KNOWLEDGE ACQUISITION FOR INTELLIGENT CONTROL OF COOPERATIVE EVOLUTIONRY ALGORITHMS

Halina Kwaśnicka

Department of Computer Science, Wroclaw University of Technology, Wyb. Wyspiańskiego 27
50-370 Wroclaw, Poland. Tel. (48 71) 320 23 97, Fax: (48 71) 21 10 18,
E-mail: kwasnicka@ci.pwr.wroc.pl http://www.ci.pwr.wroc.pl/~kwasnick

The paper presents the project of a general optimisation tool – the intelligent system called INTEVOL. A core of the system is a family of cooperating evolutionary algorithms (**EA**s). MANAGER, a kind of Expert System, is responsible for control of **EA**s to assure effectiveness of the search process. It requires provided knowledge concerning influence of some parameters of **EA**s on tempo and mode of evolution. Such knowledge in the form of inference rules should be included into MANAGER. ANALYSER – the third part of the system, will gather knowledge concerning progress and actual state of a working system. This knowledge will be used in the form of facts to fire suitable rules. Used **EA**s have embodied pleiotropy end polygene effect, and redundant genes. To acquire knowledge needed for rules formulation the wide simulation study of proposed **EA** named the *K*-Model are made. Some results are shortly presented.

Keywords: *evolutionary algorithm, intelligent control module, optimisation*

1. Evolutionary Computation

Thinking Machines – it is not only known from decades dream of people, this paradigm of Artificial Intelligence becomes reality. We can recall here Deep Blue – the computer that won the chess game with the world master G. Kasparow, or the known "CAM-Brain" – the project managed by Hugo de Garis (Buller, 1998, http://www.hip.atr.co.jp/~degaris) and developed by an international team. The final aim of that project is to obtain an artificial brain with intelligence similar to the human level, on the base of *Cellular Automata Machine* and evolutionary growing artificial neural networks. The first phase – developing and demonstration of an artificial little cat named *Robokoneko* – should be finished at the end of 1999.

Intelligent creatures arose as an effect of biological evolution. Observing and modelling evolution allows us to obtain variety of intelligent behaviours. Computer simulations of evolution are known as *Evolutionary Computation* (**EC**). Used in such simulation algorithms are *Evolutionary Algorithms* (**EA**s).

EAs are the family of algorithms based on natural process – biological evolution. The most common classification of them distinguishes the three classes: Genetic Algorithms (**GA**s), Evolutionary Strategies (**ES**s) and Evolutionary Programming (**EP**). The fourth group, Genetic Programming (**GP**) is usually seen as a subgroup of genetic algorithms. These three main types of **EA**s (**GA**s, **ES**s, **EP**) have different historical roots and philosophy, they differ also from the technical point of view. **GA**s usually work with fixed-length binary strings and they use simple genetic operators like mutation and crossover – defined to operate in a domain-independent fashion. Not any knowledge about the phenotypic interpretation of the strings is needed. One can find in the literature a broad range of applications of **GA**s. **ES**s were developed as the way to solve some difficult real-valued parameter optimisation problems. An individual is represented as a vector of real values. In the earlier works only a mutation operator was used to introduce some perturbations into genetic material in evolving populations. **EP** is an evolutionary paradigm in which individuals are represented phenotypically as finite state machines capable of responding to environmental stimuli. They are used to optimisation, machine learning and prediction problems. Initially a mutation operator was used for structure changing.

Recently the paradigm of evolutionary computation is very popular, we observe a growing number of books, papers, conferences, home pages and discussion lists in the Internet, and so on. Simultaneously an area of **EA**s applications becomes wider, from natural for them problems such population dynamic analysis, through technical applications (designing, scheduling), games, up to economic – a popular paradigm of evolutionary economic. A variety of attempts with a wide range of different coding schemas and genetic operators appear.

