Activity:

- Draw the flowchart of bacterial evolutionary algorithm
- Explain the two bacterial operators with your own words
- Draw the figure of the two bacterial operators

The original Genetic Algorithm (GA) was developed by Holland [1] and based on the process of evolution of biological organisms. These processes can be easily applied in optimisation problems where one individual corresponds to one solution of the problem. A more recent evolutionary technique is called the Bacterial Evolutionary Algorithm (BEA) [2]. This mimics microbial rather than eukaryotic evolution. Bacteria share chunks of their genes rather than perform neat crossover in chromosomes.

In evolutionary algorithms an individual can be represented by a sequence of numbers that can be bits as well. This sequence is called chromosome which is not else than the individual itself. The chromosome is built of genes which are smaller information units.

Bacterial Evolutionary Algorithm uses two operators; the bacterial mutation and the gene transfer operation. The bacterial mutation operation optimises the chromosome of one bacterium; the gene transfer operation allows the transfer of information between the bacteria in the population. The chromosome BEA has been applied to a wide range of problems, for instance, optimising the fuzzy rule bases [2, 4], feature selection [5] and combinatorial optimization problems [6].

The algorithm consists of three steps as illustrated in Fig. 4-1-1. First, a random initial population with N_{ind} individuals has to be created. Then, bacterial mutation and gene transfer are applied until a stopping criterion is fulfilled, which is usually the number of generations (N_{gen}).

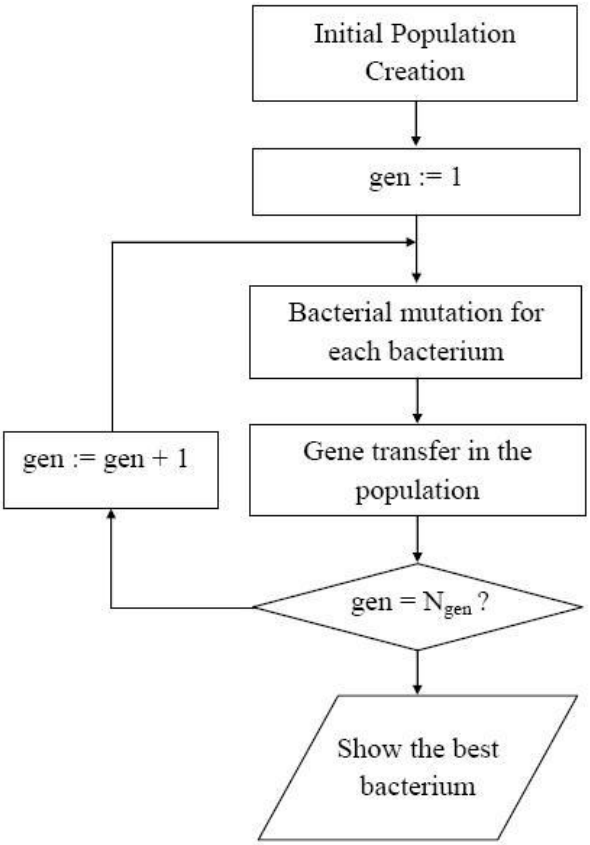

Fig. 4-1-1. Flowchart of bacterial evolutionary algorithm

When applying evolutionary type algorithms first of all the encoding method must be defined. The evaluation of the individuals (bacteria) has to be discussed, too. The encoding method and the evaluation of the individuals depend on the given problem. The operations of the algorithm have to be adapted to the given problem as well.

The bacterial mutation operation is applied to all bacteria one by one. First, N_{clones} copies (clones) of the bacterium are created. Then, a random segment of length l_{bm} is mutated in each clone except one clone which is left unmutated. After mutating the same segment in the clones, each clone is evaluated. The clone with the best evaluation result transfers the mutated segment to the other clones. These three steps operations (mutation of the clones, selection of the best clone, transfer of the mutated segment) are repeated until each segment of the bacterium has been mutated once. At the end, the best clone is kept as the new bacterium and the other clones are discharged. The bacterial mutation operation is illustrated in Fig. 4-1-2.

