Evolutionary Algorithms for Optimization of Badly Structured or Unknown Functions like Inventory Costs for example

by

JENS ARNOLD

Faculty of Informatics
Technical University of Chemnitz-Zwickau
09107 Chemnitz, Germany
E-Mail: jarn@informatik.tu-chemnitz.de

Abstract

Evolutionary algorithms are an optimization methodology based on a direct analogy to Darwinian natural selection, recombination and mutation in biological reproduction. The evolution process thus converges to individuals with an optimal fitness to the considered environment. Both variants genetic algorithms (GA) and evolution strategies (ES) are shown. GA are quite useful for solving of complex discrete optimization problems like sequencing or scheduling problems. On the other hand ES provide a powerful solving of complex problems with more or less continuously changeable parameters, e.g. current optimization, vector or parameter optimization in general. The two variants differ also by local and global convergence, search accuracy and speed.
A simple TURBO PASCAL® programmed software tool for evolutionary optimization is presented. It works with a simulation tool for a multi-location inventory model with redistribution for example to find the minimum of the unknown convex function of inventory costs.

1. Introduction

The optimization of production systems, manufacturing processes, transport and logistic problems means at present: An integral combination of quick simulators with efficient optimization tools. The classic optimization methods like the gradient method, the dynamic programming or any kind of heuristics come across limits very often concerning runtime and complexity. Therefore, we need optimization algorithms, which change automatically the input of parameters for the next simulation run by considering the user defined objectives and the results of the past runs.
Evolutionary algorithms are an optimization methodology based on a direct analogy to Darwinian natural selection, recombinations and mutations in biological reproduction. The evolution process thus converges to individuals with an optimal fitness to the considered environment. The main advantages of these algorithms lie in great robustness, problem independence and high parallel working. So far, evolutionary algorithms were most successful in parameter optimization domains. However, even there are certain problems, as lack of final tuning capabilities and severe time complexity, prohibit their wider use on most moderately and highly complex problems.
The aim of this paper is to give an overview on the development of evolutionary algorithms, beginning with the used terms and concepts, up to the basic genetic operators and stopping at a simulation-based genetic optimizer for production and logistic used nowadays on single processor computers.

2. What can we learn from nature ?

I will give the main answer to this question at first: A general disorder is no chaos, but more likely a special case of order. We need this to understand the working and philosophy of evolution. There must be a functional connection between the state and the parameters of a (natural or artificial) system and its performance. Our well known mathematical description of such an optimization problem follows shortly:

(1) $P = \{x \in W : a_i \le x_i \le b_i \, , \, i=1(1)l\}$

Where P is the *parameter space*, which characterizes the investigated system. It contains all *parameter vectors* x of the set W satisfying the restrictions of all l *components* x_i.
Notice: The components x_i can be of different type (binary, integer, real, ...).

Furthermore, an *objective function* is defined

(2) $f(x) : P \Rightarrow R^1$

as a mapping from the parameter space P to the set of real numbers R^1.
Now we look for

(3) $x^* \in P \;\; \text{with} \;\; f^* = f(x^*) \ge f(x) \;\; \text{for all} \;\; x \in P$,

a global maximum. If we model a minimization problem, only the greater-or-equal symbol will be inverted. The main difficulty is that the required objective function (point (2)) is unknown or only known partly in the most of the cases. The performance of the system defined by this function can only statistically approximated by simulation:

(4) $f(x) \approx f_{Sim}(x,t)$

Where t is the time of observing the system, that means the simulation time of the computer internal model at a given parameter vector. The length of this period has a great impact on the accuracy of approximation from $f_{Sim}(x,t)$ to $f(x)$.
In general

(5) $\lim_{t \to \infty} f_{Sim}(x,t) = f(x).$

Now it is easy to show how the biological objects are in touch with these mathematical terms.

mathematical view		*biological view*
· parameter space P of all possible solutions (set of all system states)	≡	universal population
· subset of solutions $G \subset P$	≡	generation (population at a specific time)
· one solution $X \in G$ (one system state)	≡	individual
· one vector of parameters $x \in X$	≡	chromosome
· one component of x	≡	gene
· the value of such a component	≡	allele
· objective function	≡	fitness function

All that means we have to study the mechanisms of natural evolution and the genetic code. Afterwards we can create mathematical models of genetic operators according to these biological processes. To say it in brief, from mathematicians, computer scientists and engineers point of view, the evolution is an extrem efficient optimization method. We will get much knowledge about the behaviour of these genetic operators and the whole evolution by the investigation of our mathematical models of evolution. There are two different important results of our investigation, maybe:

(I) We can get a feedback to evolutionary issues and theories which is very interesting for biologists and gene scientists who study evolutionary phenomena.

(II) We can apply the evolution process to every kind of artifical or social-economic systems.

The first one is another task and not the topic of this paper (look into [19] or [29] for example). The second point should noticed from all readers who are interested in optimization in general.