EAs are quite good and skilful method for many optimising problems, but one of the main problems is ensuring the balance between exploring a wide area of potential solution space and exploiting good solutions that are present in the current population (e.g., Eiben, 1998, Goldberg, 1989, Michalewicz 1996). It is a serious problem because operators that give a good exploring effect usually cause bad exploiting, and opposite. In this paper we propose an optimising tool named INTEVOL, consisting of a family of cooperating evolutionary algorithms and two additional modules: controller (control module) and analyser (analysing module). The control module is assumed to work as an expert system, which control selected parameters of **EA**s on the base of provided knowledge. A rule base should be supplied together with the system, and it is critical for performing of the whole system. Therefore, the first step is to deliver a general scheme of the INTEVOL system and to design a knowledge base. Designing a knowledge base requires some tests – simulation study – to obtain sufficient good

and general rules. Below a new evolutionary algorithm (*K*-Model) and selected results of simulation study are presented. Verifying and pruning of the knowledge base will be done during a testing phase of the INTEVOL.

2. Some properties of the '*K*-Model' algorithm

The inspiration of INTEVOL came from the two simultaneous studied subjects. The first concerns the two classic cooperative genetic algorithms. Both **GA**s evolve individuals defined as binary strings and the simple mutation and one-point crossover operate during reproduction. The main difference lies in the probabilities of genetic operators and selection method. One **GA** – called *Searcher* has frequent mutation, not high probability of crossover, and a new population is created from the best and the worst individuals of joined parents and babies population. The second **GA** uses elitist selection, rare mutations and a high crossover probability, it plays the role of *Climber*. The *Climber* receives the best individuals from the *Searcher*. Such a simple combination of two **GA**s demonstrates meaningful advantage, it outperform the both single genetic algorithms. The second studies are connected with a new scheme of representation and including some advanced operators. Developed model and some obtained results are briefly presented in the next sections.

2.1. *K*-Model – a short description

We consider evolution of a population of $N(t)$ individuals in discrete time t (generations). Each individual is represented by its genome, a genome is a vector of chromosomes (Figure 1). Each chromosome j consists of a number np_j of phenotype genes and a number nr_j of redundant (latent) genes, each gene is an integer value. For each individual i we define a vector AG_i of his all phenotype genes: $AG_i = \langle gp^i_1, gp^i_2, ..., gp^i_{np} \rangle$, np is assumed constant number of phenotype genes. Each phene of an individual is a product of its phenotype genes and a pleiotropy matrix $\boldsymbol{A}$ (we assume a linear dependency between phenotypes genes and phenes).

Genetic operators included in the *K*-Model

Selection: We assume that a number of offspring is calculated according to the Poisson distribution. In each generation t, after calculation of fitness function Q_i of all individuals in the population, an expected number of offspring λ_i is calculated: $\lambda_i = S(t) \cdot \frac{Q_i}{Q_{av}}$, $S(t)$ – actual rate of environment saturation: $S(t) = 1 + a \cdot \frac{\mathbf{N} - N(t)}{\mathbf{N}}$, $N(t)$ is a size of a population in generation t, a – coefficient, it responses for the rate of achieving the saturation level **N**, usually $a = 1$; Q_i is a quality (fitness) value of i-th individual, Q_{av} – average quality of the population: $Q_{av} = \frac{\sum_{j=1}^{N(t)} Q_j}{N(t)}$.

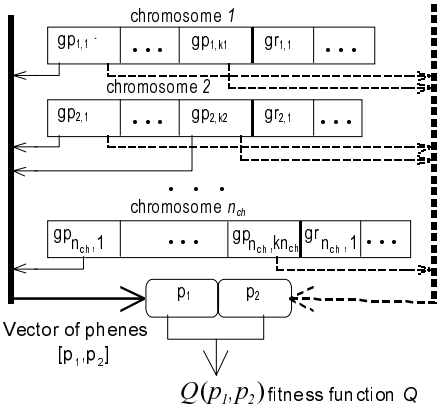

Figure 1. A general scheme of an individual representation in the *K*-Model

Recombination: Recombination operates on chromosomes. Reproduced individual can exchange its chromosome. Each chromosome can be exchanged with assumed probability p_{rkm}. The second individual for exchanging is selected randomly.

Mutation: Mutation operates on genes. It is a random change of a gene's value. Each gene mutates independently with assumed probability p_{mut}. During mutation, a mutated gene is increased or decreased about the random value from the assumed range $[-mut_{\max}, mut_{\max}]$ according to the uniform distribution. Each gene (phenotype and redundant) mutates independently.

