7TH INTERNATIONAL COLLOQUIUM ON NUMERICAL ANALYSIS AND COMPUTER SCIENCE WITH APPLICATIONS, PLOVDIV, 13 - 17.08.1998

EVOLUTIONARY COMPUTATION AND APPLICATIONS

Halina Kwaśnicka
Department of Computer Science,
Wrocław University of Technology, Wrocław, Poland
kwasnicka@ci.pwr.wroc.pl ,
http://www.ci.pwr.wroc.pl/~kwasnick

Keywords: genetic algorithm, evolutionary techniques, optimisation.
Subject Classification Primary: 68T05, Secondary: 68Q05

INTRODUCTION

- Optimisation techniques called ***Evolutionary Algorithms*** (***EAs***) are inspired by biological evolution (genetics and the process of natural selection).

Theories of Evolution

☞ Lamarck:

- Evolution occurs when environment changes, (+)
- Offsprings inheritance a part of acquired characteristic (when parents try to adopt to the environment). (–)

☞ Darwin:

- Evolution is forced by the process called ***natural selection***: better adopted individuals (fittest ones) have bigger chance to life and to have offsprings, (+)
- Variation is present in all species (due to mutations and recombinations). (+)

☞ Mendel:

- Dominant and recessive characters (genes). (+)

Optimisation process

☞ Mathematical point of view:

- The possibility of achieving the global optimum.

☞ Optimisation perceived as '*human like*' process:

- It is a process of improvement which moves a solution to the optimal points,
- Optimisation can be partitioned into the two phases:
  - the process of improvement,
  - the process of checking if the optimal solution is achieved.

☞ Man optimisation:

- We make decisions by selecting out of knowing us possibilities,
- Everyone wants to act better then others.

If we like to make the optimisation process more '*human like*', we must lay stress on the possibilities of reaching quickly the satisfactory solution. Inspiration for such techniques can be taken from the nature.

EVOLUTIONARY COMPUTATION (EC)

☞ **Evolutionary Algorithms** (***EAs***)

Taking into account the structures of individuals, strategies of reproduction, genetic operators, **EAs** can be grouped into (see Riccardo Poli):

- Evolutionary Strategies (**ESs**) (H-P. Schwefel, I. Rechenberg)
- Evolutionary Programming (**EP**) (Lawrence Fogel)
- Classifier Systems (***CFSs***) (De Jong: Pitt approach; Holland: Michigan approach)
- Genetic Algorithms (**GA**) (J.H. Holland, D.E. Goldberg)
- Genetic Programming (**GP**) (J.Koza)

☞ **Evolutionary Strategies** (***ES***s) are used for parameter optimisation problems:

- A chromosome is a vector of parameters,
- The (1+1)***ES*** goes as follow:
  - one parent produces one offspring (with normally distributed mutations),
  - if a child is better then the parent, he takes parent's place,
  - on the basis of a success rate, the standard deviation of the normal distribution is changed

    (successful_mutations/all_mutations = 1/5)
- the $(\mu +1)$***ES*** includes also recombination
- the $(\mu+\lambda)$***ES*** differs in the number of produced offsprings: (λ)
- the (μ,λ)***ES***, $\mu<\lambda$, μ offspring are selected as new parents

☞ Evolutionary Programming (*EP*):

- ***EPs*** – representation of chromosomes as finite state machines. The initial goal: to forecast n+1 event knowing n earlier events. **EPs** are similar to the $(\mu+\mu)$***ES*** without recombination.
- ***EPs*** go as follow:
  - choose randomly an initial population (μ solutions, $\mu>1$),
  - replicate each solution into a new generation of population,
  - mutate each individual according to the Gaussian distribution (standard deviation of the distribution depends on the fitness), and include him to the new generation of population (we obtain $2\cdot\mu$ individuals),
  - assess each mutated individual and select the μ best individuals as the final next generation of the population.
- The mutation operator is controlled by parameters that are also optimised.

☞ Classifier Systems (*CFSs*)

- The two ways can be applied to improve the performance of classifier systems:
  - Adaptation by *credit assignment* (unsupervised learning, some heuristic for assessing existing classifiers are needed),
  - Adaptation by *rule discovery* (classifiers are evaluated on a basis of their performance, the training set is used).
- Measure of classifiers' quality usually covers its *strength* (*usefulness*), it is a measure of performance of the whole system, and its *specificity* (*relevance*).

☞ Genetic Programming (*GP*)

- Potential solution is a *programme*, an individual is not a fixed length character string but it is a parse tree
- The set of terminal nodes (leaf) ***T***, called *terminal set* and the set of internal nodes ***F*** *– function set*, must be defined for ***GP***

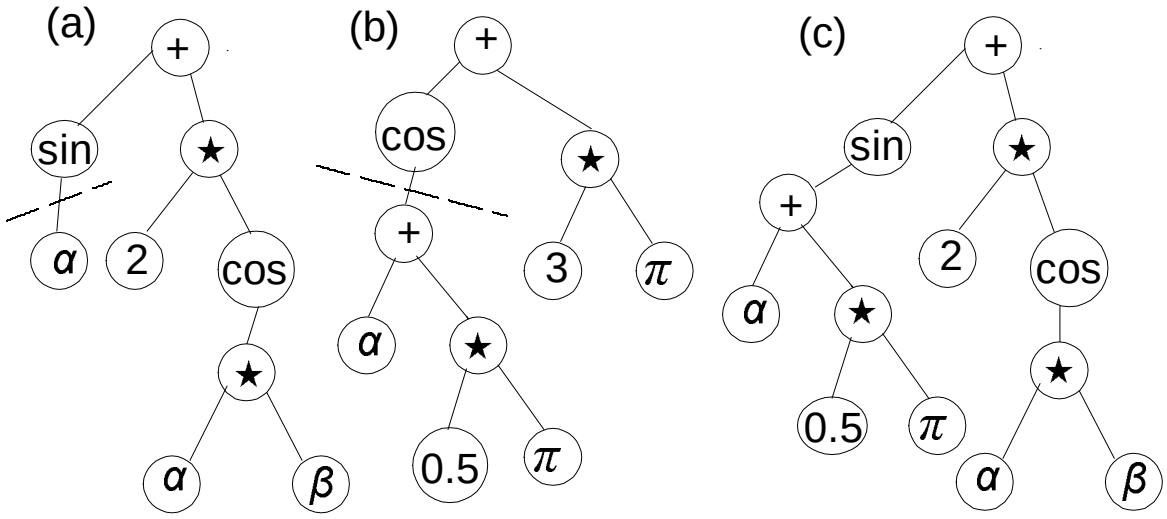


Figure 2. An example of individuals in **GP**: (a) the first parent, (b) the second parent, (c) one offspring as a result of crossover (the dotted lines show the points of crossover)

$$\boldsymbol{F}=\{\sin, \cos, +, \star\}\boldsymbol{T}=\{\alpha, \beta, \pi, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

- *Sufficiency* is guaranteed only if the function and terminal sets are suitable for a given problem (solution can be expressed by assumed functions).
- To prevent runtime errors, ***F*** and ***T*** should respect the *closure* property (each function can accept as its arguments any value that may be returned by function in $\boldsymbol{C}=\boldsymbol{F} \cup \boldsymbol{T}$).
- Special operators are developed for ***GP***.