3. The basic Evolutionary Algorithm

The optimization process with evolutionary algorithms is underlay by an iterative calculation scheme. This scheme is adaptable to specific problems by variable parameters, but it has to follow always these steps:

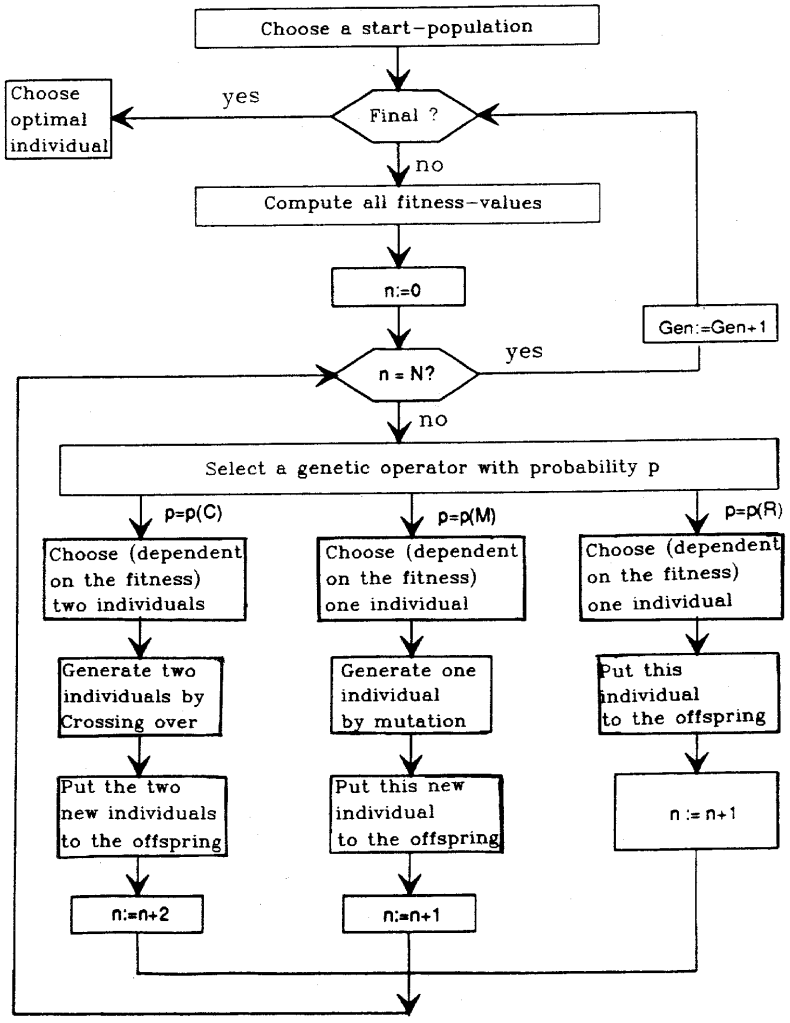

Fig. 1: The basic evolutionary algorithm (from [20]).

At first we have to choose a start- (or initial) population G_0 by selecting N individuals out of P. We can do this by mere chance (uniformly distributed) or, if we know some characteristics of the fitness function, we should take the initial individuals (solutions of our parameter space) from the vicinity of the optimum. The example at the end of this paper shows you that we can find this start-population exemplary by computing the best stocking levels for stocks without any shipment between each other.
There are many different suggestions for the right size N of the population. We must face the fact that it is nearly always important to use as small a population as possible, because the total number of fitness evaluations (N•G, where G is the number of investigated generations) should be small. This demand is necessary since the time of fitness evaluation for one individual by simulation is much more longer then all the time of creating the next generation. The values stating in literature (N = 30 [26], N = 50 [29], N > 50 [20]) have proved themselves time and again with many experiments. At least I want to give a formula suggested by [26]:

$$N \approx 1.65 \bullet 2^{0.21 \bullet l} \tag{6}$$

Where l is the number of genes (dimensions of our parameter space P), that means the length of a chromosome.
If you look for a good set of parameters to create your own evolution for solving a specific optimization, you will find them in [15]. It is very time expensive to get a suitable and acceptable parameter set, because all parameters (like population size N, number of genes l, number of generations G, probabilities of recombination p(C), mutation p(M) and reproduction p(R), e.g.) influence each other. But do not worry! The evolutionary algorithms are very sturdy against little changes of their parameters.

Let us come back to the basic algorithm. After choosing the start-population we will have to answer the Final? - question. The nature does not give us any answer to this important question! Remember, the natural evolution is a (possibly infinite) continuous process without any recognizable final aim. The reason lies in the sense of evolution: Adaptation of the beings to environmental conditions with permanent change. But our investigated artificial systems have a concrete defined, functional behaviour inside a more or less known "environment". Therefore, the search process should stop after an/the optimum was found. How can we get this?
There are three possibilities to break off the (mathematical) evolution:

(a) The value of the optimum is known:

That is the easiest case. The evolution terminates when a chromosome x was found which satisfies the fitness-value f^* with a wanted accuracy. The accuracy had to be determined before the evolution starts.

(b) The value of the optimum is unknown:

The evolution terminates when there is no essential improvement of the fitness-value for the best individual over some generations. For that we must define an *epsilon*-vicinity around f(x). All the best fitness-values inside this vicinity ε observed over some generations are regarded of equal value.

(c) The maximum number of generations is set from the start:

A maximum number of generations is set before the evolution process will start. The following figure 2 shows an estimation of how many percent of the genes (variables) will be engaged by optimal alleles (values) dependent on the size of population N and the generation G.