Transposition: The transposition process acts on genes, it allows to use neutral mutation (accumulated during evolution in redundant genes). Obviously, the redundant gene has to be phenotype one to influence the individual fitness. Described process allows for such modification. With assumed probability p_{trs} two genes, one active and one redundant, exchange their places, so, the redundant gene becomes the active one, and inversely.

Transition: This process acts on genes, it causes that a gene from one individual is copied to the second, randomly chosen individual. Each gene can be copied and it can be put on a random chromosome of randomly selected individual, but only as the redundant gene. The probability p_{trz} is given as a model's parameter.

Specialised operators for macromutation modelling

Redundancy: By the accumulation of genes' changes caused by neutral mutation (in the part of chromosomes with redundant genes) significant changes in the individual's genetic material can appear.

Recrudenscence[1]: The probabilities of described earlier genetic operators (p_{rkm}, p_{mut}, p_{trs}, and p_{trz}) are increased for the part of population. It occurs during each generation of evolution. Individuals for the recrudescence are selected randomly with uniform distribution.

Crisis: It acts as recrudescence, but all individuals are the subjects of a crisis after each t_{kr} generations.

[1] recrudescence from Latin *recrudesco* and *recrudescere* – a new outbreak after a period of abatement, inactivity or after a dormant period.

2.2. Some results of simulation study

The *K*-Model and results of its simulation study are described in (Kwaśnicka, 1999a, 1999). Here we present only a small part of them. The main aim of the study is to investigate the possibilities of escaping from local optima. The two aspects of evolution are interesting for us. The first is finding the optimal solution – it means that evolution stops with success if a single, acceptable solution (optimal or near optimal) is found. A population as a whole can occupy a lower peak, but one good solution is enough for the user. Evolution is treated as a pure optimisation tool. The tempo of evolution measured a number of generations is significant. The second aspect is analysis of dynamic of an evolving population, so, a mode of evolution is important. It means that we observe a character of evolution: diversity of the population, a character of changes the best and the average fitness in the population (gradual or not), losing good solutions, concentration of individuals in the high peak, etc. In our experiments we use a family of exponential functions:

$$Q(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{l_s} h_1 \cdot e^{-n_i \cdot \sum_{j=1}^{m} (x_j - x_j^i)^2},$$

where: x_i – j-th coordinate of fitness function Q (j = 1, ..., m, m – a dimension of a fitness function), x_j^i – value of j-th coordinate of i-th peak (i = 1, ..., l_s, l_s – a number of peaks), h_i – height of i-th peak, n_i – slope of i-th peak.

This general function is chosen taking into account possibilities of modifying a number of variables, a number of peaks, their positions, slopes and highs. Majority of experiments are made with the two functions:
Q_1 with two peaks and two variables: $h_1 = 1$, in the point $\mathbf{x} = [5, 5]$, $h_2 = 1.5$, in $\mathbf{x} = [20, 20]$, $n_1 = n_2 = 0.02$;
Q_2 with two peaks and five variables: $h_1 = 1$, in the point $\mathbf{x} = [5, 5, 5, 5, 5]$, $n_1 = 0.02$, $h_2 = 1.5$, in the point $\mathbf{x} = [20, 20, 20, 20, 20]$, $n_1 = 0.00225$.
Other multimodal functions including 10- and 30-dimension Q functions and the De Jong test functions (Goldberg, 1989) are also used in the simulations. In all experiments with the Q functions, an initial population was placed on the lowest peak.

Macromutations in the K-Model

A solution with fitness value greater than 1.35 is seen as a success of evolution. Values of all genes in an initial population: 1000, all phenes: 5. All experiments were repeated minimum ten times.

The role of the redundancy

1. Function Q_2, $mut_{\max} = 10$. Without redundancy, population cannot get the higher peak, it evolves around the lower peak independently on the probabilities p_{rkm} and p_{mut}. The same result is observed with redundancy, independently of p_{trs} and p_{trz}.
Summary: Such small maximal single mutation change unable evolution to find the global optimum of Q_2.

