Metaheuristic Algorithms

- What is this course about?
- Difficult optimization problems
- Metaheuristics taxonomy
- Main topics, organization and evaluation rules

What is this course about ?

- As almost all courses in Computer Science it is about problem solving
- Its main aim is to present techniques to solve hard problems
- There are problems which are hard:
  - both for humans and computers (e.g. large combinatorial optimization problems, multimodal / multiobjective optimization problems etc) – computationally hard problems
  - for computers but rather easy for humans (e.g. character recognition, face recognition, speech recognition etc) – ill posed problem

Computationally hard problems

- Problems characterized by a large space of solutions for which there are no exact methods of polynomial complexity (so-called NP hard problems) – they are characterized by a search space of large size (which cannot be exhaustively searched)

Examples:

- Satisfiability problem (SAT): find the values of boolean variables for which a logical formula is true. For n variables the search space has the size 2^n
- Travelling Salesman Problem (TSP): find a minimal cost tour which visits n towns. The search size is (n-1)! (in the symmetric case, it is (n-1)!/2)

Ill-posed problems

- The particularity of problems which are easy for humans but hard for computers is that they are ill-posed, i.e. there is difficult to construct an abstract model which reflects all particularities of the problem

- Let us consider the following two problems:

  – Classify the employees of a company in two categories: first category will contain all of those who have an income larger than the average salary per company and the second category will contain the other employees

  – Classify the employees of a company in two categories: first category will contain all those which are good candidates for a bank loan and the second category will contain the other employees

Ill-posed problems

- In the case of the first problem there is easy to construct a rule-based classifier:

  IF income > average THEN Class 1
  ELSE Class 2

- In the case of the second problem it is not so easy to construct a classifier because there are a lot of other interrelated elements (health status, family, career evolution etc) to be taken into account in order to decide if a given employee is reliable for a bank loan. A bank expert relies on his experience (previous success and failure cases) when he makes a decision

Well-posed vs Ill-posed Problems

- Differences between well-posed and ill-posed problems:

Well-posed problems:

- There is an abstract model which describes the problem
- Consequently, there is a solving method, i.e. an algorithm

Ill-posed problems:

- They cannot be easily formalized
- There are only some examples for which the results is known
- The data about the problem could be incomplete or inconsistent
- Thus, traditional methods cannot be applied

Ill-posed problems

The methods appropriate for ill-posed problems should be characterized by:

- Ability to extract models from examples (learning)
- Ability to deal with dynamic environments (adaptability)
- Ability to deal with noisy, incomplete or inconsistent data (robustness)
- Ability to provide the answer in a reasonable amount of time (efficiency)

Designing a system having all these characteristics usually leads to solving an optimization problem (= find the design variables which minimize an error, minimize a cost or maximize a quality criteria)

Optimization problems

Rough description: find one or several elements in the search space which optimize (minimize or maximize) one or several optimization criteria (called objective functions) and satisfy one or several constraints

Mathematical description (single objective optimization):

$$\text{Find } x^* \in D \subset R^n \text{ such that}$$

$$f(x^*) \leq f(x) \text{ for any } x \in D \; (f : D \to R \text{ is the objective function})$$

$$g_i(x^*) = 0, i = \overline{1, p} \quad \text{(equality constraints)}$$

$$h_j(x^*) \geq 0, j = \overline{1, q} \quad \text{(inequality constraints)}$$

Optimization challenges

Search space = space where the design variables take values

- Large size (many design variables)
- Characterized by complex constraints (the feasible region is not easy to reach)

Objective function

- „black-box" function (only the values of the function can be computed; no knowledge of its properties)
- many optima (multi-modal optimization)
- noisy function: different evaluations for the same argument may lead to different values (noisy or dynamic optimization)
- Multiple (conflictual) objective criteria (multi-objective optimization)

Remark:

- Traditional techniques (e.g. mathematical programming) are either inapplicable or inefficient

Classes of optimization problems

- Discrete search space → combinatorial optimization

Examples:

- Routing (e.g. vehicle routing)
- Planning (e.g. timetabling, task scheduling)
- Allocation (e.g. resource allocation)
- Selection (e.g. feature selection)

- Continuous search space → continuous optimization

Examples:

- Parameter estimation
- Finding minimal energy configurations
- Training adaptive systems

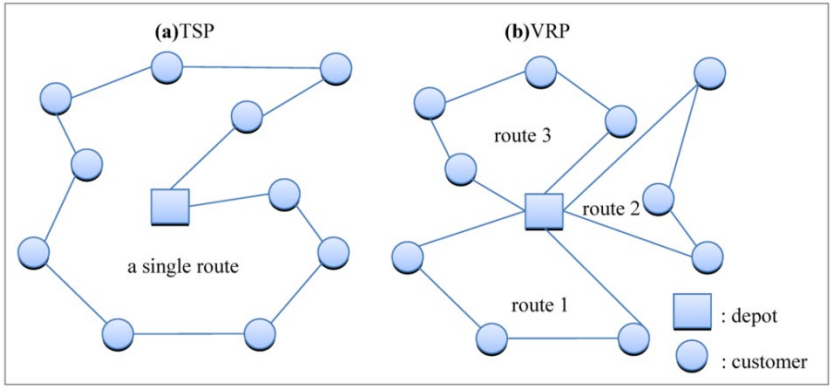
Routing

Vehicle Routing Problem:

- Find a minimal cost route which visit a set of locations and satisfies some constraints