OPTIMISATION USING *GAs*

☞ A generic scheme

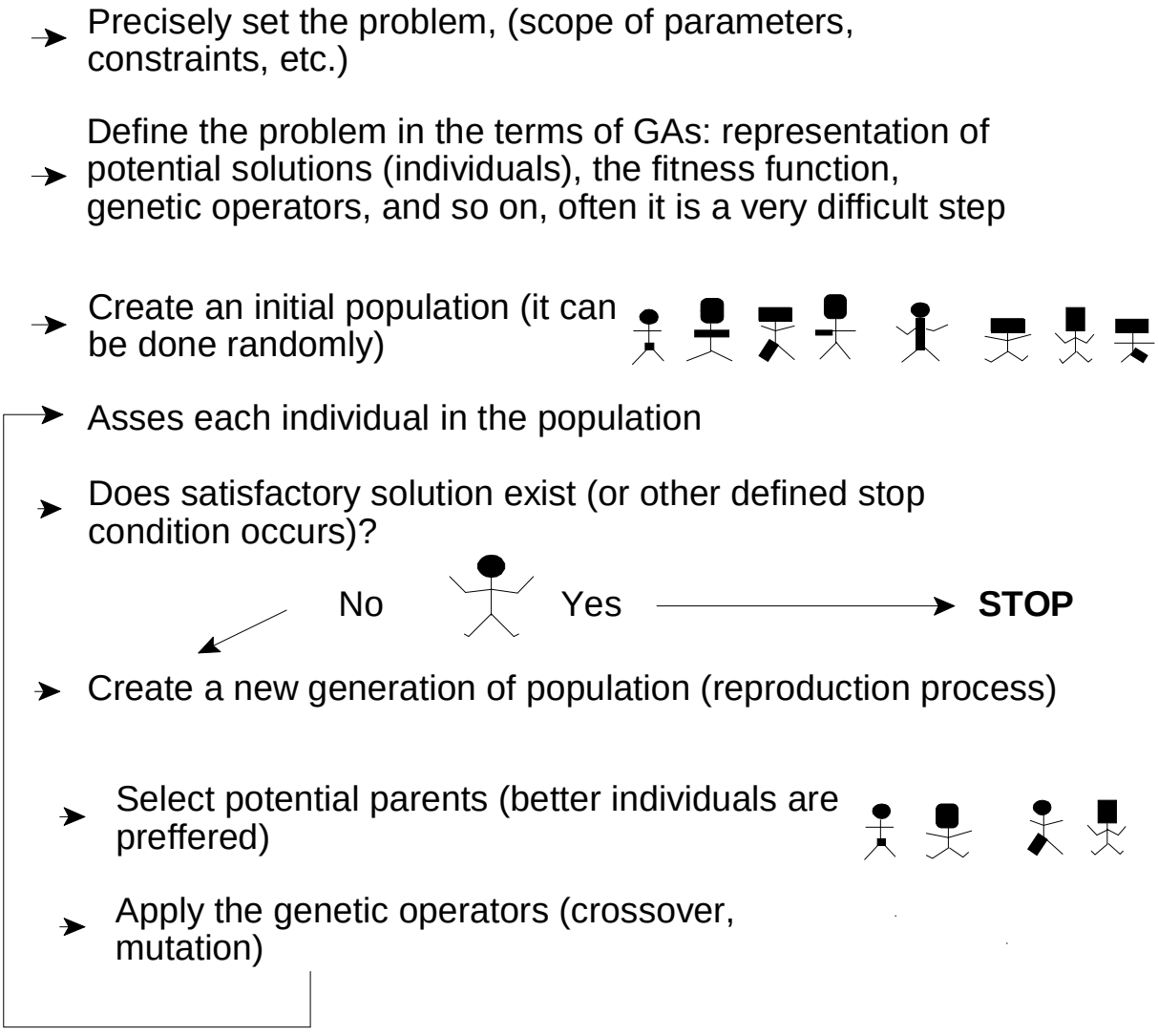


Figure 3. A generic schema of Genetic Algorithms

☞ The simple example of using a GA

To find an integer value I, $0 \geq I \leq 15$ whose binary representation has the maximum number of transitions (0→1 and 1→0).

Decisions:

representation: 4 bits
decoding: decimal conversion
fitness: a number of transitions
population: 4 individuals

selection: proportionate
genetic operators: crossover and mutation
initial population: random

Table 1. Coded and decoded solutions and their fitness

Genes (G)	**Fenes** (F)	**Fitness** (Q)
0000	0	0
0001	1	1
0010	2	2
0011	3	1
0100	4	2
0101	**5**	**3 (*)**
0110	6	2
0111	7	1

Genes (G)	**Fenes** (F)	**Fitness** (Q)
1000	8	1
1001	9	2
1010	**10**	**3 (*)**
1011	11	2
1100	12	1
1101	13	2
1110	14	1
1111	15	0

Initial population:				Generation 1:			
indiv.	G	F	Q	indiv.	G	F	Q
i_1	0001	1	1	**i_2+i_1**	**0101**	**5**	**3**
i_2	0100	4	2	i_2+m	1100	12	1
i_3	1100	12	1	i_3+m	1101	13	2
i_4	1111	15	0	i_1+i_3	0000	0	0

After selection: i_2, i_2, i_3, i_1. Operators:
(1) crossover: i_2–i_1 (last place); (2) mutation 1st bit;
(3) mutation 4th bit; (4) crossover i_1–i_3 (2nd place).

☞ A schema of representation

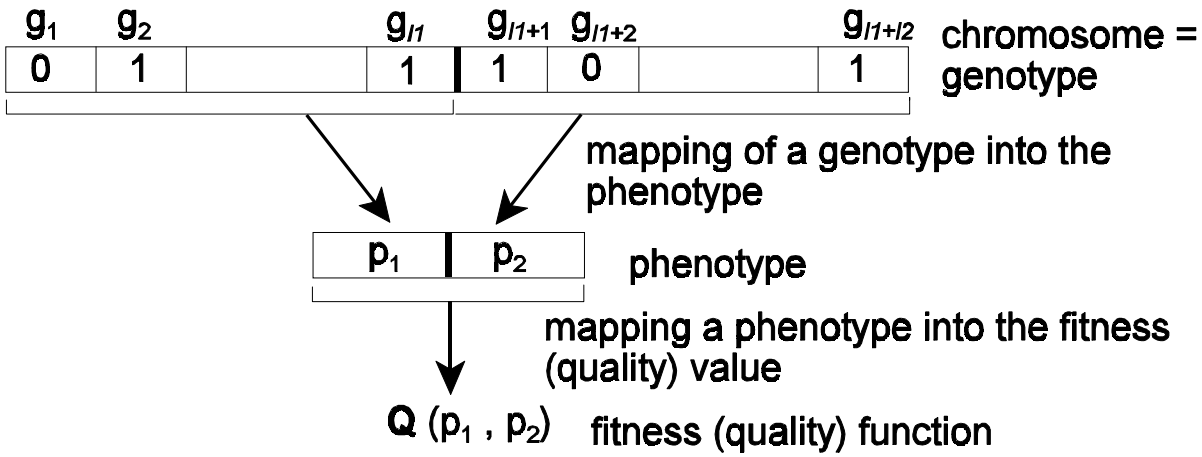


Figure 4. The binary representation of an individual in a **GA**

☞ Coding

- Phenotype: a vector of m parameters (phenes) $[p_1,...,p_m]$.
- Genotype: a single chromosome = bits string. A number of bits (l_i) required for each phene p_i is calculated:

$$(b-a)\cdot 10^k \le 2^{l_i-1}, \qquad \textbf{(1)}$$

where: $[a, b]$ – the range of a phene, k – the assumed precision, l_i – the lowest number satisfying eq.1

☞ Decoding

- The value of a phene p_i is calculated as:

$$p_i = a + decimal(0101...1)\cdot \frac{b-a}{2^{l_i}-1}, \qquad \textbf{(2)}$$

where: *decimal* (0101...1) – the decimal value of bits' string.

- Gray code can be used (000,001,011,010,110,111,101...).

☞ Fitness (quality function *Q*)

- In the simple cases fitness function *Q* is directly the maximised function *F* defined on the phenotype space:

$$Q(p_1,...,p_m)=F(p_1,...,p_m) \quad \textbf{(3)}$$

- If maximized function takes negative values, we can define a fitness function *Q* as:

$$Q(p_1,...,p_m)=\begin{cases} F(p_1,...,p_m)+C_{\min} & \text{if } F(p_1,...,p_m)+C_{\min}>0 \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(4)}$$

- If we must minimise function *F* (e.g., cost or error function) we can define fitness measure as:

$$Q(p_1,...,p_m)=\begin{cases} C_{\max}-F(p_1,...,p_m) & \text{if } C_{\max}>F(p_1,...,p_m) \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(5)}$$

- Sometimes the fitness function should be scaled because **GAs** lead to the stagnation or premature convergence

$$Q'(p_1,...,p_m)=\begin{cases} a\cdot Q(p_1,...,p_m)+b & \text{if } a\cdot Q(p_1,...,p_m)+b>0 \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(6)}$$

a, *b* is counted to assure that $Q'_{av}= Q_{av}$ and $Q'_{max}= C_{mult}\cdot Q'_{av}$.

☞ Parents selection schemes (the most popular)

☞ *Proportional* (to the individual's fitness value)

- *i*-th individual is selected with probability

$$p_i = \frac{q_i}{\sum_{j=1}^{N} q_j} \qquad \textbf{(7)}$$

where q_i – fitness of *i*-th individual, N – a size of a population.

☞ A *roulette wheel* (called also *stochastic sampling with replacement*) – a realisation of a proportional method

- Fitness values of all individuals are summarized, each individual has assigned the slot sized in proportion to its fitness,
- A random number is generated, and the individual with slot corresponding this number is selected as a parent.

☞ A *deterministic sampling* method

- Probabilities of selection and expected numbers of offsprings are (as usual) calculated as:

$$E_i = N \cdot p_i, \qquad \textbf{(8)}$$

where: E_i – an expected value of number of offsprings for *i*-th individual, p_i – as in eq. 7

- Each individual is selected to the reproduction according to the integer part of the E_i,
- A population is sorted according to the fractional parts of the E_i, individuals from the top of the list are selected to assure the constant size.

☞ A ***remainder stochastic sampling with replacement***

- It starts as the deterministic sampling up to fractional parts, fractional parts are used to create a roulette wheel,
- A deficient part of a population is selected using this roulette wheel.

☞ A ***remainder stochastic sampling without replacement***

- Fractional parts of E_i are used as probabilities for Bernoulli trials, so the size of an evolved population is constant.

☞ A ***stochastic tournament selection***

- Two individuals are selected according to the roulette wheel method,
- The best individuals are chosen as a parent.

☞ An ***elitist selection***

- The best individuals (at least one) are passed to a new generation.

☞ **Genetic operators**

☞ ***Crossover*** – two individuals exchange parts of their chromosomes (a one-point or multipoint crossover can be assumed).