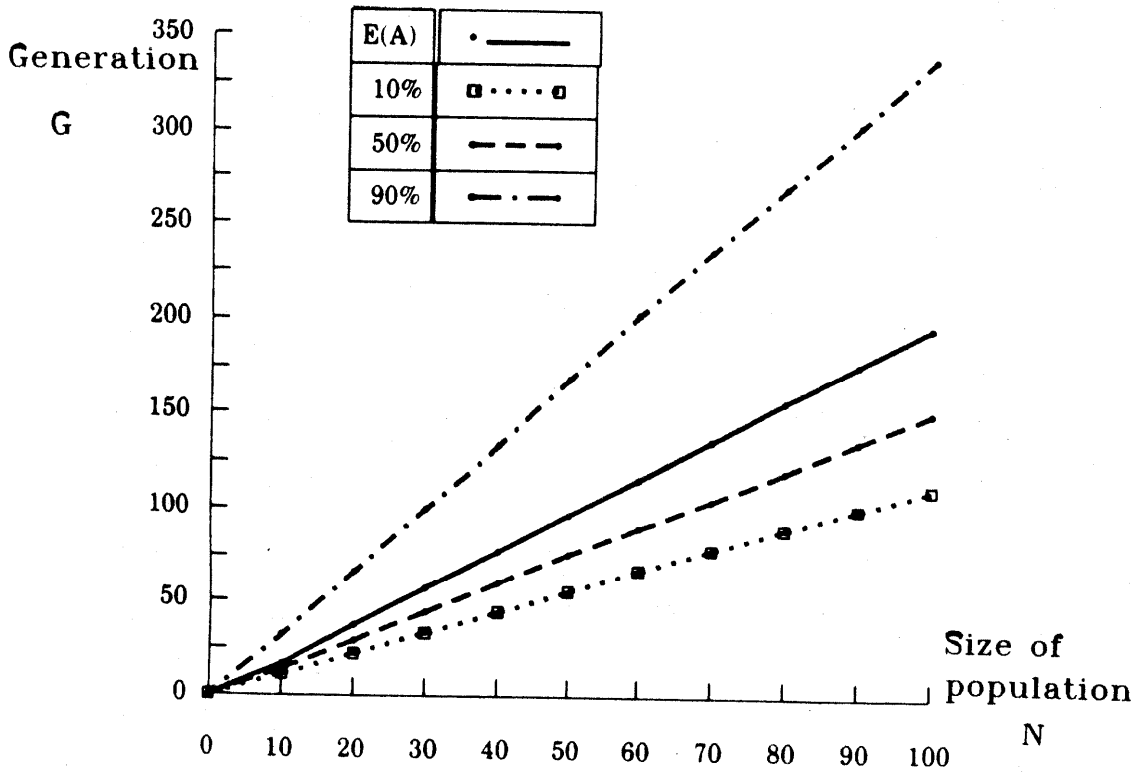

Fig. 2: Estimated percent-rate of optimal alleles E(A) dependent on the size of population N and the number of generations G (from [14])

The values of figure 2 apply only for well chosen parameters. That means if the offspring (population of children) has fewer individuals than N, we will have to select a genetic operator carefully. The probablities for the three main operators are:

- p(C) for recombination (crossing over)
- p(M) for mutation
- p(R) for reproduction

It follows:

- $p(C) + p(M) + p(R) = 1$

Maybe, we can define other new and also non-biological operators like inversion, cut- and splice, translocation for instance. There are no limits for our scientific inventivness. But notice, all these operators have to go together and to go with the optimization problem's genetic representation! This paper restricts oneself to only the three main operators mentioned above.
Each of these genetic operators needs one, two or more individuals (values) for working. These individuals are chosen from the parent population by the so-called *selection*-procedure. There are a lot of different selection methods like the *roulette*, *linear ranking*, *tournament*, *(N,μ)*-selection. Every kind of selection must prefer individuals with a good fitness to such one with a worse. Weak individuals ought to get a though little chance to pass their alleles to the next generation yet. This is very important for the spread out of the individuals over the parameter space. Otherwise the evolution will make a premature convergence to a local optimum, maybe.

4. Genetic Algorithms vs. Evolution Strategies

The idea to use principles of organic evolution processes as rules for optimum seeking procedures emerged independently on both sides of the Atlantic ocean more than three decades ago. Both approaches rely upon imitating the collective learning paradigm of natural populations, based upon Darwinian's observations and the modern synthetic theory of evolution.
In the USA JOHN H. HOLLAND introduced genetic algorithms (GA) in the sixties, embedded into the general framework of adaption (see [16]). GA work on a genotypic level. They try to imitate the biological evolution by copying the genetic code (countable allele-alphabet) and the main operator *crossing over*. Notice: A transformation to the allele-alphabet is necessary for the optimization of real-valued problems with GA! Its selection is based upon a probabilistic principle. Therefore, the pressure to survive is not very high on the individuals.
The evolution strategies (ES) were evolved by INGO RECHENBERG in Berlin (see [25]) at the same time as well, and further developed by HANS-PAUL SCHWEFEL at the university of Dortmund (see [31]-[34] and [38]). ES work on a phenotypic level, i.e. they operate directly on the set of real-valued object variable x_i. The main operator *mutation* is realized by adding to each x_i a normally distributed random number with expected value 0 and standard deviation σ_i. The so-called strategy variables σ_i are stored in an additional vector with length l.
Theoretical considerations for a maximum rate of convergence suggest that the optimal settings of the σ_i may depend on the distance from the optimum, i.e. they are a local feature of the response surface. Therefore, the genetic information of each individual not only consists of the x_i, but also of the strategy parameters σ_i which also undergo mutation and recombination before they are used to mutate the x_i. ES' selection is based upon a determined principle. That means the μ parents reproduce a set of Γ children. The μ best of the either Γ or $\mu+\Gamma$ individuals will survive. It follows from this that the pressure to survive on the individuals is normally higher than by the GA.
The proportion of publications until today can estimated at 100 : 1 for the GA. This is due to the unambiguous preference and the worldwide attention of research and high-tech-applications in the USA, and not to the superiority of GA (see also [29])!
If you have to take a decision on which kind of evolutionary algorithm you should use to optimize a specific problem, I will give a rule of thumb: Optimization problems which come to a discrete optimization are more suitable for GA. On the other hand ES are favourable to problems with more or less continuously changeable parameters and all problems of non-discrete optimization. The following table shows some good examples which were optimized by evolutionary algorithms successfully:

Genetic Algorithms	*Evolution Strategies*
- travelling salesman problem	- optimization of body shapes
- partitioning problems	- configuration problems
- scheduling problems	- function optimization in general
- selection problems	- socio-economic systems

To show you all the genetic operators, selection procedures, suitable parameter sets and coding methods for both GA and ES would take to much place here. Hence, I added an extensive reference list to the end of the paper where you will find many good books and papers for a more detailed study and further reseach. If you are interested in making your own experiences with evolutionary algorithms, please write to me or my e-mail address. There is a public-domain software tool called EVAOPT available. It is written in TURBO PASCAL® and works over a file-interface together with a simulator programmable by yourself. This program was special designed for the optimization of production-, transport- and logistic systems. You can get the source code, the compiled version and a short description to make all the changes you ever want and to develop it further.
The applications described next section are typical of the use of EVAOPT. Nevertheless, little has been published that is relevant to really existing production-, transport- and logistic systems. The recent availability of parallel high-performance computers pushes open the door for the evolutionary algorithms to such new fields of application.

5. The 3 Basic Problems of Production, Transport and Logistic

The optimization of problems mentioned in the headline of this section demands a high level of flexibility, because we cannot know all of the possible systems (which we want to investigate) before the designing of our optimization tool will start. Therefore, the optimization tool has to make available such components which are generally valid enough to model all that is in of the question. Most of the classic optimization methods have problems to give such a flexibility. They work only on a small subset of tasks. But not so the evolutionary algorithms. For the time being I will introduce the three (only) basic problems and their assigned evolutionary algorithms. Afterthat we will put together an individual of these three types of chromosomes up to a whole population ready for evolution.
The three basic tasks are similar to the three combinatorical problems like you will see.

(I) Sequencing problems (permutation):

There must be a defined order over a set M of elements (like jobs in figure 3). That means if element 1 stays before element 2 then element 1 will have done before element 2 will done. The present discrete optimization problem is easy to recognize by permutation over the set M. Therefore, this basic problem is suitable for genetic algorithms. The GA has the duty to generate a set of permutations for each generation by considering the fitness values of the parent.

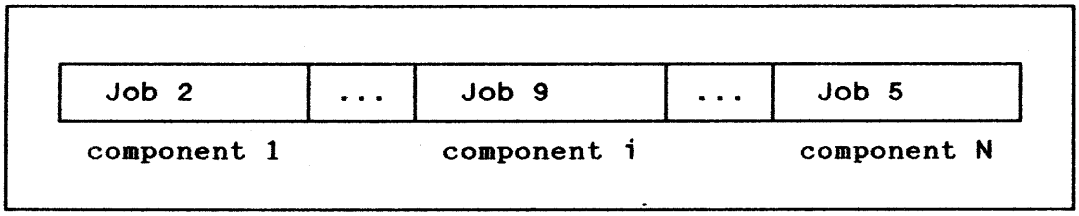

Fig. 3: Example of the chromosome type *sequencing*

Suitable applications are scheduling, job-order or transport problems like the travelling salesman problem for instance.

(II) Selecting problems (combination):

We search for a real subset S of the set M by that way which gives us the best fitness value for the subset S. There are problems which allow a repeated selection of an element and otherwise which demand an exclusive selection of every element. The present discrete optimization problem is easy to recognize by combination either with or without repetition on the set M. Therefore, this basic problem is suitable for genetic algorithms. The GA has the duty to generate a set of combinations for each generation by considering the fitness values of the parent.

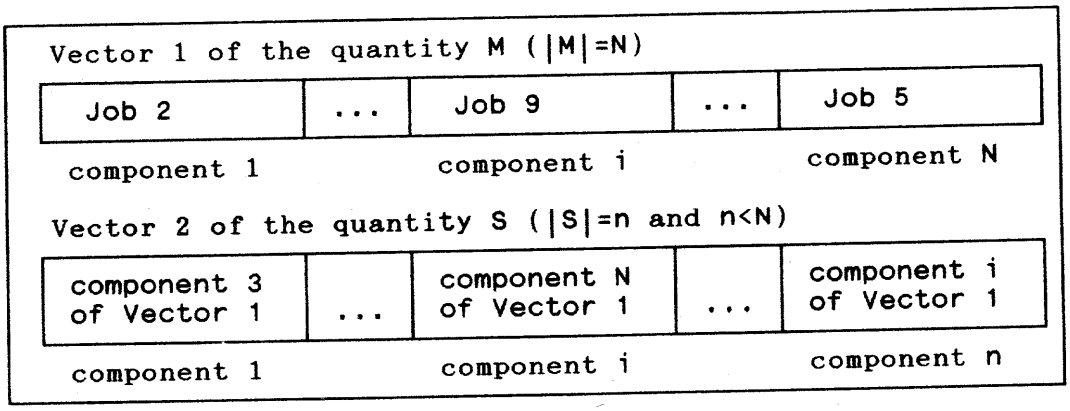

Fig. 4: Example of the chromosome type *selecting*

The most frequent application is the occupation of resources like machines, pallets or stocks at a specific time.