Particular case: travelling salesman problem

TSP= find a Hamiltonian circuit of a minimal cost in a complete graph



Solution representation:

1. Binary allocation matrix (nxn):

A_{ij} = 1 if node j is visited at step i
= 0 otherwise

Constraints:

- Each row contains exactly one 1
- Each column contains exactly one 1

Search space: set of binary matrices of size nxn

Search space size: 2^{n*n}

Objective function (to be minimized):

$$f(A) = \sum_{i=1}^{n} \sum_{j \neq k} c_{jk} A_{ij} A_{i+1,k}, \ (n+1 \equiv 1)$$

c_{jk} = cost of visiting the edge (j,k)

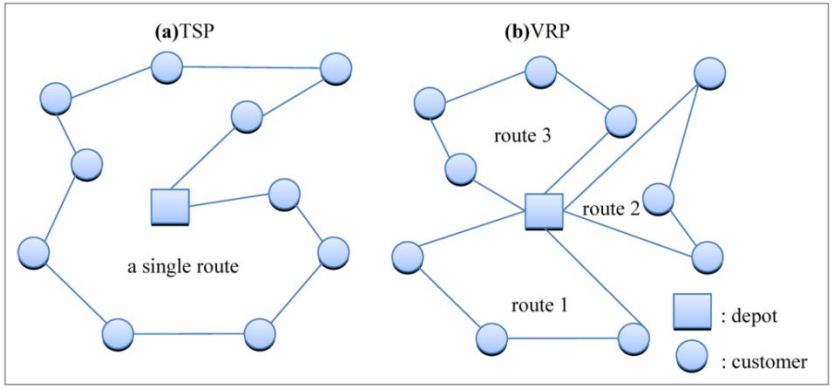
Routing

Vehicle Routing Problem:

- Find a minimal cost route which visit a set of locations and satisfies some constraints

Particular case: travelling salesman problem

TSP= find a Hamiltonian circuit of a minimal cost in a complete graph



Solution representation:

2. Permutation:

p_i = index of the node which is visited at step i

Constraints:

- The elements of p are distinct

Search space: the set of permutations of order n (order n-1 for symmetric problems)

Search space size: n! (for symmetric problems (n-1)!/2)

Objective function (to be minimized):

$$f(p) = \sum_{i=1}^{n} c_{p_i p_{i+1}} \ , \ (n+1 \equiv 1)$$

Planning

Problem: Let us consider

- a set of events (e.g. lectures, exams etc.),
- a set of rooms
- a set of time slots

Construct a schedule such that each event is assigned to a room and a time slot such that different types of constraints are satisfied.

The constraints can be:

- hard (mandatory)
- soft (can be violated)

rooms

Time slots

	R1	R2	R3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

Search space: the set of functions which associate to each pair (time slot, room) an event (or the empty activity)

Search space size (no constraints): $(k+1)^{mn}$

k events

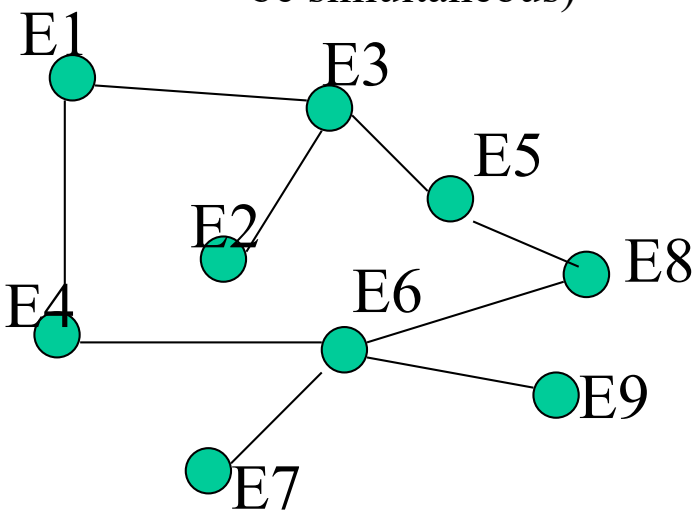
m time slots

n rooms

Planning

- Hard constraints (the solution is feasible only if they are satisfied):
  - Each event is scheduled exactly once
  - Only one event is scheduled in a room at a given time moment
  - The room satisfies the requirements of the event
  - There are no simultaneous events at which same persons should attend

Conflict graph (two connected nodes correspond to events which cannot be simultaneous)



	R1	R2	R3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

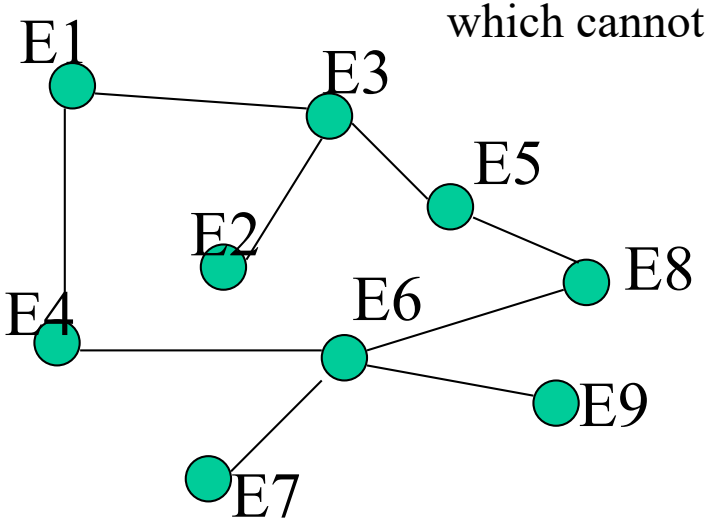
Planning

Soft constraints:

- There are no more than k successive events which should be attended by the same participant
- There are no participants for which there is only one event/day

Idea: the soft constraints can be transformed in optimization criteria (e.g. the number of participants for which some of the constraints are violated is as small as possible)

Conflict graph (two connected nodes correspond to events which cannot be simultaneous)



	R1	R2	R3
T1	E1	E3	E9
T2	E4		E8
T3	E6	E5	
T4	E2		E7

Resource allocation

Problem: cloud resource allocation

Let us consider:

- A set of tasks characterized by some requirements
- A set of virtual machines (VMs) having some characteristics

The aim is to assign the tasks to VMs such that:

- The tasks requirements are satisfied
- The number of used virtual machines (or the global cost or the consumed energy) is as small as possible

Particular case: bin packing

Solution representation:

(n tasks, m VMs)

Allocation: $V=(v_1,v_2,\ldots, v_n)$

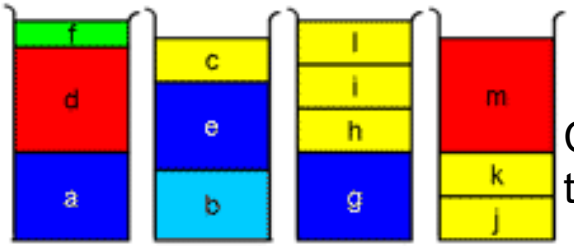
v_i = index of the VM on which is placed the task i

Search space: the set of functions defined on {1,2,...,n} and taking values in {1,2,...,m}

Search space size: m^n

Objective function:

$$f(V)=\sum_{i=1}^{n}\text{cost}(i,V_i)$$



Cost for assignment of task i on v_i

Selection

Feature/attribute selection

Let us consider:

- a dataset characterized by a large number of attributes

We are looking for a subset of features:

- which maximizes the classification accuracy
- minimizes the cost of data processing (and/or the size of the model extracted from data)

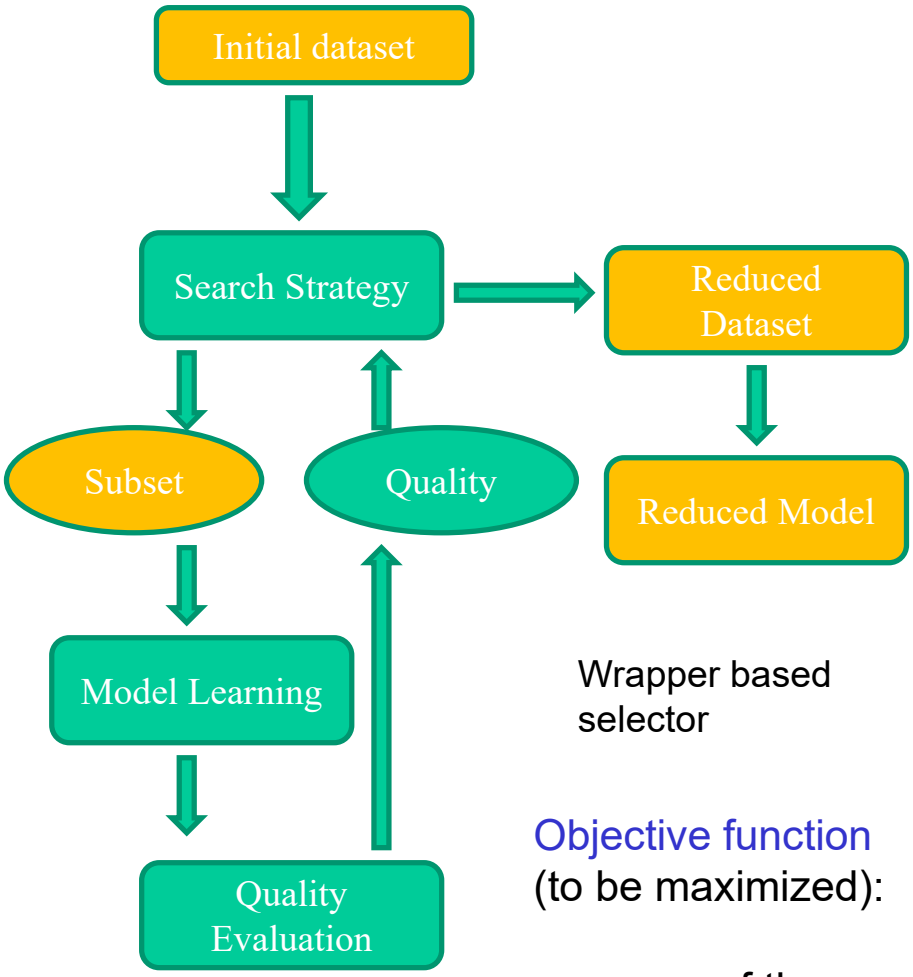
Solution representation: binary vector

S_i=1 if attribute i is selected

=0 if attribute i is not selected

Search space: the set of binary vectors with n elements

Search space size: 2^n (n=initial number of attributes)



Wrapper based selector

Objective function (to be maximized):

accuracy of the classification model

Parameter estimation

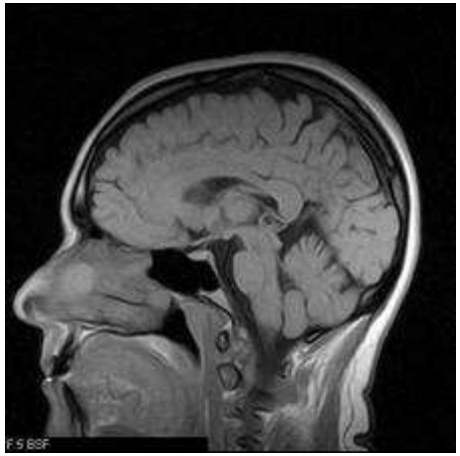
Image registration

Let us consider two images of the same object which should be „aligned" = we search for an image transformation which maximizes the similarities between the corresponding pixels of transformed images

Solution representation:

Vector of real values corresponding to the parameters of the transformation (e.g. rotation, translation etc.)