☞ ***Mutation*** – a number of bits of new individuals can be changed (from *zero* to *one* or opposite).

☞ ***Inversion*** – a simple reordering operator, a part of a chromosome (between two points) is ordered in the opposite way.

☞ Analytical backgrounds of GAs

- A ***schema*** (S) – a string of the three symbols: *0*, *1*, and ⋆ (‘*inessential*’ or ‘*anything*’).
- Each S represents 2^k strings, k is a number of ‘⋆’ in S, n individuals can represent up to $n \cdot 2^m$ schemata, m is the length of strings.
- ***Implicit parallelism*** of a **GA** – Holland assessed that n^3 schemata are successively processed by a **GA**.
  - ***order*** of schema S, $o(S)$, it is a number of fixed positions in a schema (i.e., number of *zeros* or *ones*), $o(0\star11\star)=3$,
  - ***length*** of schema S, $\delta(S)$, is the distance between the first and the last fixed position in schema S, $\delta(0\star11\star)=4-1=3$.

Examining a number of individuals matched to the S, assuming mutation and crossover, the ***Schemata Theorem*** is formulated:

> *Low order* and *short length schemata* with the *fitness above an average* fitness of a population are *multiplied exponentially* in subsequent generations.
>
> Such schemata are called *building blocks* because a **GA** seeks a good solution by placing building blocks close together (combining them).
>
> Building blocks can lead a **GA** in the wrong direction and cause a premature convergence.

☞ **Special operators for genome reconfiguration** (**sequential problems**, **e.g.**, **the *Travelling Salesmen***)

- Genes are the integer values (e.g., numbers of cities),
- Each value has to appear in the chromosome one time,
- The simple inversion does not give good effects.

☞ ***Partially Matched Crossover*** (PMX); it has a tendency to preserve absolute positions of genes

```
I1 : 1 2 3 | 4 5 | 6 7 8
I2 : 4 3 2 | 7 6 | 5 8 1
-------------------------
I1': 1 2 3 | 7 6 | 4 5 8
I2': 7 3 2 | 4 5 | 6 8 1
```

Figure 5. Two individuals before and after the PMX

☞ ***Order Crossover*** (**OX**); it has a tendency to preserve relative positions of genes

```
I1 : 1 2 3 | 4 5 | 6 7 8
I2 : 4 3 2 | 7 6 | 5 8 1
-------------------------
I1 : 1 2 3 | 4 5 | o o 8
I2 : o 3 2 | 7 6 | o 8 1
I1 : 3 4 5 | o o | 8 1 2
I2 : 2 7 6 | o o | 8 1 3
-------------------------
I1': 3 4 5 | 7 6 | 8 1 2
I2': 2 7 6 | 4 5 | 8 1 3
o - hole
```

holes shifting

Figure 6. Two individuals before and after the OX crossover

☞ *Cycle Crossover* (CX)

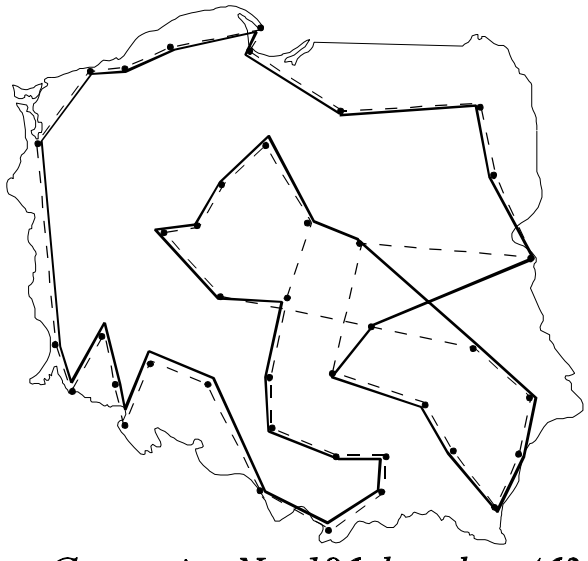
```
I1  : 1 2 3 4 5 6 7 8
I2  : 4 3 2 7 6 5 8 1

I1': 1 - - - - - - -
I1': 1 - - 4 - - - -
I1': 1 - - 4 - - 7 -
I1': 1 - - 4 - - 7 8
I1': 1 3 2 4 6 5 7 8

I2': 4 - - - - - - -
I2': 4 - - - - - - 1
I2': 4 - - - - - 8 1
I2': 4 - - 7 - - 8 1
I2': 4 2 3 7 5 6 8 1
```

Figure 7. Creation of descendants using the CX

☞ An example of the TSP problem (40 cities in Poland):



- - - *Generation No. 186, length = 4636 km.*
—— *Generation No. 456, length = 4588 km.*

Figure 8. Exemplary routs found by a GA

☞ Handling constraints

☞ A Goal function is $F(p_1, p_2, ..., p_m)$, where p_i can take a value from the range of $[p_i^{min}, p_i^{max}]$. **GAs** produce solutions out of allowed ranges.

- *Adding penalties* – it is the most popular and very simple method.

  A penalty depends on the size of exceeding of allowed ranges.

  A fitness function Q is equal to:

$$Q(p_1,p_2,...,p_m) = F(p_1,p_2,...,p_m) + coef \cdot \sum_{i=1}^{m} Pen(p_i) \qquad \textbf{(9)}$$

  where: *coef* – a penalties coefficient, $Pen(p_i)$ – penalty for p_i , $Pen(p_i)=0$ for $p_i \in [p_i^{min}, p_i^{max}]$, $Pen(p_i) > 0$ otherwise.

  Values of penalties are selected experimentally.

- *Removing* bad solutions – it consumes computational time.

- *Repairing* bad solutions – it is often used, e.g., in **NNs** designing.

☞ Parallel and distributed GAs

☞ ***Parallel Genetic Algorithms*** (**PGA**s): mainly empirical studies are made (a lack of theoretical background); it is impossible to compare different approaches

☞ **The categorization of PGAs** (**Erick Cantú-Paz**):

- ***Global parallelization*** – all genetic operators and the evaluation of all individuals are explicitly parallelized. Such implementation is easy, it does not require any modification of the classic **GA**.
- ***Coarse grained* PGA**s – a process is *coarse grained* if the ratio of the computation time to the time of communication between processors, is high.

  It requires a division of a whole population into a number of relatively large subpopulations, called ***demes*** (the term is borrowed from the biology, e.g., geographic isolation).

  An additional operator called ***migration*** is introduced: an individual can be moved from one deme to another one.

  Depending on the migration model, we distinguish an *island model* and a *stepping stone model*.
- ***Fine grained* PGA**s is based on the division of a population into very small demes (demes with one individual are preferred).
- ***Hybrid approaches*** – all above method can be combined.

EXAMPLES OF APPLICATIONS

☞ **Machine Learning**: **Genetic Algorithms in rules learning** (parametrized general rules optimization)

☞ **Construction of parametrized rule** (having data that represent examples of situations and undertaken decisions)

- Data: a sequence of *n* training patterns, e.g., $v_{1,i}$,$v_{2,i}$, d_i (*two* variables v_1, v_2; decision *d*; *n* examples, *i*=1, ..., *n*),
- One must decide what operators can be used to assign the variables and their values:
  - Divide the values of v_1 and v_2 into a number of ranges,
  - Choose operators acting on the variables and their values, for example *grater-then*, *less-then*.
- Our general rule can have a form:

  IF v_1 *operator*$_s$ $v_{1,bj}$ AND v_2 *operator*$_p$ $v_{2,bk}$ THEN $d = d_i$

☞ **Defining a chromosome**

- All operators, all limits for the assumed intervals of all variables, and values of all possible decisions should be coded. The rule is a coded sequence:

  $[\text{operator}_s, v_{1,bj}, \text{operator}_p, v_{2,bk}, d_i]$

- Fitness function is a measure of conclusions' correctness.
- If the decision is simple *yes* or *no*, we can throw out the decision d_i from the chromosome (learn only the rules which contain the decision *yes*).

☞ An example

- The learning set (7 training examples):

v_1	300	400	500	600	700	800	900
v_2	11	28	10	19	56	18	13
d	F	F	T	T	F	T	T

- We assume the two operators: o_1 as $\geq$, and o_2 as $<$
- Limits of ranges for v_1: $v_{1,b1} = 500$, $v_{1,b2} = 750$
- Limits of ranges for v_2: $v_{2,b1} = 10$, $v_{2,b2} = 20$, $v_{2,b3} = 30$
- One of the rules learned by the **GA** can be as follow:

Rule 1: IF $v_1 \geq 500$ AND $v_2 < 20$ THEN d = T

- Coding (binary):
  - operators (single bit): $\geq$ coded by *1*, $<$ coded by *0*
  - the v_1 limits (single bit): *500* by *0*, *750* by *1*
  - the v_2 limits (two bits): *10* by *01*, *20* by *10*, *30* by *11*

  The chromosome coded the *Rule 1*is: **[1 0 0 10]**
- A fitness function: a number of correct decisions
- An initial population: random generated
- Genetic operators: standard crossover and mutation.