(III) Assigning problems (variation):

To every element e of a defined set A will assigned an element of a set M by that way which gives the best fitness value for the set A. The present optimization problem is easy to recognize by variation. If the set M is not countable, the optimization problem will be non-discrete. Since this is the most frequent case an evolution strategy is suitable to solve such a problem. Incidentally, it is no problem for the algorithm to work with an countable set M. Like you can see in figure 5, we have to define the set M with a lower and upper bound and a step for each element of the set A at first. You will get an optimizer of real-valued functions for setting step = 0, like the most objective functions of inventory costs too.

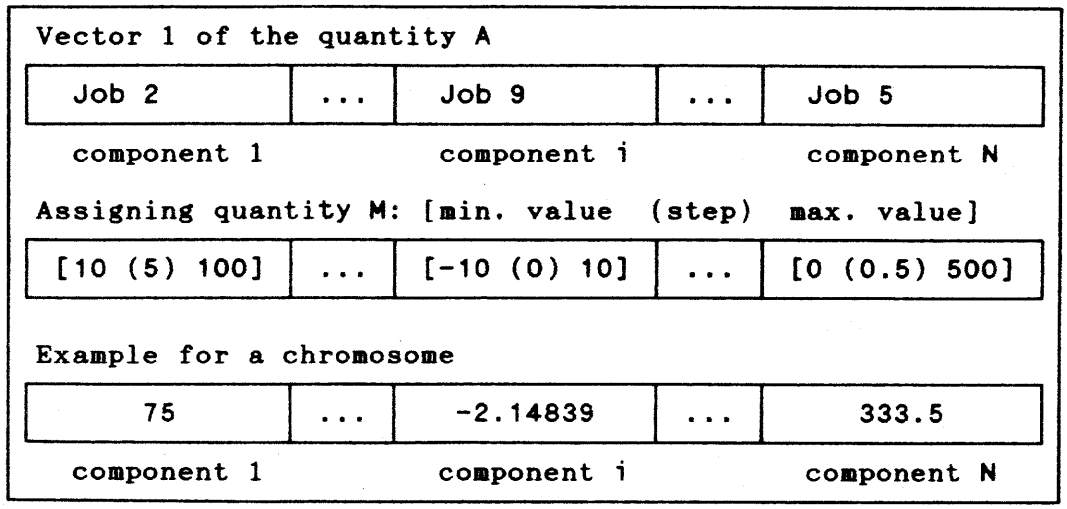

Fig. 5: Example of the chromosome type *assigning*

This basic task contains all problems of determination of lot-sizes, stocking levels, transport quantities, i.e.

Now it is time to put the three basic tasks together to one exemplary individual. At the same time we have to demand that any number of the three basic problems can work with each other. That means we can assemble an optimization problem by any number of several partial problems (chomosome types). The example in figure 6 shows an individual which consists of three chromsomes. Each chromosome is of another type. Notice: There are also individuals with two or more chromosomes of the same type possible! The interpretation of the individual in figure 6 follows: Job number 1 will start at period 3 with 50 pieces on machine 2, and so far and so on up to Job M. Now the evolution can start by working of the genetic operators. Notice: All operators work *only* on one chromosome belonging to themselves! There is also no code changing between two or more chromosomes of the same type. Only the selection works over the whole individual because the fitness of an individual is a function of all alleles engaged with the chromosomes of the individual. Therefore, we need an selection operator which contains both philosophies: probability (GA) and firmness (ES). The EVAOPT-selection procedure follows these steps: μ parents reproduce Γ children without any selection (determined method like used in ES). After that, μ of the $\mu+\Gamma$ individuals (parent and children) will survive by a probabilistic selection dependent on the fitness of each individual and the fitness of the whole population (like used in GA).

Notice: The best individual found by the past evolution can die! This fact makes the evolution possible to leave local optima better. Nevertheless, the best individual is always kept in memory.