Search space:
$[a_1,b_1]x[a_2,b_2]x...x[a_n,b_n]$



I_1

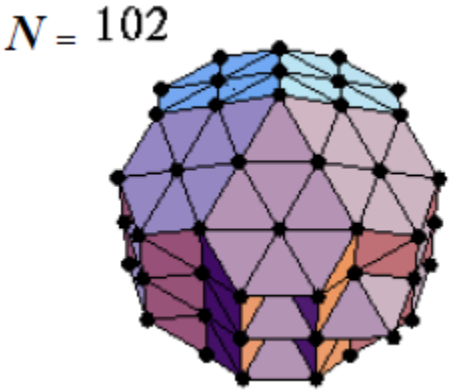
I_2

Objective function (to be minimized wrt the set of parameters, p)

$Dist(I_1,T(p;I_2))$ = distance between a reference image (I_1) and the transformed image (using the set of parameters p)

Parameter estimation

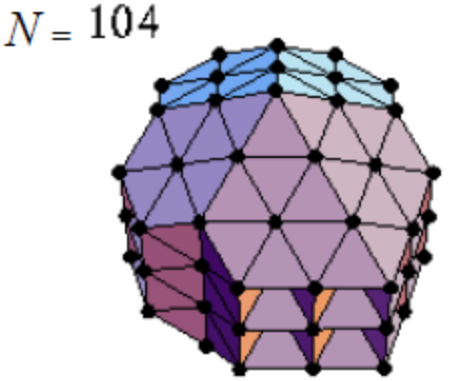
Problem: Find the geometric coordinates of a set of atoms which minimizes the internal energy of the system (there are different forms of the internal energy which are determined by the used potential function, e.g. Lennard-Jones)



Solution representation: vector of real values containing triplets of atoms coordinates

Search space: $[a_1,b_1]x[a_2,b_2]x...x[a_n,b_n]$

(for instance: [0,4]x[0,4]x[-4,4]x...x[-4,4])



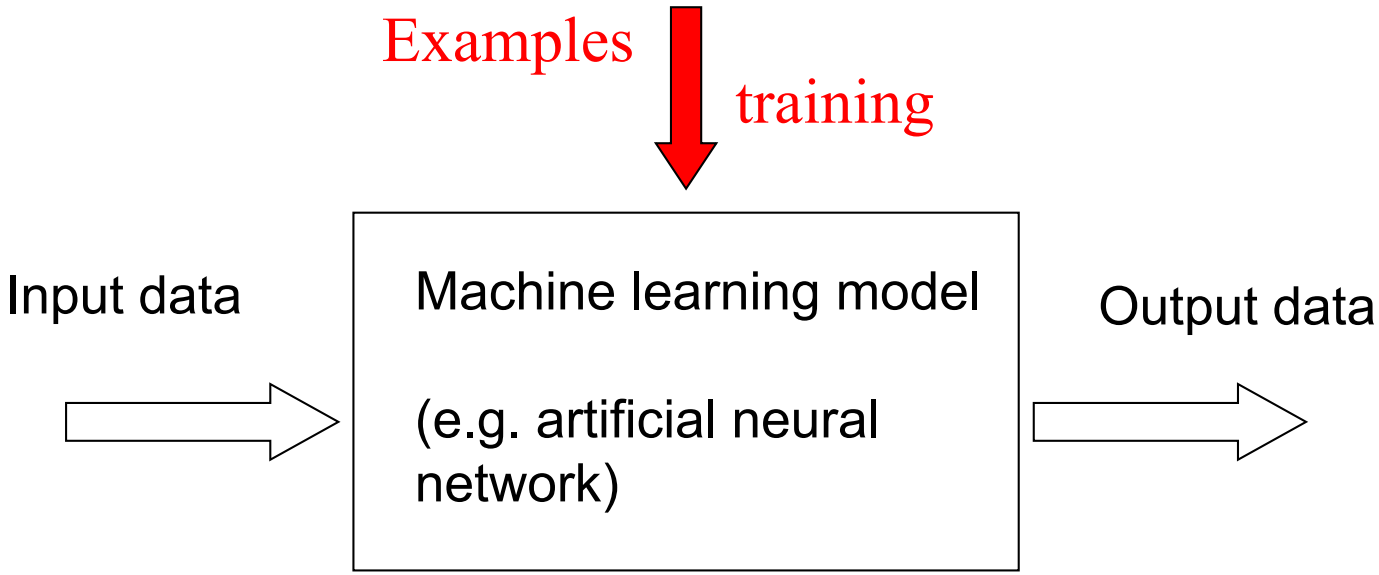
Challenge: it is difficult to find the global optimum (particularly for large values of the number of atoms)