2. Function Q_2, $mut_{\max} = 100$, $p_{\text{rkm}} = 0.25$, $p_{\text{mut}} = 0{,}2$, p_{trs} and p_{trz} are changed from 0.1 to 1. Evolution goes without success. Increasing recombination and mutation probabilities allows population to reach the higher peak. It occurs when $p_{\text{rkm}} = 0.5$, $p_{\text{mut}} = 1$, $p_{\text{trs}} = p_{\text{trz}} = 0.2$, after 3300 generations. With so high recombination and mutation population loses the good solution and cannot occupy the global optimum.
Summary: Without redundant genes, population is not able to find the global optimum. With redundancy and high p_{rkm} and p_{mut}, population finds a relatively good solution, but cannot reach a high average fitness. The best solutions appear in the best area of phenotype space, but population is not able to exploit them. We can say, that our model can explore potential solutions space but it cannot exploit them. Frequent recombination and mutation cause that the diversity of the population is high.

3. Function Q_2, $mut_{\max} = 500$, $p_{\text{rkm}} = 0.25$, $p_{\text{mut}} = 0{,}2$, $p_{\text{trs}} = p_{\text{trz}} = 0.2$. Population needs about 4500-5500 generations to get the global optimum.
Summary: With greater p_{trs} and p_{trz} population usually find the best solution quicker, but the average quality is low (compare Figure 2 and Figure 3). The $mut_{\max}$ parameter is significant especially for multidimensional, multimodal functions. Greater $mut_{\max}$, similarly as greater p_{trs} and p_{trz}, causes increasing diversity in the population and facilitates discovering good solution.

4. Experiments with 2-dimensional function Q_1. For evolving population this function is easier then Q_2. For $mut_{\max} = 10$, $p_{\text{rkm}} = 0.25$ and $p_{\text{mut}} = 0.2$, without redundancy population cannot find global solution during 1000 generations. Adding redundant genes ($p_{\text{trs}} = p_{\text{trz}} = 0.05$) causes that in all ten simulation runs, 439 to 824 generations were enough to reach the higher peak.
Summary: Evolution with a relatively small $mut_{\max}$ and macromutation in the form of redundancy is able to find the global optimum when a dimension of a fitness function is low.

The role of the recrudenscence

5. Function Q_2, $mut_{\max} = 10, 100, 500$; $p_{\text{rkm}} = 0.25, 0.5, 0.8, 1$; $p_{\text{mut}} = 0.2, 0.5, 0.8, 1$; $rec = 10\%, 50\%$, without redundancy evolution fails. In all 10 runs with above parameters, evolution stops without success (after assumed 5000 generations).
Summary: We can say that under those conditions – high all parameters of genetic operators but no redundancy – the recrudescence is not able to assure efficiency of evolution.

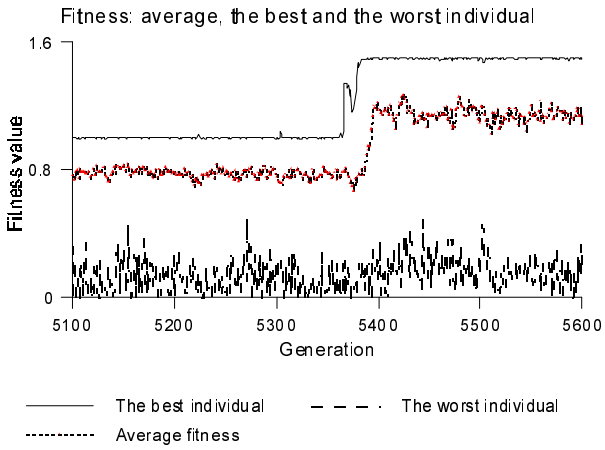

Figure 2. Average fitness and fitness of the best and the worst individuals during evolution with parameters: mut_{max}=500, p_{rkm}=0.25, p_{mut}=0.2, p_{trs}=p_{trz}=0,01