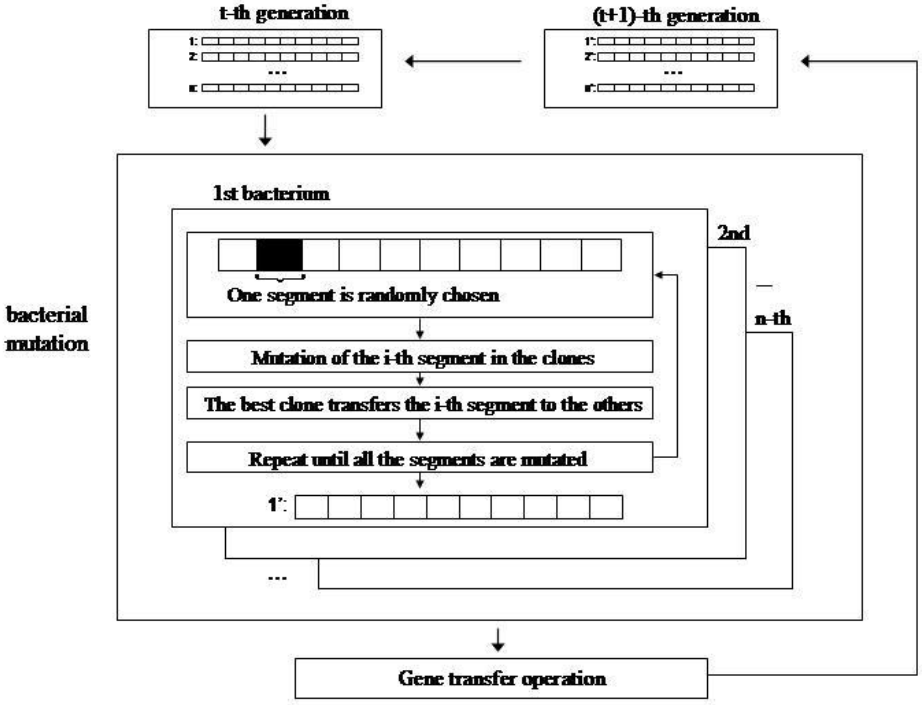

Fig. 4-1-2. Bacterial evolutionary algorithm

In the gene transfer operation first the population must be sorted and divided into two halves according to their evaluation results. The bacteria with better evaluation are called superior half, the bacteria with worse evaluation are referred to as inferior half. Then, one bacterium is randomly chosen from the superior half and another from the inferior half. These two bacteria are called the source bacterium, and the destination bacterium, respectively. A segment of length l_{gt} from the source bacterium is randomly chosen and this segment is used to overwrite the same segment of the destination bacterium. The above steps (sorting the population, selection of the source and destination bacteria, transfer the segment) are repeated N_{inf} times, where N_{inf} is the number of "infections" per generation. The gene transfer operation is illustrated in Fig. 4-1-3.

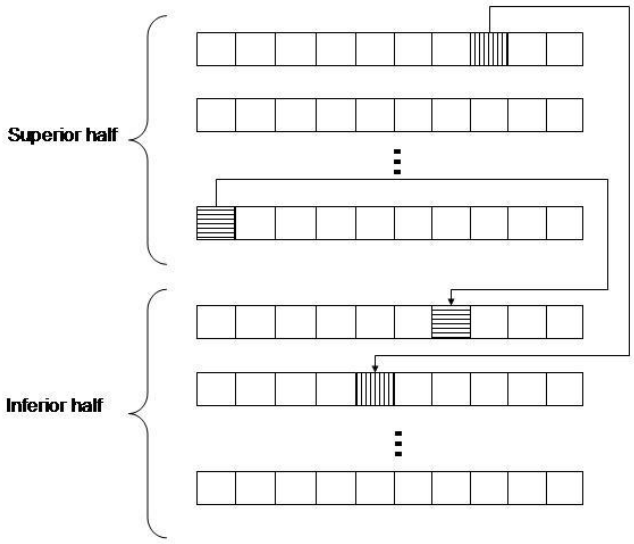

Fig. 4-1-3. Gene transfer operation

Activity:

- See how the bacterial evolutionary algorithm can be applied for fuzzy rule base optimisation
- Try to encode a fuzzy rule base with triangular shaped membership function into a bacterium