$$f(P_1,P_2,...,P_{3n}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\frac{\sigma^{12}}{dist(p_i,p_j)^{12}} - \frac{\sigma^{6}}{dist(p_i,p_j)^{6}}\right),$$

$$p_i = (P_{3(i-1)+1}, P_{3(i-1)+2}, P_{3(i-1)+3})$$

Training adaptive systems

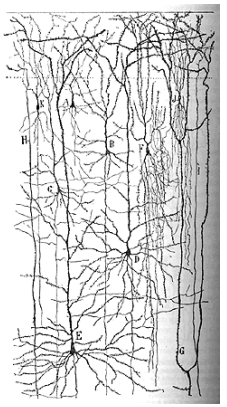
Machine learning approach



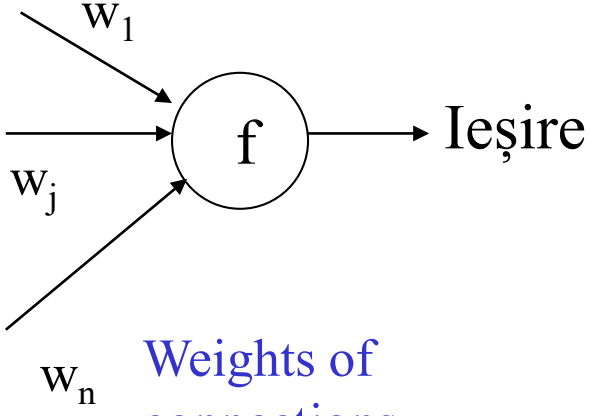
Training = estimate the adaptive parameters of the model such that an error/ loss function is minimized or a quality criterion is maximized

Artificial neural networks

ANN= set of interconnected artificial neurons (functional units characterized by adaptive parameters – the weights)



inputs



Weights of connections

$$y = f\left(\sum_{j=1}^{n} w_j x_j - w_0\right)$$

Aggregation function

$$u = \sum_{j=1}^{n} w_j x_j - w_0$$

$$f(u) = \mathrm{sgn}(u) \qquad f(u) = \tanh(u)$$

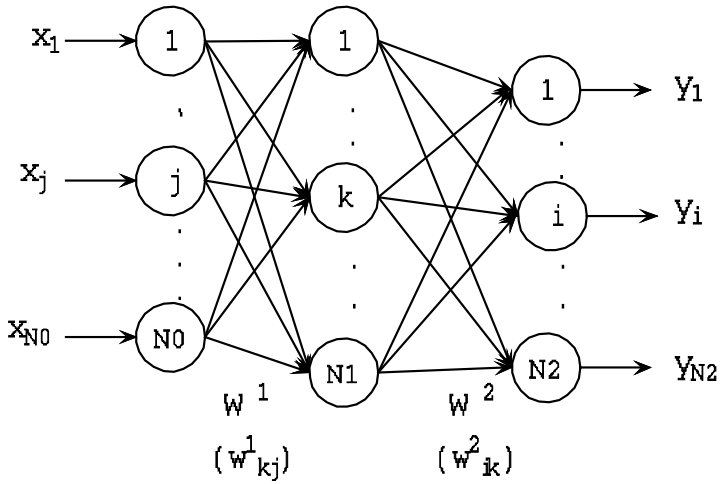
$$f(u) = H(u) \qquad f(u) = \frac{1}{1+\exp(-u)}$$

Activation function (computation of the output signal)

ANN training

Structure of an artificial neural network:

- Architecture
- Functioning
- Supervised training: find the weights which minimizes an error function (difference the expected answer and that produced by the network)



Search space: the space of the weights W (vectors with (N0+1)x(N1+1) components)

$$y_i = f\left(\sum_{k=0}^{N1} w_{ik}^2 f\left(\sum_{j=0}^{N0} w_{kj}^1 x_j\right)\right),\ i = \overline{1, N2}$$

Feedforward neural network (used for classification/association problems)

Objective function:

$$f(W) = \sum_{l=1}^{L}\sum_{i=1}^{N_2}(d_i^l - y_i^l(W))^2$$

$d_1^l,...,d_{N_2}^l$ – correct answer for example l

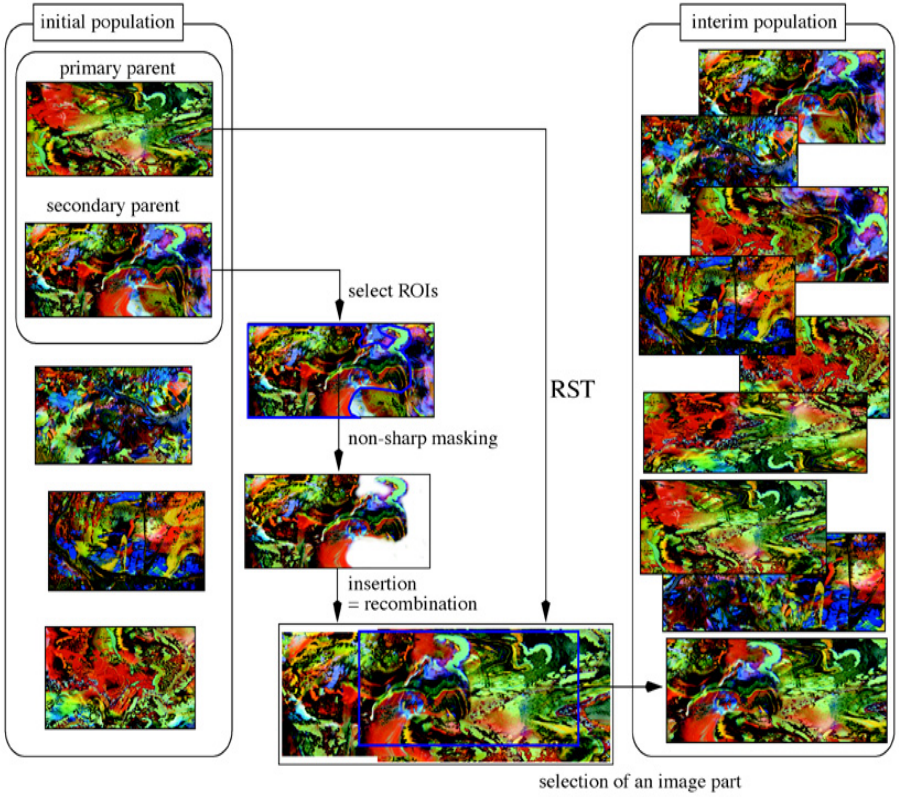
$y_1^l(W),...,y_{N_2}^l(W)$ – network answer for example l