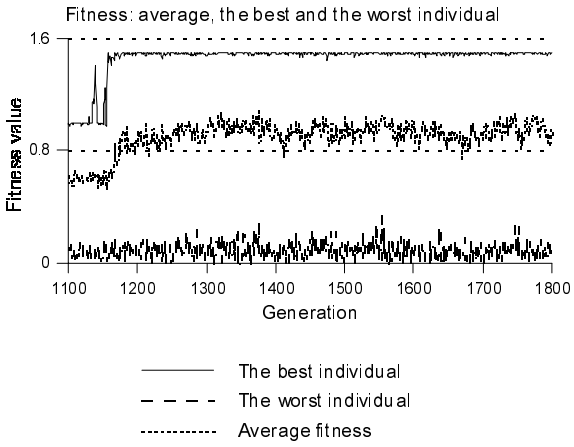

Figure 3. Average fitness and fitness of the best and the worst individuals during evolution with parameters: mut_{max}=500, p_{rkm}=0.25, p_{mut}=0.2, p_{trs}=p_{trz}=0,05

The role of the crisis

6. Function Q_2. For based operators mut_{max} = 100, p_{rkm} = 0.25, p_{mut} = 0,2, we add the crisis. We assume t_{kr} = 20, and 50, and in the crisis the probabilities are enlarged to p_{rkm} = 0.25, 0.5, 0.8, 1, p_{mut} = 0,2, 0.5, 0.8, 1.
Summary: All simulation runs failed. Redundant genes were absent in these experiments. It seems that the role of redundant genes is significant, they enable population to settle the best fitness niche.

Effects of joined redundancy, recrudescence and crisis

7. Function Q_2, mut_{max} = 200, p_{rkm} = 0.25, p_{mut} = 0.2, p_{trs} = p_{trz} = 0.1, *rec* = 5%, t_{kr} = 20 (for individuals under the recrudescence and during crisis: p_{rkm} = 0.5, p_{mut} = 0.2, p_{trs} = p_{trz} = 0.2).
Summary: Success of evolution is possible during about 10000 generations. Average quality of the population is higher than the lower peak and we observe short fluctuations of average fitness. Fluctuations are caused by the crisis, when the population explores larger area of phenotype space (Figure 4, 5). After finding the better fitness niche population needs about 20 generations to achieve average fitness higher then the high of the lower peak.

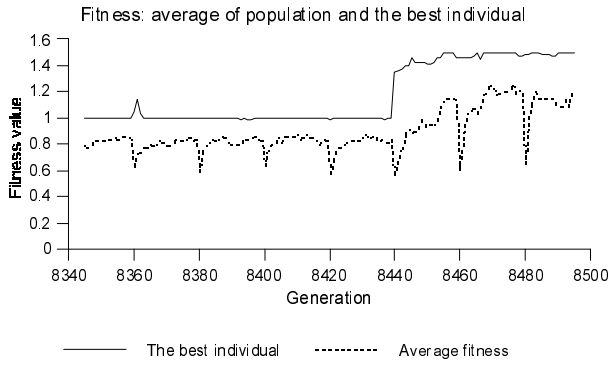

Figure 4. Average fitness and the best individual fitness: mut_{max}=200, p_{rkm}=0.25, p_{mut}=0.2, p_{trs}=p_{trz}=0.1, *rec*=5%, t_{kr}=20

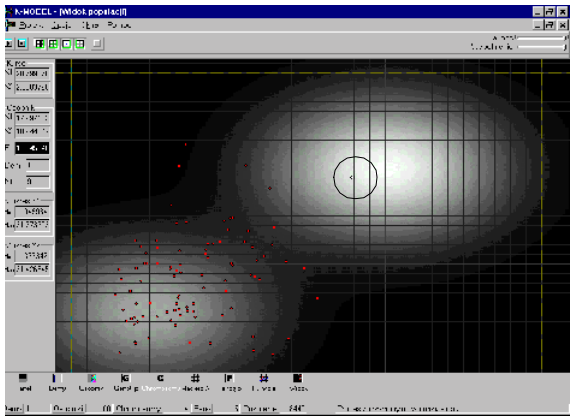

Figure 5. Generation 8440, parameters as in Figure 6, the first individual is seen on the higher peak (in circle)

8. Function Q_1. We apply redundancy with crisis and redundancy with recrudescence.
Summary: More effective is evolution with crisis than with recrudescence. Adding crisis causes that evolution goes quicker about 15%, but we have not observed this using recrudescence instead of crisis. We must remember that crisis is a form of macromutation, it enables population to test wide area of phenotype space, but on the other hand, it decreasing average fitness of population.