☞ **Machine Learning**: **Hierarchical GA in knowledge discovery from data bases** (data mining)

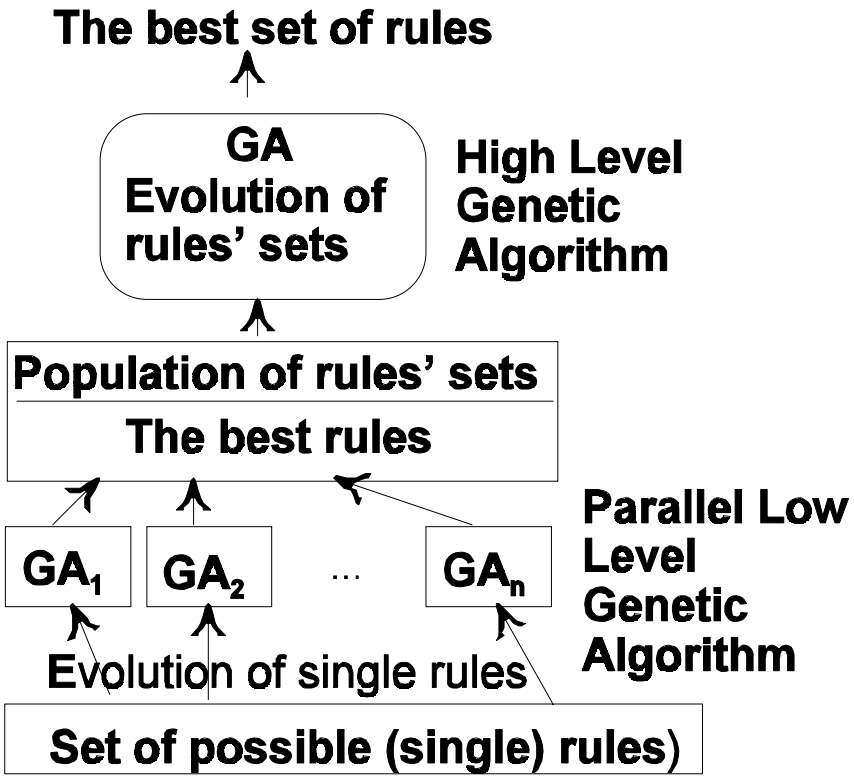


Figure 9. A schema of hierarchical GA

- A generic rule:

$$S \text{ IF } W_1 \text{ AND } W_2 \text{ AND } \ldots \text{ AND } W_n \text{ THEN } K$$

S – a *specificity*, it specifies an object and/or time of rule,

W_i – a conditional clause, K – a predictive clause.

An exemplary clause:

price(apple, today)>3·price(oranges, last_week)

- Biased genetic operators are designed and applied,
- Difficulties in fitness function designing (it is done experimentally).

☞ **Machine Learning**: **GA in features selection** (and/or construction of new features) for inductive learning algorithms

- Input data: a set of possible features,
- Output of a **GA** system: a set of the best features,
- Two GA-modules can be applied:
  - GA-Selector for selection of important features,
  - GA-Constructor for construction of new features, based on possible features.

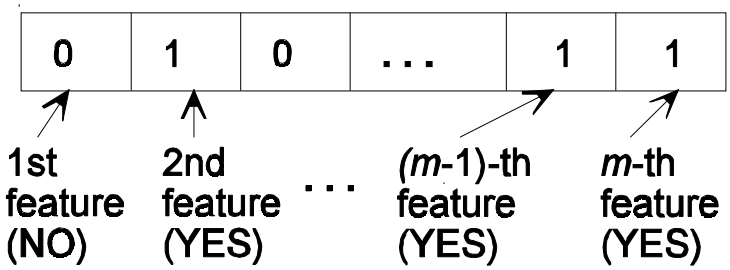


Figure 10. An individual in a GA-Selector

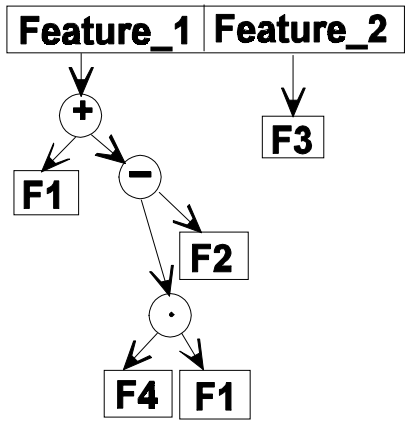


Figure 11. An individual in a GA-Constructor

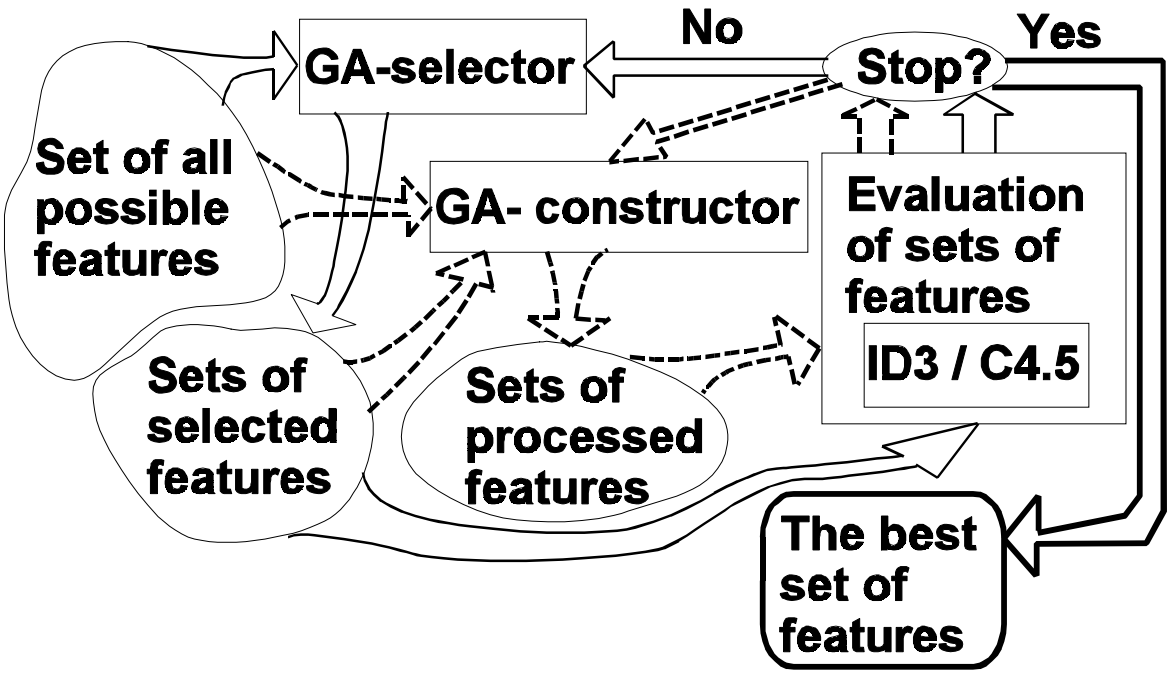


Figure 12. A scheme of learning system with features selection and construction by **GAs**

Tests:

- Image recognition, 30x30 pixels, 200 learning vectors (divided into learning and testing sets), 8 possible features.
- GA-Selector has reduced number of features up to 4,
- GA-Constructor has reduced up to 3 (two single features and the third as a combination of all fourth).

Effectiveness of recognition based on a preprocessed set of features is better than without preprocessing.

EXAMPLES OF APPLICATIONS

☞ **Neural Networks (NNs) and GAs** (last decade)

☞ **Introduction to NNs**

- The nervous systems of animals are composed of pieces, called neurons,
- The paradigm of artificial neural networks (**NNs**) is used to solve difficult problems,

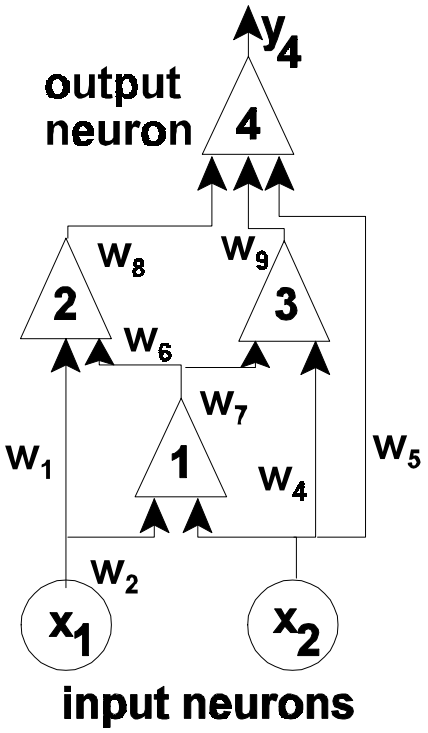


Figure 13. A feedforward network

- **NNs** consist of a number of artificial neurons and connections between those neurons,
- Performance of a **NN** depends on a number of engaged neurons and a structure of all connections,
- A **NN** does not require any algorithm of the task solution; instead of this, a **NN** has to be learned (trained) to do a task,
- Learning neural networks means to adjust the weights of all connections,
- Typically, the development of a **NN** includes four stages:

▸ Select a problem domain,

▸ A network architecture design – skeletal structure of a NN,

▸ Select a suitable algorithm for training the network,

▸ Evaluate the trained network according to measure of objective performance.