Interactive optimization

- Design of various artistic structures (images, sounds) which optimizes a subjective criterion provided by a user

- The objective function is evaluated by the user (e.g. a score on artistic quality of the structure)

- DarwinTunes http://darwintunes.org/



[http://www.evogenio.com/de/GBEvoArt/EvoArt1.html]

Metaheuristic Algorithms

"A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions"

[I.H. Osman, G. Laporte, Metaheuristics: a bibliography, Annals of Operations Research 63 (1996) 513–623]

"A top-level general strategy which guides other heuristics to search for feasible solutions in domains where the task is hard."

[dictionary.reference.com]

Terminology

- Heuristic (from greek: heuriskein = to find, to discover)
- Meta (from greek: beyond in the sense of high-level)

Metaheuristic Algorithms

Key ideas:

- Metaheuristics rely on iterative improvements of candidate solutions by using in a clever way heuristics which are specific for the problem to be solved

- The main aim is to ensure a balance between
  - exploration of the search space
  - exploitation of the information gathered during the previous search

- They are rather general techniques (same algorithm can be applied with minor changes to different problems)

- Most of them rely on the usage of stochastic operators (involve randomness to compensate for the lack of knowledge on the problem properties)

Metaheuristic Algorithms

Advantages:

- They require only the values of the objective function (which does not have to be smooth or continuous, i.e. black box optimization)
- They can estimate the global optimum (if the search space is properly explored)

Disadvantages:

- There are few theoretical results concerning the convergence and the quality of the approximation (they are not as sound as mathematical programming methods)
- Their design is mainly based on intuition and analogy with other processes (in many cases observed in nature or in social interactions) which ensure the improvement in some sense of the behavior of a system

Metaheuristics Taxomomy

W.r.t. search scheme:

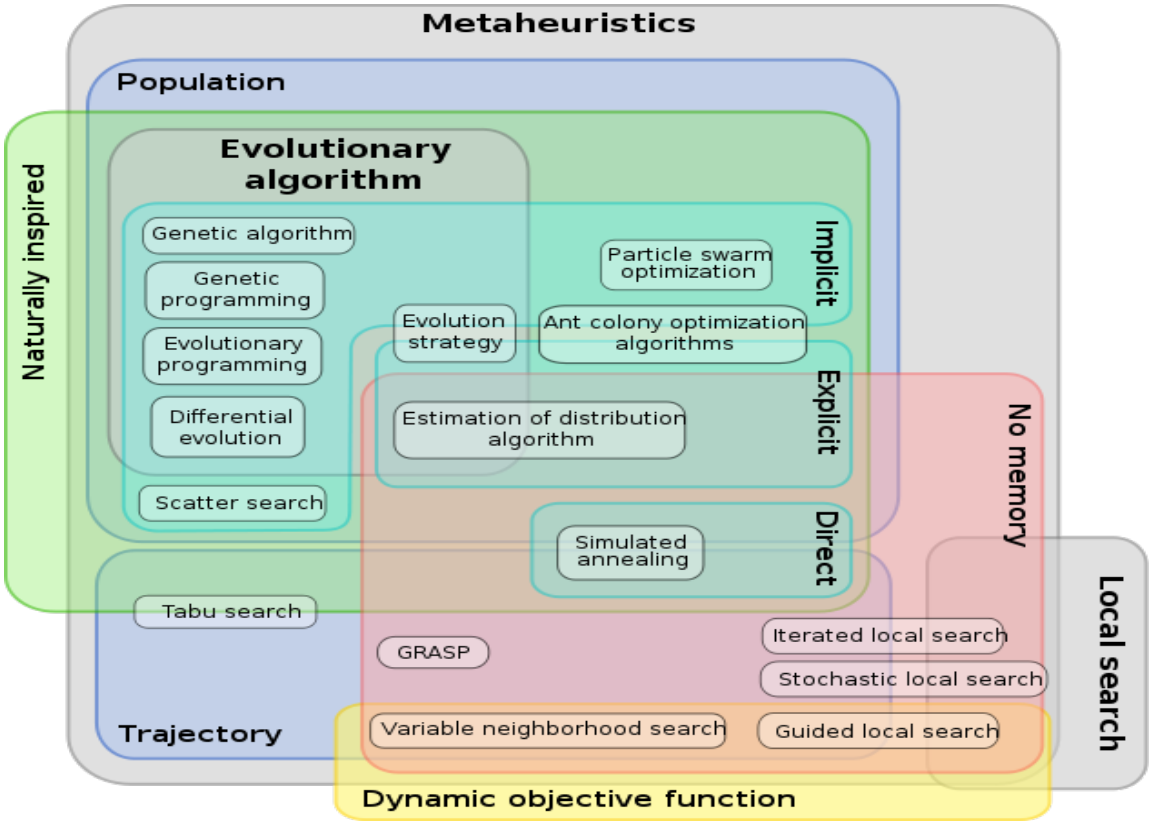
- Trajectory based (one candidate solution)
- Population based (several candidate solutions)