Population size in the K-Model

Numbers of experiments show that for Q_2 function evolution can go with success with following parameters: mut_{max} = 600, p_{rkm} = 0.25, p_{mut} = 0.1, p_{trs} = p_{trz} = 0.05. These parameters' values suit as the base for further experiments with this function. Assumed number of generations is 2000, we have evolved population consists of 10, 100 and 100 individuals. The results are in Table 1. The results surprise us, we see that smaller populations evolve quicker then the larger ones. Let us see the changes of average qualities of above populations (Figure 6). Population of 1000 individuals evolves gradually, after finding a good solution the population goes to the higher peak (Figure 7). The average quality rises but it is not high because of macromutations. Different situation we can see in Figure 8. Observed fluctuations are rather big. Evolution of such population is similar to the random

walk through the phenotype space. Other experiments show that such population is able to find quickly the highest peak when we have a multimodal fitness function (e.g., with four peaks).
Summary: Small population can walk through the valley, and find a good area, but it cannot persist in this place.

Experiments with other function and parameters were made, for example to test the influence of redundancy and dimension of fitness function (10 and 30 variables). For Q_1 function, without redundancy, and $mut_{\max} = 20$, $p_{\text{rkm}} = 0.25$, $p_{\text{mut}} = 0.2$, we obtained quite different results – 1000 individuals need from 314 to 1879 generations, 100 individuals – from 1532 to 22935 generations (about 6000 in average), and 10 individuals cannot reach global optimum. Adding redundancy ($p_{\text{trs}} = p_{\text{trz}} = 0.05$) changes above results: population with 1000 individuals needs 42-205 generations, 100 individuals: 151-439 generations, 10 individuals: 81 to 121 generations to reach optimum.

Table 1. Results of evolution with Q_2 and different size of population

Population's size	Number of generations after which a solution with a quality grater than 1.35 is achieved in particular simulation run									
	1	2	3	4	5	6	7	8	9	10
10	649	652	532	548	681	586	697	462	348	1330
100	1693	1041	1170	1132	1005	1380	1021	1269	1514	1330
1000	–	2598	2266	–	2049	–	2438	–	–	3863

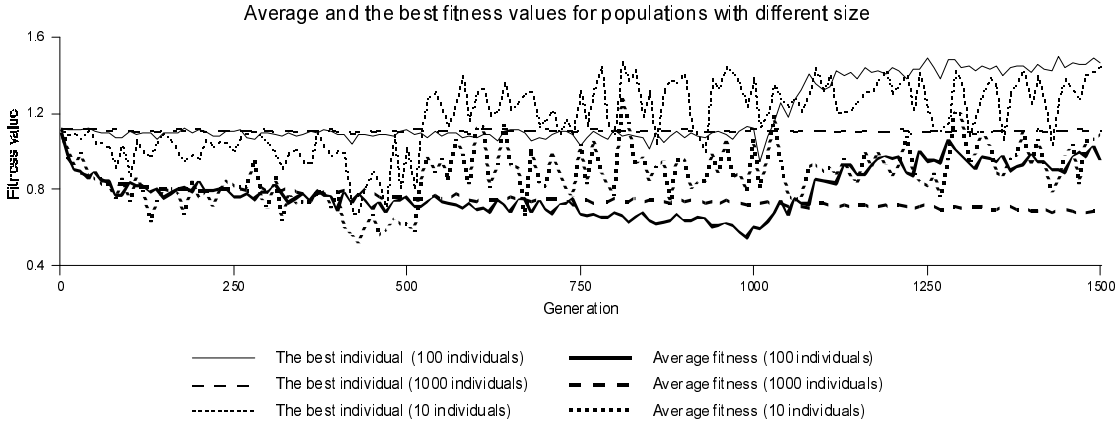

Figure 6. Changes of the best and average qualities in the population depending on the populations' size, the coordinate 'Generation' is scaled after each 10 generations (evolution of 1000 individuals failed)

