Metaheuristic Algorithms.

Lab 3: Implementation of Evolutionary Algorithms

- **Combinatorial optimization (Genetic Algorithms - GAs)**

---

1. Implementation of GAs

Let us consider the case when the elements of the population are binary strings. Problems for which such an encoding is appropriate are optimization problems involving Boolean functions (e.g. ONEMAX, satisfiability problems) or selection problems (e.g. knapsack problem).

In order to implement a GA there are several components (implemented as separate modules or functions) which should be designed:

- *Population initialization*: usually the population elements are randomly selected from the search space (in the case of binary encoding the search space is $\{0,1\}^n$).
- *Population evaluation*: computes the fitness of each element in the population; the way the fitness is computed depends highly on the problem to be solved. Particular attention should be given to constrained optimization problems (in such a case, the constraints can be included in the objective function).
- *Selection of parents/survivors*: use a stochastic selection procedure; the typical variants are:
    - *Proportional selection:* the selection probability is proportional with the fitness value; the indices of the selected elements are generated using the method of inverting the probability distribution function (e.g. roulette or stochastic universal sampler)
    - *Rank based selection*: the selection probability is proportional with the rank of the element in the increasingly sorted (by fitness – if the fitness should be maximized) population;
    - *Tournament selection:* several elements are randomly chosen (with or without replacement) from the population and the best one of them is selected; in practice samples of small size are used (in most cases the sample consists of 2 elements);
    - *Uniform selection:* the elements are randomly selected, independent of their fitness values (it can be used only for the selection of parents).
- *Crossover:* starting from two parents constructs two offspring (children):
    - *One cut-point:* for parents $x=(x_1,x_2,\ldots, x_n)$ and $y=(y_1,y_2,\ldots, y_n)$ and a cut point k, the children will be $c1=(x_1,x_2,\ldots,x_k,y_{k+1},\ldots y_n)$ and $c2=(y_1,y_2,\ldots,y_k,x_{k+1},\ldots x_n)$
    - *Two cut-points:* for parents $x=(x_1,x_2,\ldots, x_n)$ and $y=(y_1,y_2,\ldots, y_n)$ and the cut points k1 and k2, the children will be $c1=(x_1,x_2,\ldots,x_{k1},y_{k1+1},\ldots y_{k2}, x_{k1+1,\ldots} ,x_n)$ and $c2=(y_1,y_2,\ldots,y_{k1},x_{k1+1},\ldots x_{k2}, y_{k1+1,\ldots} ,y_n)$.
    - *Uniform:* the components of the offspring are taken randomly from the parents, i.e. the parent which provides the value of each component of the first child is selected randomly (usually with probability 0.5); the value of the component in the second child will be taken from the other parent.

- *Mutation:*
    - *Chromosome-based*: for each chromosome it is decided (based on the mutation probability) if it should be mutated or not; if the chromosome should be mutated choose a random component and mutate it (by complementing its value).
    - *Gene pool-based:* scan all genes (all components of all elements in the population) and decide based on the mutation probability which genes should be mutated.

Application 1. ONEMAX problem

Find the binary sequence ($x_1, x_2,\dots, x_n$) which maximizes $x_1+ x_2+\dots+ x_n$.

Hint: see function onemaxGA.sci

Exercise: Analyze the influence of the population size and of the mutation probability on the quality of the estimated solution.

Application 2. Knapsack problem

Find the binary sequence ($x_1, x_2,\dots, x_n$) which:

- Satisfies the constraint: $w_1*x_1+w_2*x_2+\dots+ w_n*x_n<=C$ (the total weight of the selected objects is smaller than the knapsack capacity)
- Maximizes the function: $v_1*x_1+v_2*x_2+\dots+ v_n*x_n$ (the total value of the selected objects is maximized)

Hint: The fitness value is constructed using the penalty technique:

$$\begin{aligned}\text{Fitness}(x_1, x_2\dots, x_n)=&(1-\text{alpha})*(v_1*x_1+v_2*x_2+\dots+ v_n*x_n)\\ &+\text{alpha}*H(C - w_1*x_1+w_2*x_2+\dots+ w_n*x_n)\end{aligned}$$

where $H(u)=0$ if $u>=0$ and $H(u)=u$ if $u<0$; alpha is a value from (0,1) and is used to control the relative importance of the constraint satisfaction with respect to the maximization of the objective function

An example is implemented in knapsack.sci

Exercise:

1. Analyze the influence of the parameter alpha on the quality of the solution.
2. Replace the proportional selection with a tournament selection and use the following criterion to decide that a candidate solution *x* is better than a candidate solution *x'*:
    a. Both x and x' satisfy the constraint but the value of x is higher than the value of x'
    b. x satisfies the constraint and x' does not satisfy the constraint
    c. Neither x nor x' satisfy the constraint but the weight corresponding to x is smaller than the weight corresponding to x'

Application 3. Traveling Salesman Problem (TSP)

The particularities of a GA designed to solve TSP are:

- *Solution encoding.* The natural encoding variant for TSP is the permutation (any tour over n nodes is a permutation of order n specifying the order in which the nodes should be visited).
- *Parent selection.* Any selection mechanism can be used. For instance:
  - Proportional selection
  - Tournament selection

- *Crossover operator.* A simple crossover variant for permutation-like configurations is obtained by extending the one-cut point crossover: for two parents two offsprings are constructed, as follows:
  - Choose a random cut point (k)
  - Transfer the first k components from the first parent to the first offspring; the other components of the first parent are transferred to the first offspring in the order given by the second parent.
  - The second offspring is constructed starting from the second parent in a similar way

Example: Let P1=(E,A,C,D,B,F), P2=(D,C,F,A,E,B) and k=3. The children will be C1=(E,A,C,D,F,B) and C2=(D,C,F,E,A,B).

- *Mutation operator (constructing a new configuration starting from the current one).* Several mutations can be used for a permutation-like encoding:
  - *2*-opt (the variant used in the Simulated Annealing variant):
    - (E,A,C,D,B,F) → (E,D,C,A,B,F)
  - Exchange of two randomly selected elements:
    - (E,A,C,D,B,F) → (E,A,F,D,B,C)
  - Transfer of a randomly selected element on a randomly selected position:
    - (E,A,C,D,B,F) → (E,C,D,B,A,F)

Remark: For TSP, the mutation operator is considered to be more effective than the crossover.

- *Survivors selection.* From the joined population of parents and offspring the elements which survive for the next generation can be selected by:
  - tournament: randomly select two elements from the joined population and select the best one
  - from the set of 2 parents and 2 offspring choose the best 2 elements
  - choose the best m elements (m is the population size) from the joined population

- *Control parameters.* The main control parameters are:
  - Population size
  - Mutation probability

- *Stopping condition.* It could be related to the number of generations, to the value of the best element in the population or to other characteristics of the population (e.g. diversity)

Hint. A variant based on the classical operators (2-opt mutation) is described in GA_TSP.sci

Exercise: Implement the other two types of mutation.

Homework.

1. Modify the onemaxGA.sci function by:

a) implementing the uniform crossover
b) implementing the "pool genes"-based mutation
c) introduces selection of survivors (instead of accepting all offsprings select survivors from the joined population containing both the parents and the offsprings); ensure the elitism (by preserving the best element in the population)