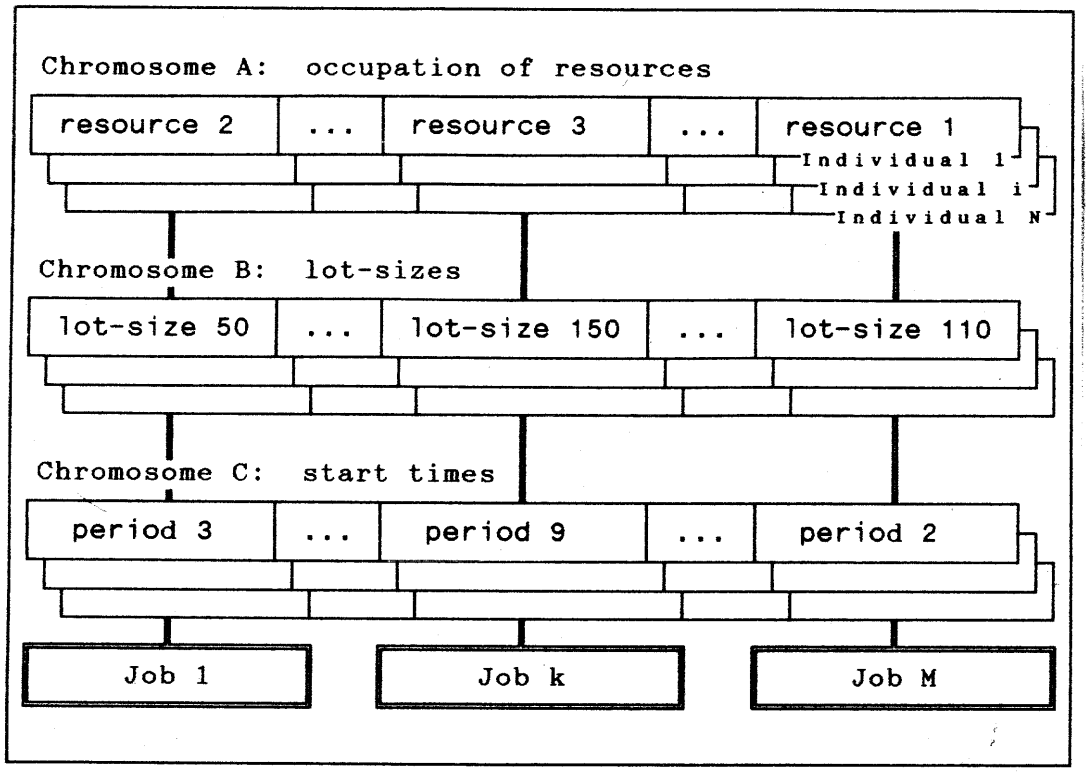

Fig. 6: Example of the parallel concatenation of 3 chromosomes to one each individual

To draw a conclusion back to mathematical terms: Figure 6 shows a set of solutions (one generation). The evolutionary algorithms do *not* guarantee the consistency of the solutions! The unpossible "solutions" must get a so-called penalty-fitness by the simulation program or another fitness evaluation tool. This bad fitness causes the evolution to let the individuals die out.

6. Optimization of a multi-location inventory model with redistribution

Now I want to introduce a exemplary application of evolution strategies. It is a stochastic inventory model with several stocks connected with each other and one type of product. The model was introduced at first by KÖCHEL in [21] and further investigated in [22]. The minimization of the one-period expected costs is not solvable for more than two stocks in the framework of an analytical model until today. Therefore, a simulation tool was designed in [13] which approximates the inventory costs for given stocking levels and expected demands (different distributions are possible) over an observation time of maximum 1000 periodes. Now it was the turn to connect this simulation program with the optimization tool EVAOPT.

The the following assumptions are made:

K_i	$i = 1..n$	ordering costs per product at stock i,
h_i	$i = 1..n$	holding costs per product at stock i,
p_i	$i = 1..n$	rejection costs per product at stock i,
c_{ij}	$i,j = 1..n, i \neq j$	transport costs per product from stock i to stock j,

There are three economic demands suitable:

(I) The efficiency of transport:

$c_{ij} < h_i + p_j$ $\quad i,j = 1..n,\ i \neq j$

Any transport should cause lower costs than no transport at all.

(II) The relative independence of stocks:

$K_i + c_{ij} > K_j$ $\quad i,j = 1..n,\ i \neq j$

$p_i + c_{ij} > p_j$

$h_i + c_{ji} > h_j$

Each stock has to meet the costs incured by itself and must not shift them on other stocks.

(III) The shortest ways:

$c_{ik} + c_{ki} \geq c_{ij}$ $\quad i,j,k = 1..n,\ i \neq j \neq k$

The direct transport between two stocks has to be always cheaper then any transport through another stock.

It is proved in [21] that if the holding, rejection and transport costs are linear, the function of inventory costs will have a *convex* structure!
Our exemplary inventory model should consist of 4 stocks with the following parameters:

$K = (0, 0, 0, 0)$; no order costs
$h = (1, 2, 4, 3)$; units per product
$p = (10, 9, 11, 8)$; units per product

$$C = \begin{bmatrix} \blacksquare & 5 & 7 & 4 \\ 7 & \blacksquare & 8 & 5 \\ 9 & 8 & \blacksquare & 7 \\ 8 & 7 & 9 & \blacksquare \end{bmatrix}$$; units per product

Furthermore, we assume a normal distributed demand with a mean of
$\mu = (200, 300, 250, 150)$ and deviation $\sigma = (30, 40, 20, 10)$;
units per periode.

Now we have all parameters to start the simulation for a given stocking level. The next step we have to do is choosing the right evolutionary algorithm and suitable parameters for it. The problem seems to be an assigning task, because to each stock (set A) will assigned an element of the set (0..∞). Therefore, we use an evolution strategy (chromosome type *assigning*). If we assume the product as bulk good, the optimization problem will be non-discrete, else we have to set step = 1. The obvious conclusion is that one individual consists of only one chromosome with 4 genes.
The evolution process was started with these parameters:

- size of parent population μ: 2
- size of children population Γ: 10
- number of generations G: 250
- rate of crossing over p(C): 0.2
- rate of mutation p(M): 0.7
- rate of reproduction p(R): 0.1 (1 - p(C)-p(M))

The rates p(C), p(M) and p(R) will not changed during the evolution takes place. The first individual of the start-population can be created by analytical solving of the problem for fully independently working stocks (see [21] and [22]):

$x^{(1)} = (240, 336, 262, 156)$

The second individual of the start population is reproduced simply from $x^{(1)}$ by mutation. Now we can start the evolution to find the minimum inventory costs and to observe its work.