☞ Genetic training methods

- The first approach: D. Montana and L. Davis. They evolve the weights and the threshold in the fixed, layered feedforward NN (classification of underwater sonic "lofograms" for two classes). Results: the proposed GA's approach is better then a simple backpropagation.

☞ Optimization of a training set

- Evolving a learning rule for a simple, single layer network (David Chalmers). It is assumed as a linear function of weights, inputs, neuron's outputs, desired outputs, and theirs pairwise products. Results: quite good.
- Known commercial tools: BrainMaker.

☞ Selection (preparation) of training data

- Usually it is a transformation of input data and selection of the best subset of transformed data.

☞ Optimization of a structure – numbers of neurons and/or layers

- A number of hidden neurons and a learning rate (Murray). A chromosome is not coded, the fitness measure is the dollar amount earned or lost by this model on the stock market in a given period. Results are promising.
- Known commercial tools: NeuroForecaster.

GAs are used for optimisation of structure of **NNs** for difficult tasks, e.g., satellite image recognition (36 millions pixels), stock market forecasting (data from 12 years, daily, every hour, or every minute), setting a position of satellite antenna

Lecture of prof. J. Korczak, "Genetic Search in NNs optimizations" in Kule k/Częstochowy, 14-18 October 1997. http://dpt-info.u-strasbg.fr/lsiit/ orii/gria.

☞ Designing networks' architecture

- Miller et al. show: a **GA** is a promising method to automate design processes. They have developed a topology of connections of feedforward **NN** with a fixed number of neurons. For simple tasks (XOR, pattern coding) a **GA** was able to find suitable networks.
- Bornholdt and Graudenz have developed the asymmetric NNs design method using a **GA** (input and output neurons, and a cortex region). Results for some Boolean function are very promising.
- Dasgupta and McGregor used a simple structured **GA.** A higher level of a chromosome (bits string) represents connections between neurons and a low level represents the connections' weights. A fitness function reflects a measure of the sum-square-error, the correctness of the structure, and its complexity. Results for the XOR problem and for *four* by *four* encoding problem are satisfactory, but designing big **NNs** causes problems.

☞ Our experiences

☞ *Learning of the weights*

- Combining the *Instar* and GA learning for hand written digit recognition. Results are good, similarly to *Instar*.

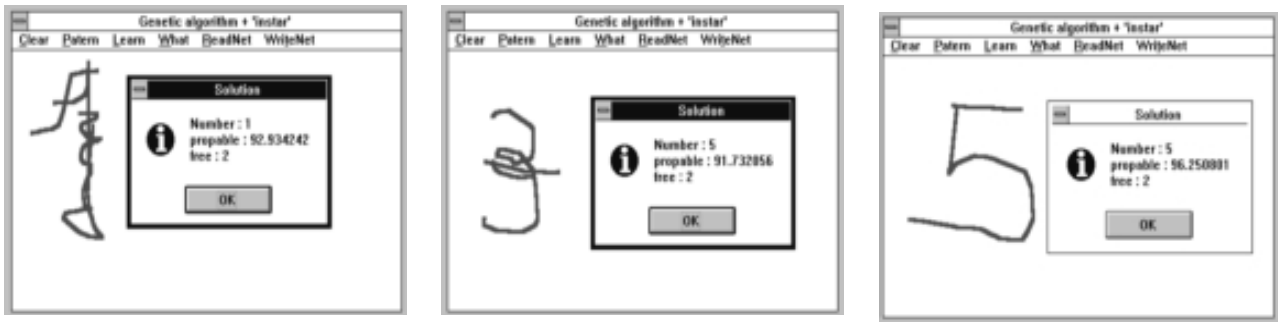


Figure 14. Digits recognition by the NN learned by Instar and GA methods

- Tasks: adding the binary numbers, recognition of weather forecasting, classifying of characterology types for psychologists (METEO and WOCC projects) – **GAs** do not give satisfactory results (backpropagation gives better results).

  Only for XOR problem results are satisfactory.

> Our experience does not allow to state that the genetic learning is better than conventional methods.

☞ *Optimising of NNs' structures*

- Developing optimal architecture for the layered **NN** which was assumed to learn play game similar to the *tick-tack-toe*. Binary coding has been used.

  Results are not satisfying, the backpropagation gives better results.

- Developing feedforward multilayered **NNs**, represented as grammar formulas (only correct networks are produced by **GA**).

  The system is still in the phase of developing, and the full results will be known in the future.

- Designing of whole structures of **NNs** (**NNG** system):

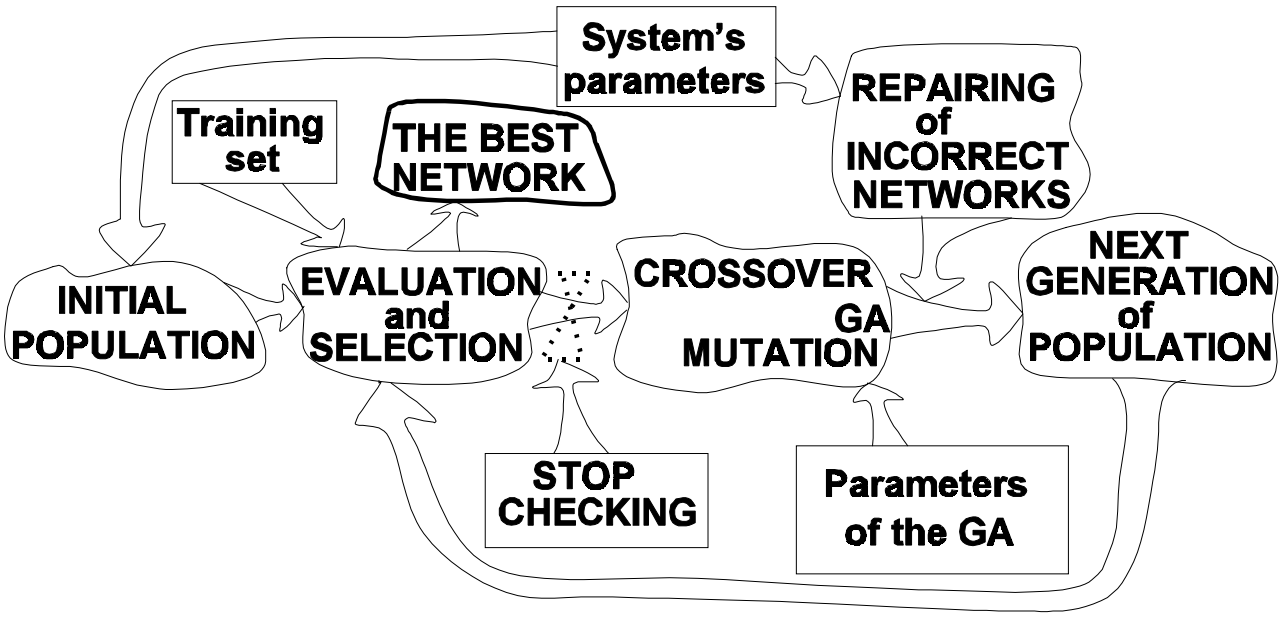


Figure 15. Neural networks architecture design using genetic algorithms

- **NN** is represented as a list of neurons, each neuron has a list of incoming connections (not coded chromosome),
- All types of connections are allowed, a user can put some restrictions, e.g., only feedforward networks,
- A user can select activation function (predefined set),
- Fitness depends on the average and maximal (in the single output) errors,
- Specialized operators are applied,
- Results: for XOR, 4x3 coding, the results are quite good. System tends to produce rather small **NNs**.

The **NNG** is able to find easy a quite good network but the tuning causes the problem. It seems that combination of genetic designing with conventional learning methods for tuning the best networks can give good results.
More research is required especially to develop a proper fitness function.

