Metaheuristic algorithms.

Lab 7:
Dynamic optimization
Evolutionary training of neural networks

---

1. Dynamic optimization

The dynamic optimization problems are characterized by changes in the problem specification (e.g. size, constraints, objective function etc). One of the most popular example of dynamic optimization problem is the Dynamic Traveling Salesman Problem (DTSP) which is a generalization of the classic TSP where changes consist of:

- adding new nodes or deleting existing nodes
- changing the location of some nodes (i.e. replacing some nodes with other ones)
- changing the costs of the connection between nodes.

When a change occurs, the salesman has to re-plan the route and the main idea is to avoid starting the optimization search from the beginning. Instead the search strategy should exploit as much possible the already constructed solutions. In order to adapt for dynamic optimization an algorithm which has been initially designed for a static optimization one have to take into account the characteristics of the changes:

- how frequent are the changes?
- how large are the changes?
- are the changes periodic or can they be predicted?

The strategies used in dynamic optimization are:

- avoid convergence by stimulating the population diversity through:
  - fitness sharing
  - crowding based selection
  - random immigrants

  Remark: this is useful in small but frequent changes in the objective function
- store some information related to solutions corresponding to older variants of the problem and re-use them (by including them in the population) when a change occurs in the problem

  Remark: such an approach might be useful in case of periodic changes; in this case it is important to detect the moment when the change occurs

Application 1. Modify the ACO_TSP algorithm (lab 5) in order to incorporate changes which switch the status of some nodes (an active node becomes inactive and the reverse case).

Hint: store the activity status of all nodes (e.g. in a table); at each iteration this activity status is used to:

- ignore the inactive nodes when a tour is constructed
- keep "frozen" the pheromone values associated to edges containing at least one inactive node

2. Evolutionary training of neural networks

Evolutionary algorithms (but also other population-based metaheuristics) can be used for neural networks design at least in two cases:

- Estimation of the synaptic weights in the case of networks with non-differentiable activation/transfer functions or in the case of recurrent networks (when traditional algorithms like Backpropagation cannot be applied)
- Establishing the architecture of the network (both in the case of explicit and in the case of implicit encoding – see Lecture 13) .

Application 2. Let us consider the problem of training a neural network in order to represent the XOR function.

We can use the following architecture:

- 2 input units + 1 dummy unit (used to provide the biases for the hidden units)
- K hidden units (K is an input parameter) with a Heaviside activation function + 1 dummy unit (used to provide the bias for the output unit)
- An output unit with a Heaviside activation function

Each element of the population will have 4*K+1 components corresponding to all weights and biases. The function to be minimized is the MSE (Mean Squared Error) computed for the training set containing all four examples: ((0,0),0), ((0,1),1),((1,0),1), ((1,1),0). Analyze the influence of the population sizes, mutation parameter, number of parents used for recombination, selection type (truncation or tournament) and value of K.

Hint. see SE_nn.sci. The function used by the evolution strategy to evaluate a network is SE_XOR.

Exercise: Use DE (differential evolution) or PSO (particle swarm optimization) instead of the evolution strategy to estimate the parameters of the neural network.

Homework. Implement an evolutionary approach which estimate simultaneously both the number of hidden units and the corresponding weights.