The bacterial evolutionary algorithm can be applied for automatic fuzzy rule base generation. The first step is to find the way how the problem can be encoded in a bacterium (chromosome). Trapezoidal shaped membership functions are used in the fuzzy rules. The task is to find the optimal fuzzy rule base to a pattern set. Thus, the parameters of the fuzzy rules must be encoded in the bacterium. The parameters of the rules are the breakpoints of the trapezoids, thus, a bacterium will contain these breakpoints. Each rule has 4(n+1) parameters, where n is the number of rule inputs. For example, the encoding method of a fuzzy system with two inputs and one output can be seen in Fig. 4-1-4. In Fig. 4-1-4 the number of rules is R. For example Rule 3 in Fig. 4-1-4 means:

R3: **If** x_1 is $A_{31}(4.3; 5.1; 5.4; 6.3)$ and x_2 is $A_{32}(1.2; 1.3; 2.7; 3.1)$ **then**
y is $B_3(2.6; 3.8; 4.1; 4.4)$.

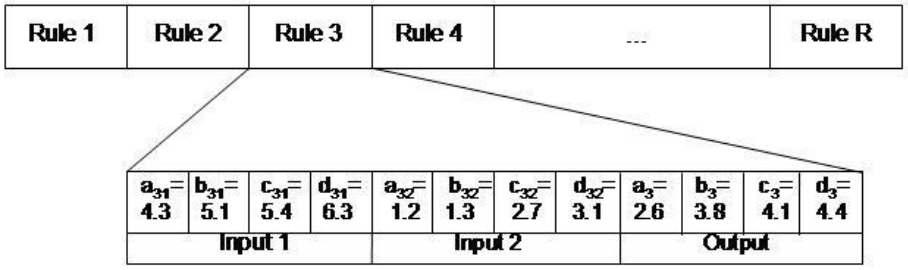

Fig. 4-1-4. Fuzzy rules encoded in a chromosome

The evaluation of the individuals is performed by the Mean Squared of Errors (MSE):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - t^{(i)})^2 \qquad (1)$$

where $y^{(i)}$ is the output of the fuzzy system for the i-th pattern, $t^{(i)}$ is the desired output for the i-th pattern and m is the number of patterns.

Activity:

- See how the rule reducing operators work
- Try to use the rule reducing operators on example membership functions

In the evolutionary process between the bacterial mutation and gene transfer operations rule reduction operators can be applied for each individual [4]. The evolutionary operators optimise the membership functions, besides, it is important to optimise the structure of the rule base, too. Ineffective rules must be eliminated and similar rules must be contracted. The following operators help to optimise the number of rules.

Annihilation operator: When a membership function becomes too narrow, the rule using it must be deleted. The evaluation criterion is as follows (see Fig. 4-1-5) [4]:

$$l_i \mu_i\left(\frac{a_j + b_j + c_j + d_j}{4}\right) \geq \beta l_j \qquad (2)$$

where l_i and l_j are the lengths of the medians of the membership functions of the given input or output variables in the i^{th} and j^{th} rule and β is the annihilation parameter. The larger is the value of β, the more severe is the annihilation criterion.