☞ *Recent studies*

- We are developing user friendly system, **NETGEN**, for further study

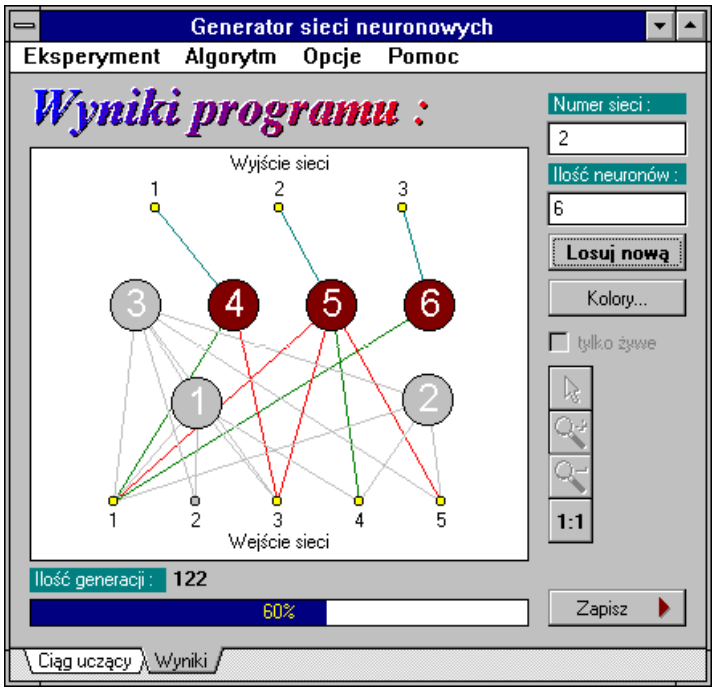


Figure 16. Visualisation of NN in the **NETGEN**

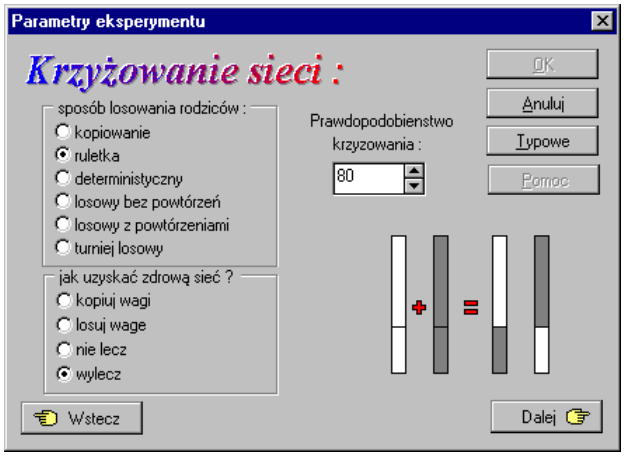


Figure 17. Setting the parameters in the NETGEN

- The system allows to evolve all types of **NNs** (planed),
- It allows to combine traditional and genetic learning,
- A user can edit structures of evolved networks,
- Networks can contain “dead” neurons (without output).

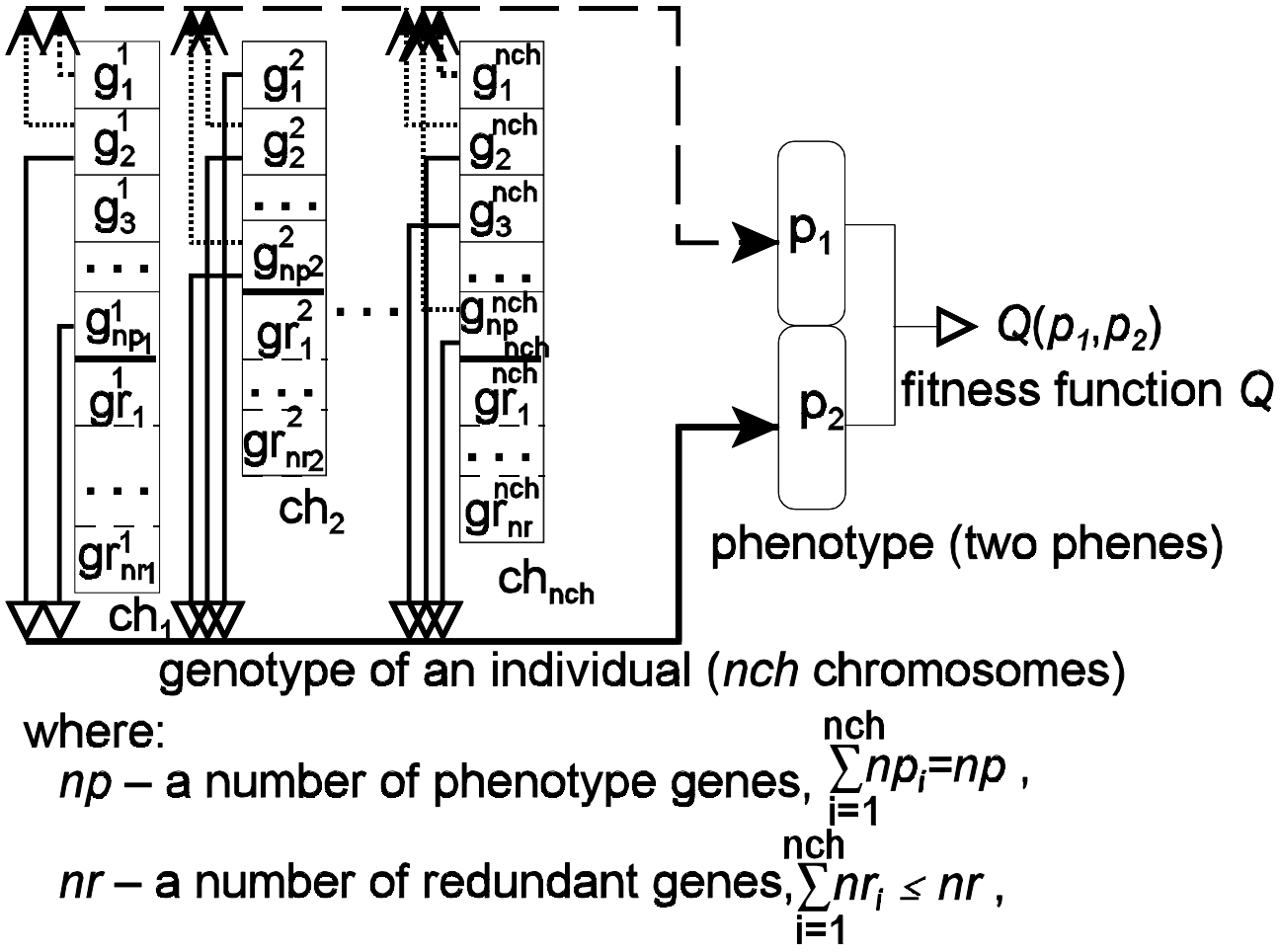
- Experiments show the necessity of further studies:

– Developing the set of “good” fitness measures,
– Dynamic control of the GA’s parameters (fuzzy logic),
– Improving the speed by applying parallel (or distributed) environments ,
– Including genotype level with pleiotropy and polygene effects.

EXAMPLES OF APPLICATIONS

☞ **Economics and Games**: **The *K*-Model** used in modelling of firms competition (industrial dynamics)

- We have developed and used the *K*-Model (with pleiotropy and polygene effects, redundand genes and macromutations):



where:

np – a number of phenotype genes, $\sum_{i=1}^{nch} np_i = np$,

nr – a number of redundant genes, $\sum_{i=1}^{nch} nr_i \le nr$,

Figure 18. The representation of an individual in the *K*-Model

- Phenes are linear combinations of a number of phenotype genes,
- An average number $\mathbf{N_e}$ of individuals is assumed,
- Numbers of offsprings are calculated according to the Poisson distribution,
- Genetic operators:
  - *Recombination*,
  - *Mutation* (also so called *neutral mutation*),
  - *Transposition* ,
  - *Transition* ,
  - *Recrudescence* (Mayr's "*loosing the cohesion of genotype*", in biology, internal stabilizing factors are responsible for that process),
  - *Crisis* (in biology, external factors are responsible for such processes).