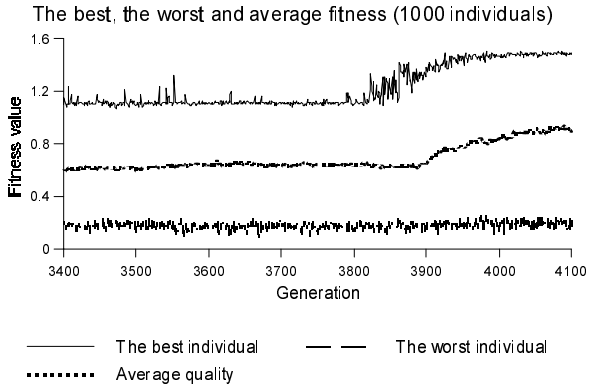

Figure 7. Fitness values of population consists of 1000 individuals (the best, the worst individual, and the average population's fitness)

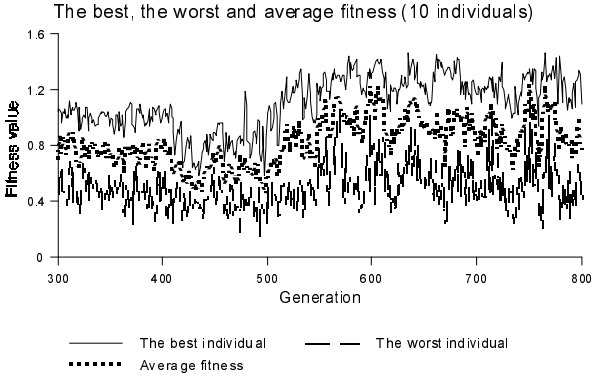

Figure 8. Fitness values of population consists of 10 individuals (the best, the worst individual, and the average population's fitness)

We have used function with 30 variables and two peaks, setting parameters as follow: $mut_{\max} = 600$, $p_{\text{rkm}} = 0.25$, $p_{\text{mut}} = 0.2$, $p_{\text{trs}} = p_{\text{trz}} = 0.05$. All populations evolve with success (1000 individuals: 3181 to 3543 generations, 100 individuals: 3422-3775 generations, 10 individuals: 4505-6854 generations).
Summary: The smallest populations usually reach place near the global optimum quickly, but it cannot find the top of the peak. We see that the dimension of fitness function changes the tempo of evolution, taking into account the size of a population. Population of 10 individuals walks more randomly then the bigger ones. Therefore the small populations are more effective for smaller dimension fitness functions. Redundancy of genotypes gives a population possibilities to test wide area of phenotype space. This effect is seen during evolution of all populations but the redundancy hardly influences smaller populations.

3. INTEVOL – a general scheme

The final INTEVOL system is planned to be an optimisation tool for variety of tasks. It is a hybrid system, consists of a small Expert System (called MANAGER), cooperating evolutionary algorithms (FAMILY OF **EA**S), and the ANALYSER module – it collects information about effectiveness of evolution and produces facts needed by the MANAGER (Figure 9). An inference engine in the MANAGER processes rules, and produces required changes in the FAMILY OF **EA**S, for example a number and sizes of $\mathbf{EA}_i$, parameters of their genetic operators.

Expert Systems are consulting computer programs that aid decisions making and substitute domain experts. They require good domain knowledge, usually given in the form of inference rules: **if** ⟨circumstances⟩ **then** ⟨do actions, or conclude something⟩. Collected in the set of rules domain knowledge (rule base) together with given knowing true facts (base of facts) constitutes a *knowledge base*. An inference engine performs knowledge base: it fires these rules which circumstances are true and adds concluded facts to the base of facts or performs concluded actions. Typical expert systems contain also a module of knowledge acquisition – it enables modifying a knowledge base, and a user interface with an explanation module – it is very significant in the test phase, and it allows the system to be in confidence in end users.

ANALYSER should observe performance of **EA**s and collect gathered knowledge concerning a tempo of solutions' improvement, actual diversity in **EA**s, similarities between good solutions, etc. On the base of analysis of this knowledge some parameters (facts) must be generated and provided to the MANAGER, where they are used by its inference engine to produce appropriate control.