W.r.t. search area:

- Local
- Global

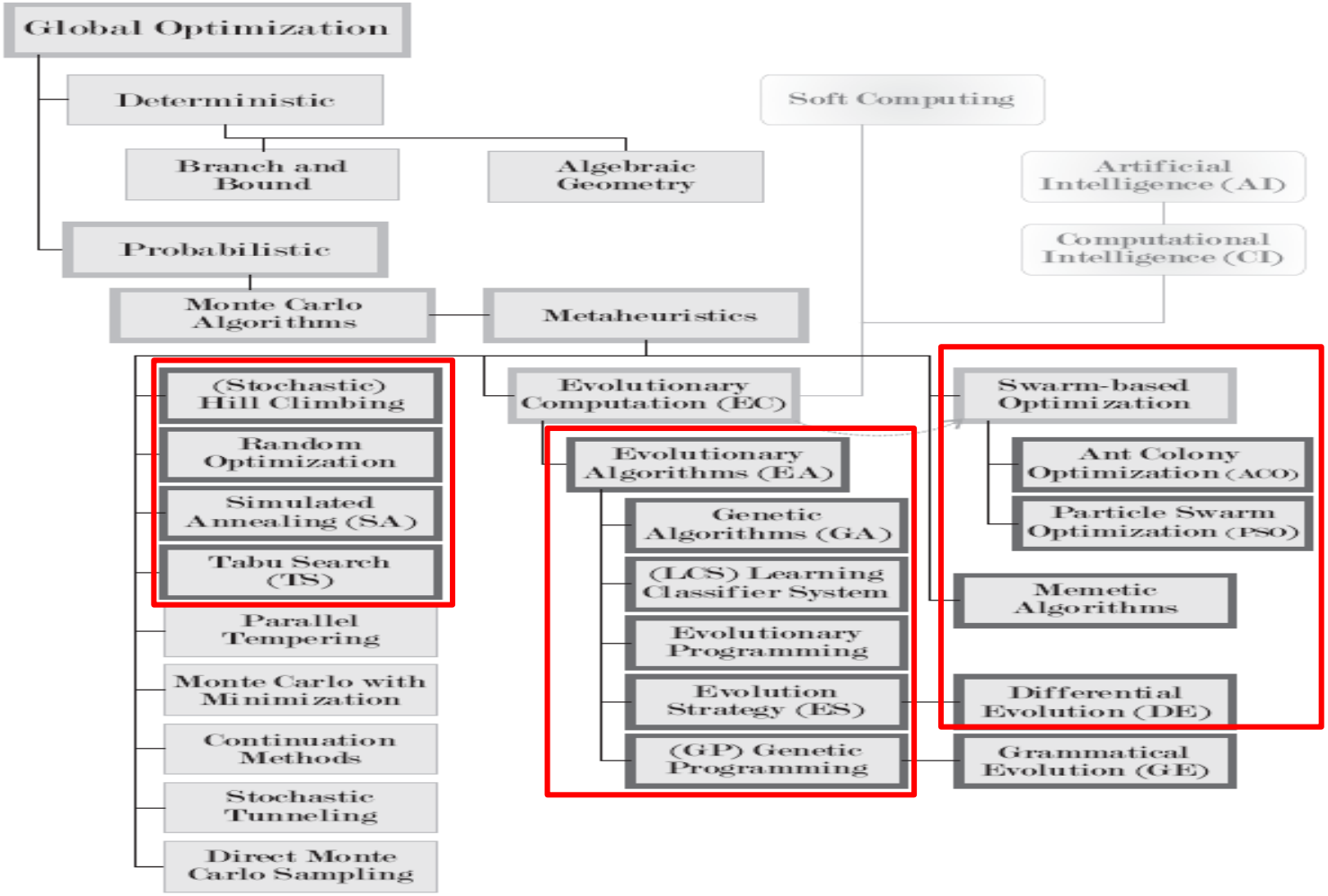
W.r.t. underlying metaphor:

- Biological evolution (e.g. evolutionary algorithms)
- Social interaction (e.g. swarm intelligence)
- Physics (e.g. simulated annealing)
- etc.