☞ Economy and Games: Modelling and simulation of firms' competition on a market

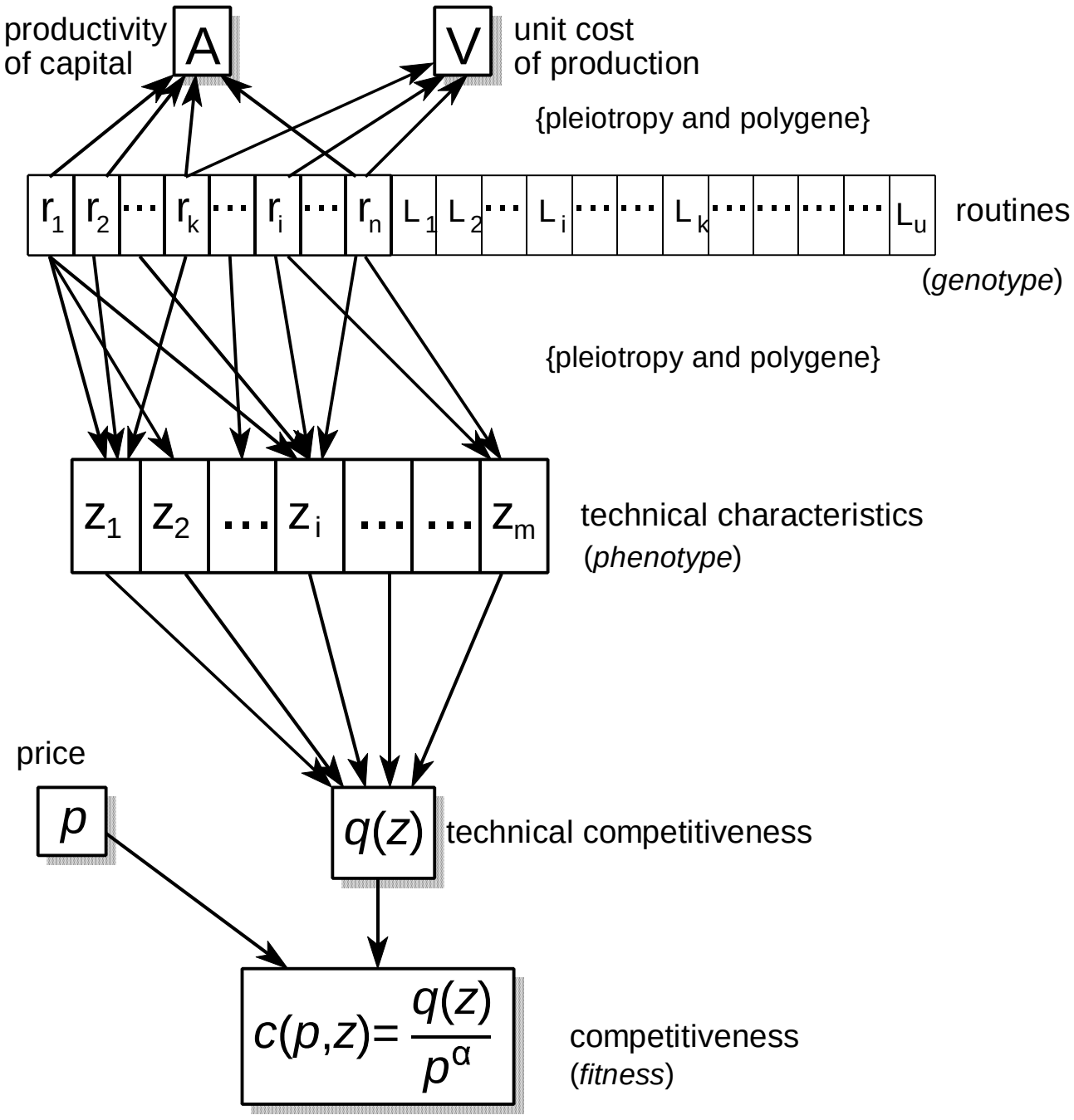


Figure 19. From routines to competitiveness, productivity of capital and unit cost of production

- Each firm has its own set of employed routines (production, management, marketing, advertisement, etc.),
- A set of routines can be partitioned into a number of subsets or segments (chromosomes),
- All segments constitute a genom of an individual,
- Some of the firm's routines are not used by the firm, they correspond to redundant (latent) genes in the *K*-Model,
- Phenes correspond to some usable and technical features of firm products (e.g., power, fuel consumption, solidity, etc.),
- A genom of firm influence the firm's characteristics,
- A situation of firm in the market depends on these characteristics.
- Simulation of firm development requires the evolution model and some economic knowledge, about firms strategies and economic laws:

  According to firm's characteristics and decision about the price of the products, the firm occupies a part of the market and collected new capital (investment)
- Each firm makes decisions concerning capital expenditure, research, investment, etc. These decisions influence the probabilities of genetic operators (e.g. mutations, recombination) in the next generation.
- A new firm can enter on the market.

  Simulation allows to observe characteristics of firms' development using different strategies.

☞ Economy and Games: Using the *K*-Model for manager game developing

- Based on the *K*-Model, the simulation games have been elaborated
- Adding some rules that reflect an assumed firm's strategy (e.g., maximal profit), a number of evolved firms can be controlled by a computer, but one (or more) is controlled by the man-player (all decisions are made by him).
- Such management games play a significant role in education.

Bankruptcy in an artificial market is not dangerous, only a player can feel a discomfort because he is a bad manager.

☞ **Economy and Games**: a **GA** is used to find a game's strategy

- In this approach, **GA** is used to find a strategy. This strategy is implemented in a game-program, and the game-program does not require any evolutionary computation.
- The GOLEM programme has been elaborated and used for searching a strategy for firm producing drinks.
- A chromosome consists of two parts:
  - 6 functions, F_1 to F_6, are represented as trees, they describe values of variables in the production phase (e.g., amount of production particular products). Nodes of a tree contain information values (e.g., actual prices, trade mark, amounts of reserves, production costs) and operators taken from a predefined set.
  - 9 pointers (integer values) used in the sale phase (e.g., margins of profits, capital for investment).
- Individual plays assumed number of games and an average profit is assumed as fitness value.

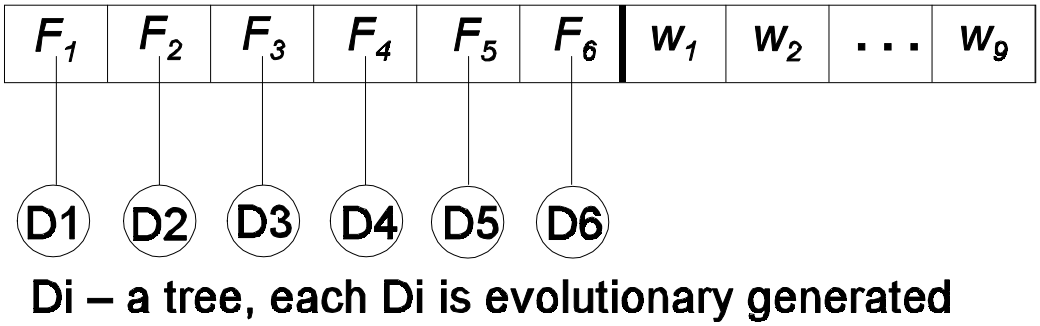


Figure 20. An individual in the GOLEM program

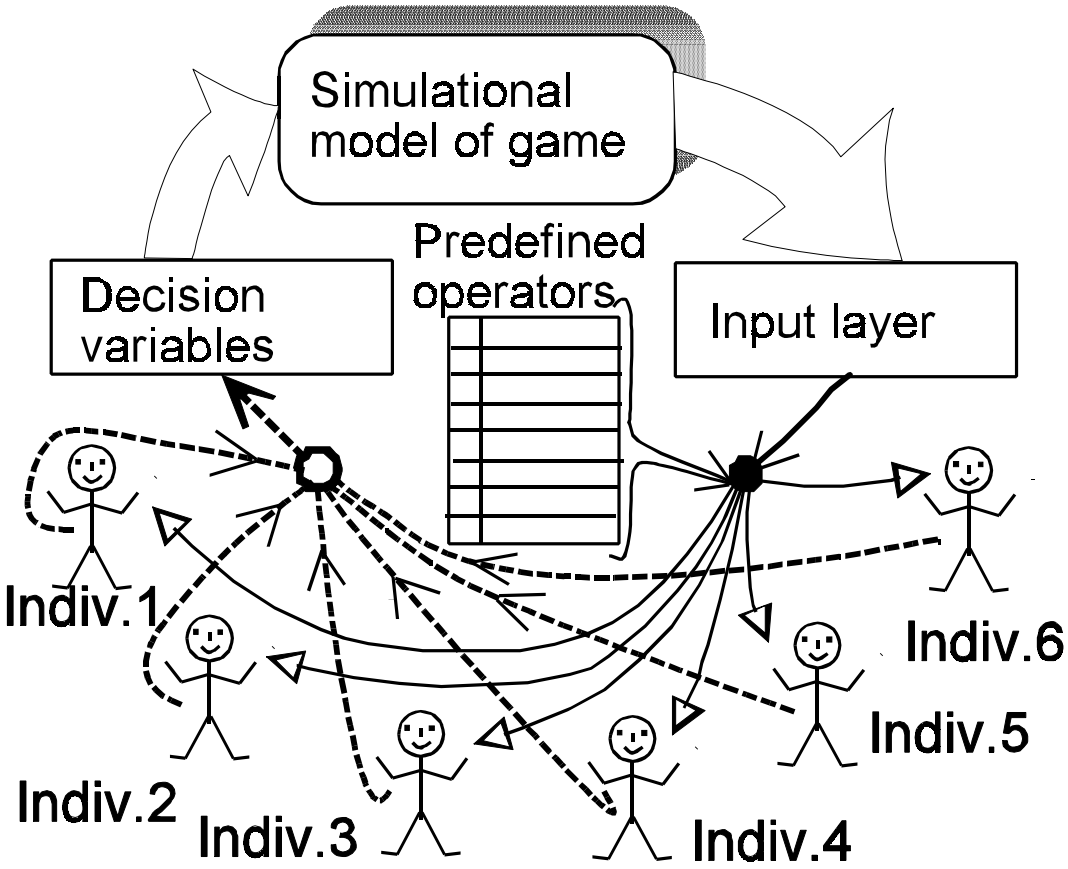


Figure 21. A generic scheme of game in the GOLEM program

- Obtained strategies are simple because the assumed market model is simple. A **GA** was able to find assumed simplifications and to exploit them. It finds a simple strategy:

  Sell with high profit.

  This strategy has high fitness value in the assumed simple market.

- Developed by a GA strategy is implemented as one (or more) of the players into a manager game programme. Such games are very useful as educational games.

References

[1] Back, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, Oxford, 1996.

[2] *Building Classification Models: ID3 and C4.5* in http://yoda.cis.temple.edu: 8080/UGAIWWW/lectures95/learn/C45/

[3] Bala J., Huang J., Vafaie H., DeJong K., Wechsler H., *Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification*, IJCAI Conference, Montreal, August 19-25, 1995.

[4] Bornholdt S., Graudenz D., *General Asymmetric Neural Networks and Structure Design by Genetic Algorithms*, Neural Networks, Vol.5, pp. 327-334, 1992.

[5] Buckles B.,P. and Petry, F.,E. (Ed.), *Genetic Algorithms*, IEEE Computer Society Press Technology Series, Los Alamitos, California, 1992.

[6] Cantú-Paz E., *A summary of Research on Parallel Genetic Algorithms*, IlliGAL Report No. 95007, Irbana, Ilinois, July 1995.