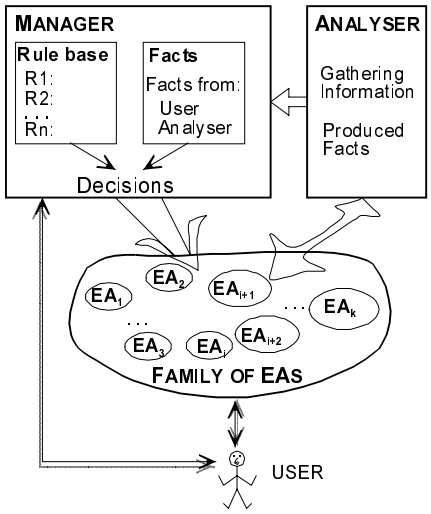

Figure 9. The general schema of the **INTEVOL** system

4. Summary

The prototype of INTEVOL system should enable its wide investigation, especially pruning and verifying the knowledge on the base of tests with different tasks. We assume that end users of the system have no experience with evolutionary algorithms, so they have difficulties with tuning parameters of an **EA**. Therefore MANAGER and ANALYSER ought to relieve users of this duty. The system can need simple information from the user, for example, does his task belongs to the class of sequential problems. A user must be asked to define his task in the term of evolutionary algorithms and – optionally – to change conjectured parameters' values on the predefined sheet. Designed system must be user friendly.

Wide simulation studies have been made to gather base for preparing a set of rules for MANAGER. One of the interesting results is that concerning the optimal division of a whole population for evolving demes: a group of small demes with one bigger seems to be the most effective. Actually we are defining and coding of rules.

References

Buller A. (1998), *Sztuczny mózg, to już nie fantazje* (*An artificial brain – it is not fantasy*), Prószyński i S-ka, 1998.

De Jong K. (1998), *Evolutionary Computation: Where We Are and Where We're Headed*, Fundamenta Informaticae, Vol. 35, No. 1-4, August 1998, pp. 247-259.

Eiben A.,E., Schippers C.A. (1998), *On Evolutionary Exploration and Exploitation*, Fundamenta Informaticae, Vol. 35, No. 1-4, August 1998, pp. 35-50.

Fogel D.,B. (1998), *Unearthing a Fossil from the History of Evolutionary Computation*, Fundamenta Informaticae, Vol. 35, No. 1-4, August 1998, pp. 1-16.

Fogel L.J. (1996) *Artificial Intelligence Through Simulated Evolution*, John Wiley, Chichester, UK.

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

Holland J.H., (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan, USA.

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

Kwaśnicka H. (1997), *Redundancy of Genotypes as the Way for Some Advanced Operators in Evolutionary Algorithms – Simulation Study*, VIVEK A Quarterly in Artificial Intelligence, National Centre for Software Technology, Mumbai, Indie, Vol. 10, No. 3, July 1997.

Kwaśnicka H. (1998), *Evolutionary Computation in Applications*, The invited lecture at the Seventh International Colloquium on Numerical Analysis and Computer Science with Applications, (accessible in the http://www.ci.pwr.wroc.pl/~kwasnick), August 13-17, 1998, Plovdiv, Bulgaria.

Kwaśnicka H. (1999), *K-MODEL – An Evolutionary Algorithm With New Schema of Representation*, accepted to appear in the Proceedings of CIMAF'99 Symposium on Artificial Intelligence, 22-26 March 1999, Havana, Cuba.

Kwaśnicka H. (1999a), *Obliczenia ewolucyjne w sztucznej inteligencji* (*Evolutionary Computation in Artificial Intelligence*, in Polish), to appear in Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 1999.

Michalewicz Z. (1996), *Algorytmy genetyczne + struktury danych = programy ewolucyjne* (*Genetic Algorithms + Data structure = Evolution Programs*), WNT, Warszawa.

Seredynski F. (1998), *New Trends in Parallel and Distributed Evolutionary Computing*, Fundamenta Informaticae, Vol. 35, No. 1-4, August 1998, pp. 211-230.