: exploration of the unvisited regions

Implementation:

- Compute the frequency of values used for each component. The values with low frequencies are under explored
- Restart the search process from configurations which contain under explored values or change the fitness function by penalizing the very frequent values of the components

Related techniques – VNS

VNS - Variable Neighborhood Search [Mladenovic, P. Hansen, 1997]

- Idea: they use a set of neighborhoods V_1, V_2,...,V_{kmax} which is explored in an incremental way; in each neighborhood the search is done using a local search method

- Obs: the neihborhood set for a configuration x is established depending on the problem to be solved but such that if k<k2 then the elements of $V_{k1}(x)$ can be obtained from x using fewer operations than are necessare to construct the elements of $V_{k2}(x)$
- Example: for the traveling salesman problem V_k (x) can contain the configurations (routes) obtained from x by applying k swaps of some randomly selected nodes

Related techniques – VNS

VNS - Variable Neighborhood Search [Mladenovic, P. Hansen, 1997]

General structure

```
Initialize x (randomly in the search space)
k:=1
WHILE k<=kmax DO
    select x' randomly from $V_k(x)$;
    construct x'' from x' by applying a local search method
    IF f(x'')<f(x') THEN x:=x''; k:=1
                    ELSE k:=k+1
```

Related techniques – direct local search

- Nelder – Mead
  - It is based on successively applying some simple transformations on the elements of a set of n+1 candidates (a simplex):
    - Reflection
    - Expansion
    - Contraction
    - Reduction
- Hooke – Jeeves (pattern search)