Neural and Evolutionary Computing.

Lab 7: Multiobjective optimization. Ant Colony Optimization and Particle Swarm Optimization.

---

1. Multiobjective optimization

Multiobjective optimization means to simultaneously optimize several objective functions (criteria). The function to be optimized is vectorial $F:R^n->R^r$, and its components can be denoted as follows $F=(f_1,f_2,\ldots,f_r)$.
The optimization criteria are usually conflicting, therefore the problem does not have a unique solution. In such a case we are looking for some compromise solutions (called Pareto optimal) characterized by the fact that they cannot be improved with respect to all their components (any improvement with respect to one criterion leads to a decrease of quality with respect to other criteria).
There are different approaches of this problem. The main approaches are:

- *Aggregation methods:* the multiobjective problem is transformed in a one-objective optimization problem by combining all optimization criteria in a single one. Thus the new objective function becomes: $f(x)=w_1f_1(x)+w_2f_2(x)+\ldots+w_rf_r(x)$ where $w_1,w_2,\ldots,w_r$ are weights associated to objective functions. For each set of weights one can obtain a different solution.
- *Direct approximation of the Pareto optimal set:* it uses a population of elements which will approximate the Pareto optimal set. The approximation process can be a evolutionary one. The main difference between multiobjective EAs and single objective EAs is related with the selection process. In the MOEAs the selection process is based on the dominance relationship between the elements (see Lecture 11).

Examples of test functions used to evaluate the performance of multiobjective algorithms are available at: http://en.wikipedia.org/wiki/Test_functions_for_optimization

Application 1. Let us consider the function $F:[0,4]->RxR$, $F(x)=((x-1)^2,(x-2)^2)$. Estimate the optimal Pareto set and the corresponding Pareto front.

Variant 1. By using the aggregation technique

a) Construct the aggregated objective function:

```
function y=fw(x)
    w=0.1;
    y1=(x-1)*(x-1);
    y2=(x-2)*(x-2);
    y=w*y1+(1-w)*y2;
endfunction
```

b) Apply an evolution strategy (for instance, that described in SE.sci) to optimize the aggregated objective for the following values of w: (0.1,0.2,0.3,…,0.9). The corresponding results are collected in a list.

c) Plot the points having as coordinates the values of the objective functions computed at the previous step (the plotted set of points will be illustrate an approximation of the Pareto front):

```
function pareto(x)
    f1=(x-1).^2;
    f2=(x-2).^2;
    plot(f1,f2,'*');
endfunction
```

Variant 2. Use the algorithm NSGA-II and MOGA algorithms implemented in SciLab (functions optim_nsga2 and optim_moga).

Hint: exNSGA.sci

2. Ant Colony Optimization (ACO)

ACO is a metaheuristic inspired by the behavior of the ant colonies. It is especially used in solving combinatorial optimization problems (e.g. routing, scheduling, assignment) It uses a population of artificial ants (agents) which is changed during an iterative process. At each iteration each ant constructs a potential solution. The values for the solution components are chosen randomly based on a probability distribution. The probability distribution is computed by using both local information (what the ant can collect from its neighbourhood) and global information (obtained by using the indirect communication process between ants based on pheromone trails).

Solving TSP using ACO. The input data consists of the graph describing the direct connections between towns and their costs. A population of ants is initially placed on random nodes (or all of them in the first node). At each iteration, each ant visits *n* distinct nodes, constructing a tour. The ants have a local memory where the list of visited nodes is stored in order to avoid visiting twice the same node. The transition of an ant *k* from the node *i* into the node *j* at step *t* is based on the following probability:

$$P_t^k(i,j) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \in N(k)} \tau_{il}^{\alpha}\eta_{il}^{\beta}} & j \text{ was not visited} \\ 0 & j \text{ was visited} \end{cases}$$

The factors appearing in the computation of the probability are:

- τ_{ij} models the pheromone concentration released by the ants on edge *(i,j);* the pheromone concentration is randomly initialized with small positive values. Each ant which visits an edge *(i,j)* can release some pheromone on it contributing to the update of the pheromone concentration:

  $$\tau_{ij} = (1-\rho)\tau_{ij} + \rho \sum_k Q_{ij}(k) / \text{cost}(T_k)$$

  ρ is a constant less than *1* which controls the evaporation process, $Q_{ij}(k)$ is *0* if *(i,j)* does not belong to the tour constructed by ant *k*. $Cost(T_k)$ denotes the cost of the tour constructed by the ant *k*.
- η_{ij} models the local information concerning the quality of the edge; the simplest variant is when it is *1/cost(i,j).*

- α and β are parameters which control the relative importance of those two types of information: the global information provided by the pheromone concentration and the local one provided by the cost of the edge.

Application 2. Implement an ACO algorithm for TSP.

Hint. See function ACO_TSP.sci

Exercise 1. Change the previous implementation such that when the pheromone matrix elements are updated the tours visited by all ants are taken into account.

Hint. The updating terms are cumulated after each tour construction.

3. Particle Swarm Optimization (PSO)

PSO is a metaheuristic used for continuous function optimization inspired by the behavior of bird swarms. It uses a population of *m* "particles", each particle *i* being characterized by its position (x_i) and its velocity (v_i). Moreover, each particle memorizes the best position it visited up to the current moment ($xbest_i$). There is also another variable which contains the best position found up to the current iteration by the entire swarm (*xbest*). The evolutionary process consists in the change, at each generation *t*, of the position of all particles in the population according to the following rules:

$$v_i(t+1) = \gamma(v_i(t) + r_1 rand(0,1)(xbest_i - x_i(t)) + r_2 rand(0,1)(xbest - x_i(t)))$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

where:

- gamma is a constriction factor (a typical value for gamma is 0.7)
- r_1 and r_2 are two constant values (e.g. r1=r2=2.05)

Besides this variant, where *xbest* is the global best element from the swarm, there is also another variant where for each particle *i*, *xbest(i)* is selected as the best element from the neighborhood of the particle *i*. The neighborhood of a particle can be defined by various topologies, one of the most used is the ring topology (in this case the neighborhood of size K of particle i is represented by the particles having the indices *{i-K,i-K+1,...,i-1,i,i+1,...,i+K-1,i+K}*).

Application 3. Implement a PSO algorithm (using the above eqs.) and test its behavior for a unimodal function (e.g. sphere) and for a multimodal function (e.g. Griewank).

Hint. See function PSO.m

Exercise 2. Change PSO.m such that it implements the "local best" variant using a ring topology to define the neighbourhood.