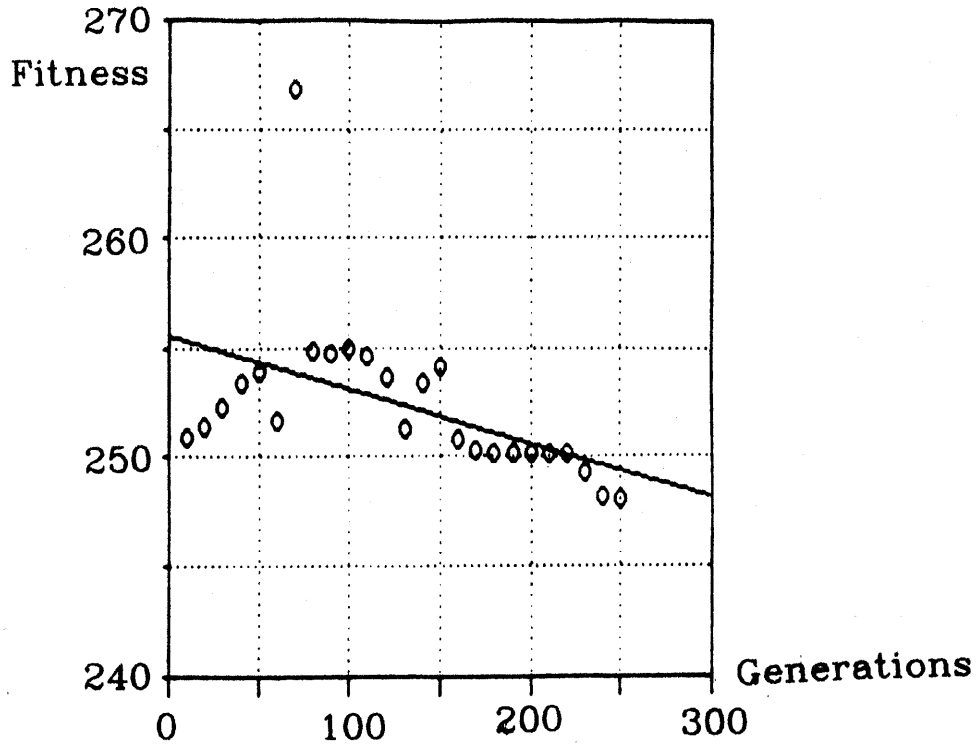

Fig. 7: The fitness-value of the best individual in every tenth generation.
The curve was got from 25 fix points by exponential regression:
certainty: 0.25253 correlation coeff: 0.50252 deviation: 0.000159

There are a lot of interesting things recognizable in figure 7. At first the individuals spread out from the start-point over the parameter space. The best individuals die because there are too many suboptimal individuals! I call it as the boy scout phenomenon. The evolution gains implicit knowledge about the structure of the parameter space for itself. Only after a few generations of knowledge acquisition the evolution will start to search for the real optimum. The optimum was found after 250 generations by

$x^* = (238.8, 324.1, 256.6, 151.9)$ with $f(x^*) = 248.1$ units of inventory costs.

7. Conclusions

The evolutionary optimization took 2500 simulation runs at it. It must not hold back that an optimization algorithm special designed for convex functions (introduced in [13]) took only 80 simulation runs (!) at the same problem. But remember, evolutionary algorithms are suitable for a high performance optimization of very chaotic, completely unknown or non-differentiable functions. You can say that the inventory problem mentioned above is not hard enough for an impressing optimization by evolutionary algorithms. Nevertheless it shows how these algorithms work, and it supports the theoretic assertion that they will find the or a good optimum in the most frequent cases . Try to use the evolutionary algorithms always when the other methods, which you know, had failed. Or better: use them as soon as you feel that there is no known method to solve your optimization problem at all.

References

[1] Ablay, P.:
Optimierung mit Evolutionsstrategien. Spektrum der Wissenschaften, Vol. 7, 1987

[2] Arnold, J.:
Die Verwendung von Evolutionären Algorithmen bei der Optimierung von Fertigungssystemen. Diplomarbeit, TU Chemnitz-Zwickau 1995

[3] Balci, O., Sharda, R. and Zenios, S.A. (Ed.):
Computer Science and Operations Research - New Developments in Their Interfaces. Proceedings of the conference 1992 in Williamsburg, Virginia, Pergamon Press Ltd., Oxford 1992

[4] Beightler, C.S., Phillips, D.T. and Wilde, D.J.:
Foundations of optimization (2nd. ed.). Prentice-Hall, Englewood Cliffs 1979

[5] Belew, R.K. and Booker, L.B. (Ed.):
Proceedings of the fourth international conference on Genetic Algorithms. Morgan Kaufmann Publishers, Inc., San Mateo 1991

[6] Box, G.E.P.:
Evolutionary operation: A method for increasing industrial productivity. Journal of the Royal Statistical Society, Vol. 6(2), pp. 81-101

[7] Bremermann, H.J.:
Optimization through evolution and recombination. In M.C. Yovits, G.T. Jacobi and G.D. Goldstein: Self-organizing systems. pp.93-106, Spartan Books, Washington D.C. 1962

[8] Davis, L. (Ed.):
Handbook of genetic algorithms. Van Nostrand Reinhold, New York 1991

[9] Fandel, G., Gulledge, Th. and Jones, A. (Ed.):
Operations Research in Production Planning and Control, Proceedings of a Joint German/US Conference, Hagen 1992. Springer-Verlag, Berlin, Heidelberg 1993