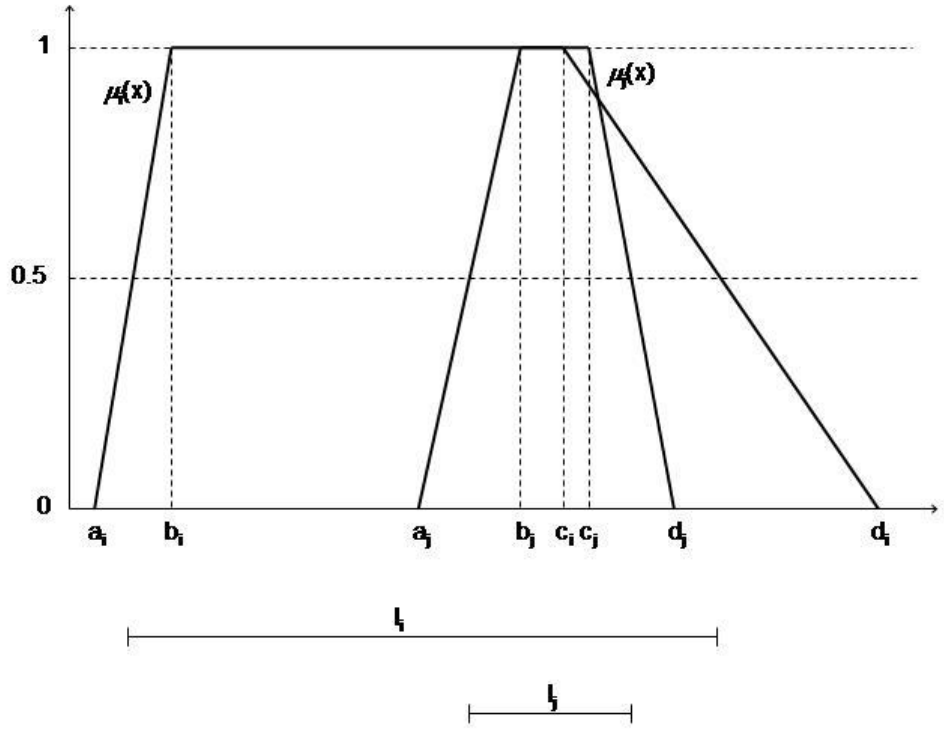

Fig. 4-1-5. Annihilation operator

Fusion operator: If two membership functions, belonging to the same variable are near to each other, and the difference between the lengths of their median is small enough, then they are fused in a single membership function (Fig. 4-1-6). There are two criteria of fusion [4]:

$$\left|\frac{l_i}{l_j}-1\right|<\gamma \quad \text{and} \quad |f|<\gamma \qquad (3)$$

where l_i and l_j are the lengths of the median of the membership functions of the given variable in the ith and jth rule, and f is the distance between the centres of l_i and l_j. To execute fusion both criteria must be satisfied, but only one parameter, γ is used. The smaller is the value of γ, the more severe is the criterion of fusion. The parameters of the fused membership function will be:

$$z_{fus}=\frac{z_i l_i + z_j l_j}{l_i + l_j} \qquad (4)$$

where z stands for the four breakpoints (a, b, c and d) of a trapezoidal membership function.

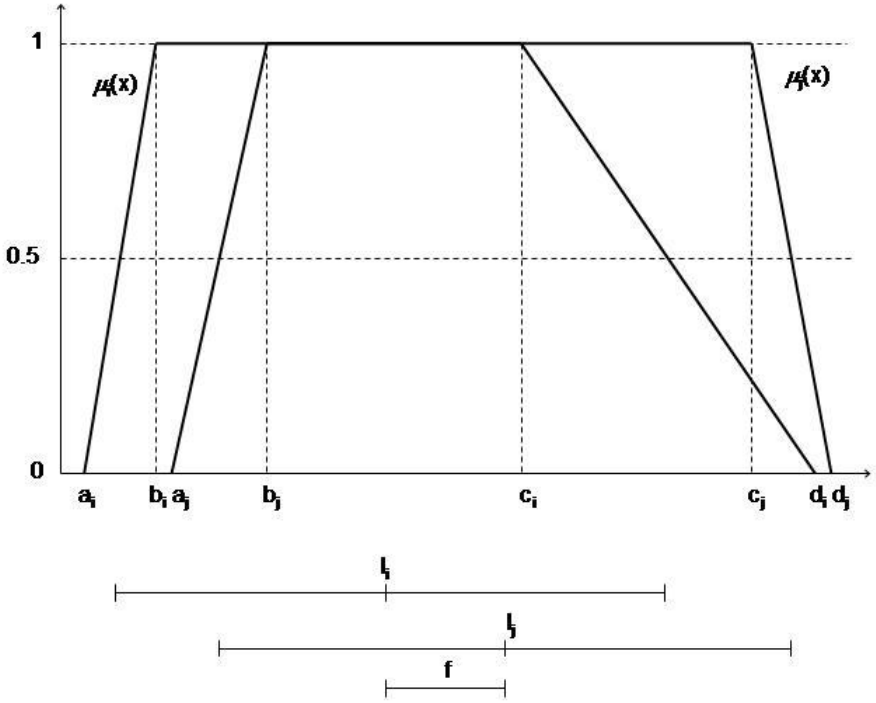

Fig. 4-1-6. Fusion operator

Semantic analysis operator: If two rules have the same antecedents but a different consequent, the membership functions of the output variable are fused in one output membership function by using the fusion described above [4].
Rule removal operator: If two rules are identical the second one is deleted. The remaining rule can be let unchanged, or its importance can be increased as in [4].
After the fusion operators it can happen that the antecedent parts of two rules become the same. In this case the algorithm merges their consequent parts, thus the two rules become identical, one of them can be eliminated. So the effect of the fusions can be seen after applying these two other operators.