[7] Cox, E., *The great Myths of Fuzy Logic*, AI EXPERT, January 1992, pp. 40– 45.

[8] Dasgupta D., Mcgregor D., R., *Designing Application-Specific Neural Networks using the Structured Genetic Algorithms*, International Workshop on Combinations of Genetic Algorithms and Neural networks, COGAN-92, IEEE Computer Society Press, 1992.

[9] Davis, L., *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1987.

[10] Davis, L. (Ed), *Handbook of Genetic Algorithms*,Van Nostrand Reinhold, New York, 1991.

[11] Dawkins, R., *Rzeka genów* (*River Out of Eden. A Darwinian View of Life*), Wydawnictwo CIS, Oficyna Wydawnicza MOST, Warszawa, 1995.

[12] Dawkins R., *Samolubny gen* (*The Selfish Gene*), Prószyński i S-ka, Warszawa, 1996.

[13] Fogel, D.,B., *Evolving Artificial Intelligence*, A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Engineering Sciences, University of California, San Diego, 1992.

[14] Fogel L.J.: *Artificial Intelligence Through Simulated Evolution*. John Wiley, Chichester, UK, 1966

[15] Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., 1989.

[16] Goldberg, D.,E., *Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking*, Complex Systyems, 5(2), pp.139-168.

[17] Goldberg, D.E., *A Wright brothers theory of genetic algorithm flight*, ISCIe, Vol 37, No. 8, pp. 450-458, Japan, 1993.

[18] Goldberg, D.E., *The Wright brothers, genetic algorithms, and the design of complex systems*, Proceedings of SYNAPSE'93, Japan, 1993.

[19] Goldberg, D.E., *Genetic and Evolutionary Algorithms Come of Age*, Communications of the ACM, March 1994/Vol. 37, No. 3

[20] Harp S.A., Samad T., Guha A., *Towards the genetic synthesis of Neural Networks*, Proceeding of the third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[21] Heider, H., Tryba V., Mühlenfeld E., *Automatic Design Of Fuzzy Systems By Genetic* Proc. of the Fifth International Conference On Information Processing and Management Of Uncertainty In Knowledge-Based Systems, IPMU. Paris. July 1994. pp. 665-670.

[22] Hoffman, A.,*Wokó³ ewolucji*, (*About evolution*), Biblioteka Myśli Wspó³czesnej, PIW, Warszawa, Poland, 1983.

[23] Holland, J.,H., *Adaptation in Natural and Artificial Systems*, The University of Michigan, USA, 1975.

[24] Horn, J., Nafpliotis, N., *Multiobjective optimization using the niched Pareto genetic algorithm*, Technical Report No. 93006, IlliGAL, Ilinois, 1993. (http://GAL4.GE.UIUC.EDU)

[25] Kilińska M., Kwaśnicka H, *Application of Genetic Algorithm to Neural Networks Design*, XII International Conference System Science, Wrocław, Poland, 1995.

[26] Knight L., Sen S., *PLEASE: A Prototype Learning System Using Genetic Algorithms*, Proceedings of the Sixth International Conference on Genetic Algorithms (pp. 429-435), Pittsburgh, Pennsylvania USA, July 1995 (http://euler.mcs.utulsa.edu /~sandip/PLEASE.ps)

[27] Koza J.R.:*Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*. Report No. STAN-CS-90-1314, Stanford University, 1990.

[28] Koza J.R., *Future Work and Practical Applications of Genetic Programming*, Version 3, June 25, 1996, for "Handbook of Evolutionary Computation", electronical version in: http://www-cs-faculty.stanford.edu/~koza

[29] Kwaśnicka H., *Neural Network Design with Genetic Algorithm*, Conference on Computer Science, Session 26, Ostrava, Czech republic, September 5-7, 1995.

[30] Kwaśnicka, H., *The Role of Redundant Genes in Evolutionary Algorithms – Simulation Study*, Draft, Wrocław, Poland, 1996.

[31] Kwaśnicka H., *The Role of Redundant Genes in Evolutionary Algorithms – Simulation Study*, 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Berlin, August 1997, Germeny.

[32] Kwaśnicka, H., Markowska-Kaczmar, U., *Do Semi-Isolated Subpopulations Evolve Quicker? Genetic and Evolutionary Algorithms*, MENDEL'96 2nd International Mendel Conference on Genetic Algorithms, June 26-28, 1996, Brno, Czech Republic.

[33] Kwaśnicka H., Nowostawski M., *The Search Engine for Information Systems Based on Parallel Genetic Algorithm*, submitted to the 2nd International Conference on Parallel Processing & Applied Mathematics PPAM'97, Zakopane, 2-5 September 1997, Poland.

[34] Kwaśnicka H., Szerszon P., *NetGen – Evolutionary Algoriths in Designing Artificial Neural Networks*, III Conference on Neural Networks and their Applications, Kule k/Częstochowy, Poland, 14-18.10.1997.

[35] Lieske B., *Automatyczna optymalizacja sztucznych sieci neuronowych za pomocą algorytmów genetycznych* (*Automated optimization of NNs using GAs*), Master Dissertation Thesis, Department of Computer Science, Technical University of Wrocław, 1995.

[36] Mayr, E., *Speciation and Macroevolution*, Evolution 36 (6), 1982.

[37] Medsker L.,R., *Hybrid Intelligent Systems*, Kluwer Akademic Publishers, Boston/Dordrecht/London, 1995.

[38] Michalewicz Z., *Algorytmy genetyczne + struktury danych= programy ewolucyjne*, PWN, Warszawa, 1996 (translation: *Genetic Algorithms +Data Structures = Evolution Programs*, IIIed., Springer Verlag, 1966).

[39] Miller G.F., Todd P.M., Hedge S.U., *Designing Neural Networks using Genetic Algorithms*, Proceeding of the third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[40] Montana J.D., Davis L., *Training Feedforward Neural Networks Using Algorithms*, BBN Systems and Technologies Corp., Cambridge, 1989.

[41] Murray D. *Tuning Neural Networks with Genetic Algorithms*, AI Expert June 1994.

[42] Nadel, L. and Stein, D. (Ed.),*Lectures in Complex Systems*, Lecture volume V, Santa Fe Institute, Addison-Wesley Publishing Company, 1993.

[43] *NeuroForecaster – State of the Art Bussines Forecasting System, Information about the system, Futura*, Singapure, 1994

[44] Ossowski S., *Sieci Neuronowe* (*Neural Networks*), Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 1994.

[45] Petry F.,E., Buckles B.,P., Kraft D.,H., *Generating fuzzy information retrieval queries via genetic programming*, Proceedings Fifth International Fuzzy Systems Association World Congress (IFSA'93), pp. 481-484, Seoul, July 1993.

[46] *Pierwsze kroki z programem BrainMaker for DOS* (The first steps with the BrainMaker program), Arkus Electronics, Wrocław, 1994 (written on the base of "*Getting Started with BrainMaker*", California Scientific Software.

[47] Porto V.,W., Fogel D.,B., Fogel L,.J., *Alternative Neural Network Training Methods*, IEEE EXPERT, June 1995, pp. 16-21.

[48] Potter M.,A., De Jong K.,A., Grefenstette J.,J., *A Coevolutionary Approach to Learning Sequential Decision Rules*, Proceedings of the Sixth International Conference (ICGA), July 15-19, 1995, Pittsburgh, Pennsylvania USA , Morgan Kauffmann Pub.

[49] Rusoń A., *Analiza skuteczności projektowania – z wykorzystaniem AG – NN dla określonego zadania*, (*Efficiency of NN Design using GA*) Master Dissertation Thesis, Technical University of Wrocław, Wrocław, 1995.

[50] Schafer J.D., Whitley D., Eshelman L.J. *Combinations of Genetic Algorithms and Neural Networks: A Survey of the State* International Workshop on Combinations of Genetic Algorithms and neural Networks, Baltimore, Maryland 1992.

[51] Schwefell H., P., *Collective Phenomena in Evolutionary Systems*, Abteilung INFORMATIK, Universität Dortmund, Dortmund, 1989.

[52] Sen S., Knight L., *A Genetic Prototype Learner*, in the Proceedings of the International Joint Conference an Artificial Intelligence (pp. 725-731), Montreal, Canada, August 1995, (http://euler.mcs.utulsa.edu/`sandip/prototype.ps)

[53] Smith, R.E., Goldberg, D.E., *Diploidy and Dominance in Artificial Genetic Search*, Complex Systems, Vol. 6, pp. 251-285, 1992.

[54] Voigt H-M., Mühlenbein H., and Schlierkamp-Voosen D., *The Response to Selection Equation for Skew Fitness Distributions*, Proceedings of International Conference on Evolutionary Computation (ICEC'96), Nagoya, Japan, 1996, pp. 820-825.

[55] Waddington, C.,H., *Stabilization in Systems. Chreods and Epigenetic Landscapes*, Futures, April 1977.

[56] *International Workshop on Combination of Genetic Algorithms and Neural Networks*, Whitley, L.,D. and Schaffer J.,D. (Ed.), IEEE Computer Society Press Technology Series, Los Alamitos, California, 1992.