Metaheuristics Taxonomy [Wikipedia, nojhan.free.fr]

Metaheuristics Taxomomy



Structure of a generic trajectory based metaheuristic

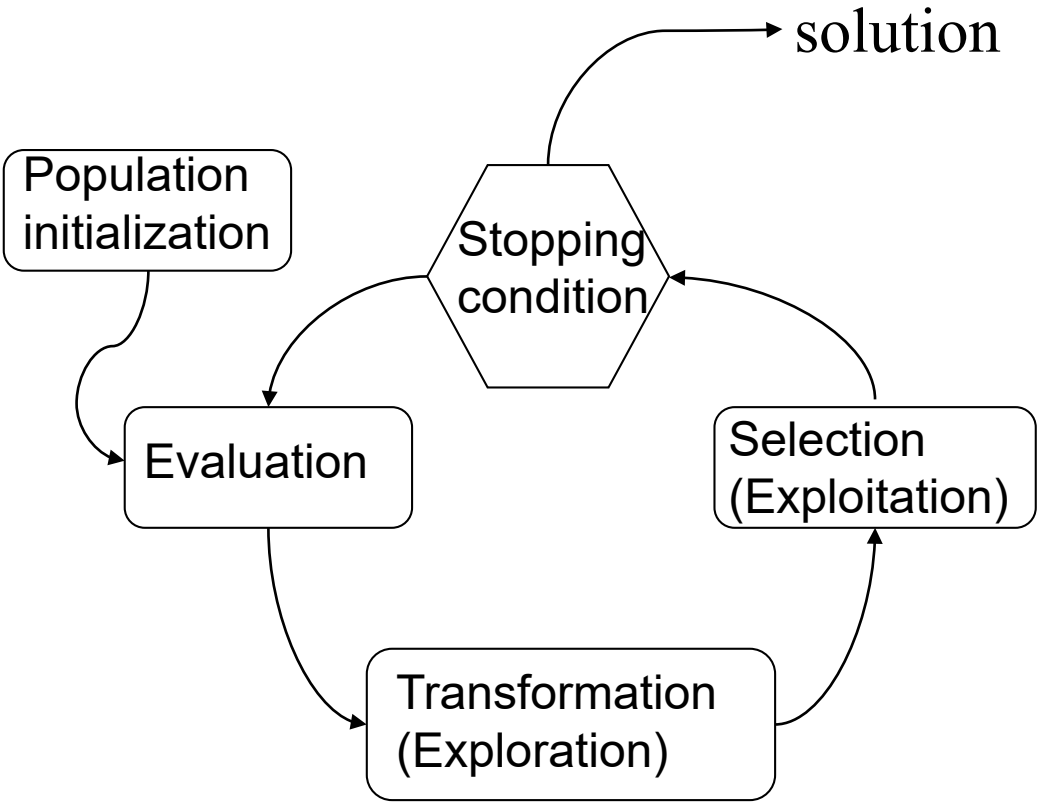
```
S=initial candidate solution
Repeat
    R=modify(S)
    if quality(R)>quality(S) then R=S
Until <stopping condition>
```

Components:

- Initialization
- Modification (ensures the exploration of the solution space)
- Selection (ensures the exploitation of good quality solutions)

Key element: ensure the balance between exploration and exploitation

Structure of a generic population based metaheuristic




```
P=initial population
Repeat
     evaluation of P
     P'= transformation(P)
     P = selection (P')
Until <stopping condition>
```

Population based metaheuristics= Iterative process based on successive application of

- exploration
- exploitation

operators on a randomly initialized population

Topics

- Metaheuristics which use a candidate solution (trajectory-based) – Lectures 2-3
  - Pattern Search (Hookes Jeeves)
  - Nelder Mead
  - Random search (e.g Matyas, Solis Wets)
  - Restarted Local Search
  - Iterated Local Search
  - Simulated Annealing
  - Tabu Search
  - Scatter Search
  - Variable Neighborhood Search
  - Greedy Randomized Adaptive Search

Topics

- Metaheuristics which use a population of solutions (population-based)
  - Evolutionary algorithms: (lectures 4-6)
    - ES – evolution strategies,
    - EP- evolutionary programming
    - GA – genetic algorithms
    - GP – genetic programming
  - Swarm intelligence algorithms: (lecture 7)
    - PSO - Particle Swarm Optimization,
    - ACO - Ant Colony Optimization
    - ABC - Artificial Bee Colony etc
  - Differential Evolution (lecture 8)
  - Algorithms based on estimation of the probability distribution (lecture 8)
    - PBIL – Population Based Incremental Learning
    - EDA – Estimation of Distribution Algorithms

Course structure

- Scalable metaheuristic algorithms (Lecture 9)
  - Cooperative coevolution
  - Parallelization models:
    - Master-slave
    - Island model
    - Cellular model
- Algorithms for some classes of optimization problems (Lectures 10-12)
  - Constrained optimization
  - Multimodal optimization
  - Multi-objective optimization
  - Dynamic optimization
- Applications (Lectures 13-14)
  - Evolutionary design of neural networks
  - Data mining: feature selection, rules mining, clustering
  - Planning: routing, scheduling and resource allocation

References

1. Sean Luke: *Essentials of Metaheuristics,* Lulu, second edition, 2013, available for free at http://cs.gmu.edu/~sean/book/metaheuristics/
2. Z. Michalewicz, D. Fogel: *How to Solve It. Modern Heuristics*. Springer, 1999
3. Jason Brownlee: *Clever Algorithms. Nature-inspired Programming Recipes*, 2011, available at http://www.CleverAlgorithms.com

Lab

Lab 1: Traditional optimization methods (gradient based methods, linear programming)

Lab 2: Implementation of trajectory based methods. Discrete optimization

Lab 3: Evolutionary Algorithms. Continuous optimization

Lab 4: Swarm intelligence algorithms (PSO, ACO)

Lab 5: Multi-objective optimization - MOEA

Lab 6: Evolutionary design of neural networks

Lab 7: Data mining and planning applications

Tools: (mainly) Python

Evaluation

Course materials: http://staff.fmi.uvt.ro/~daniela.zaharie/ma2019

Evaluation:

Final project: 60-80%

Written test (open book): 20%

Lab activity: 20%