[10] Filipic, B.:
Enhancing Genetic Search to Schedule a Production Unit. In Neumann, B. (Ed.): ECAI 92. Proceedings of the 10th European Conference on Artifical Intelligence, Vienna 1992, Published by Wiley & Sons Ltd., Chichester 1992

[11] Forrest, S. (Ed.):
Proceedings of the fifth international conference on Genetic Algorithms. Morgan Kaufmann Publishers, Inc., San Mateo 1993

[12] Goldberg, D.E.:
Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, New York, Sidney 1989

[13] Hader, S. and Winkler, F.:
Projektarbeit zur Simulation von Lagerhaltungsmodellen.
TU Karl-Marx-Stadt, Sektion Informatik, 1990

[14] Heistermann, J.:
Genetische Algorithmen: Theorie und Praxis evolutionärer Optimierung.
B.G. Teubner Verlagsgesellschaft, Leipzig 1994

[15] Hesser, J.:
Parameteroptimierung bei Genetischen Algorithmen. Dissertation at the Faculty of Natural and Mathematical Sciences at the Ruprecht-Karls-University of Heidelberg, 1992

[16] Holland, J.H.:
Adaptation in natural and artifical systems. The University of Michigan Press, Ann Arbor 1975

[17] Illgen, H. and Schlott, M.:
Projektarbeit zur Simulation von Lagerhaltungsmodellen. TU Chemnitz-Zwickau, Faculty of Informatics, 1994

[18] Kadelka, D., Kolonko, M. and Voget, S.:
Epistasis Variance in Genetic Algorithms. Hildesheimer Informatikberichte 18/94, Institute of Mathematics at University of Hildesheim, 1994

[19] Keller, U.:
Modelltheoretische Untersuchungen evolutionsbiologischer Mechanismen mit Hilfe Genetischer Algorithmen. Dissertation at the Faculty of Mathematical and Natural Science at University of Cologne, 1994

[20] Kinnebrock, W.:
Optimierung mit genetischen und selektiven Algorithmen.
R. Oldenbourg Verlag GmbH, Munich 1994

[21] Köchel, P.:
Ein stochastisches Lagerhaltungsmodell für mehrere miteinander verbundende Lager. Math. Operationsforschung und Statistik 6 (1975) 3, S. 413-426

[22] Köchel, P.:
Über die optimale Lagerhaltung in einem System von Lagern: eine Näherungslösung. Math. Operationsforschung und Statistik 8 (1977) 1, S. 105-118

[23] Michalewicz, Z.:
Genetic Algorithms + Data Structures = Evolution Programs.
Springer-Verlag, Berlin, Heidelberg 1992

[24] Rawlins, G.J.E. (Ed.):
Foundations of Genetic Algorithms. Proceedings of the workshop 1990 in Bloomington, Indiana, Morgan Kaufmann Publishers, Inc., San Mateo 1991

[25] Rechenberg, I.:
Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Published by Frommann Holzboog, Stuttgart 1973

[26] Schaffer, J.D. (Ed.):
Proceedings of 3rd Intern. Conference of Genetic Algorithms & Applications, Arlington 1989

[27] Schöneburg, E. and Heinzmann, F.:
PERPLEX: Produktionsplanung nach dem Vorbild der Evolution.
Zeitschrift Wirtschaftsinformatik, Vol. 2, 1992

[28] Schöneburg, E.:
Auftrags- und Montagereihenfolgeoptimierung mit Expertensystemen und Evolutionsstrategien. Tagungsband PPS im Wandel, Ed. Prof. Mertens, IFA Institute, Munich 1992

[29] Schöneburg, E., Heinzmann, F. and Feddersen, S.:
Genetische Algorithmen und Evolutionsstratgien. Addison-Wesley, Bonn, 1994

[30] Schulte, J.:
GENIAL 1.0: Genetische Algorithmen zur flexiblen, simulationsbasierten Optimierung. Handbook, AESOP GmbH, Stuttgart 1994

[31] Schwefel, H.P.:
Evolutionsstrategie und numerische Optimierung. Dissertation 1975

[32] Schwefel, H.P.:
Numerical Optimization of Computer Models.
Published by Wiley & Sons Ltd., Chichester 1981

[33] Schwefel, H.P.:
Collective phenomena in evolutionary systems: Reprints of the annual Meeting of the International Society for General System Research, Vol. 2, pp. 1025-1033, Budapest 1987

[34] Schwefel, H.P., Bäck, T. and Kursawe, F.:
Naturanaloge Verfahren. Grundlagen und praktischer Einsatz in der Optimierung. Tutorial at University of Dortmund, 1993

[35] Soucek, B. and The IRIS Group (Ed.):
Dynamic,Genetic, and Chaotic Programming: The Sixth-Generation.
Published by Wiley & Sons, Inc., New York 1992

[36] Sydow, A. (Ed.):
Simulationstechnik - 8. Symposium in Berlin, September 1993. Proceedings, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig, Wiesbaden 1993

[37] Whitley, L.D. (Ed.):
Foundations of Genetic Algorithms · 2. Proceedings of the workshop 1992 in Vail, Colorado, Morgan Kaufmann Publishers, Inc., San Mateo 1993

[38] InterNet-Server at University of Dortmund, Faculty of Informatics,
Chair of Systemanalysis Prof.Dr. Schwefel, H.-P.
WorldWideWeb-Address: www.uni-dortmund.de
FTP-Server: lumpi.informatik.uni-dortmund.de
Directory: /pub