Transactions on the Built Environment vol 28, © 1997 WIT Press, www.witpress.com, ISSN 1743-3509

# Search methods in structural optimization – mixed-discrete variables and parallel computing

J. Cai and G. Thierauf
*Department of Civil Engineering, University of Essen,*
*45117 Essen, Germany*
*E-Mail: th@bauwesen.uni-essen.de*

## Abstract

The basic concepts of two search methods in structural optimization, genetic algorithms and evolution strategies, are introduced. These methods require only information of function-values. Because of their simple search mechanisms, they are well suited for wide classes of optimization problems . The increasing availability of high-speed and parallel computing caused a renewed interest in these zero-order methods, in particular in Monte-Carlo techniques, genetic algorithms and evolution strategies, which are described herein.

## 1 Introduction

Since more than thirty years mathematical programming has become a standard tool for structural optimization. Initially, methods which used only local information of the functions and which required no gradients were favoured. Grid- and tabu-search were first proposed and faced with highly non-convex problems and searching for a most general tool for their solution, the statistical trial and error methods found renewed interest in the mid-sixties. The Monte-Carlo-Method (MCM) as introduced by the famous "Metropolis-Algorithm" [1], generates a set of uncorrelated statistically independent events. These events are random numbers assigned to the variables and the problem under consideration is "tested" by computing and comparing the function values. The MCM has been used for the approximation of integrals and also for the computation of failure probabilities of structures [2]. When applied to optimization problems, the MCM can be compared to a pattern search where the grid size is chosen at random and varies after each function evaluation. In general, in a Monte-Carlo approximation like in a grid search, the number of function evaluations grows

exponentially with the number of variables.

These zero-order and direct methods were followed in the seventies by first- and second-order methods, which required the analytical or numerical computation of first- and second-order gradients. The restriction to the class of differentiable optimization problems was honoured by faster and even superlinear convergence. Due to the high non-linearity and because of the non-convexity of most engineering problems, these gradient-based methods ended up at best at a local optimal design point. Furthermore, their lack of robustness for problems with hundreds of design variables and constraints became apparent. A decomposition of the engineering problems which preserved the superlinear convergence proved to be difficult: The decomposability decreases with the complexity of the problems and the complexity increases with the realistic modelling of the problems. For this reason, the zero-order and direct methods gained renewed interest in the early eighties. This tendency was further enhanced by the availability of high-speed parallel computing. Influenced by this development, stochastic search methods and Darwinian methods have been the subject of many publications during the past fifteen years. Among these, the genetic algorithms (GAs) and the evolution strategies (ESs) are two of the most discussed search strategies. In the following, the basic concepts and a parallelization technique of GAs and ESs are described.

## 2 The basic genetic algorithm

The basic genetic algorithm was proposed in 1975 by Holland [3] and Dejong [4]. In 1989 Goldberg [5] extended the early work to optimization and machine learning. Applications to structural analysis and design were proposed by many other researchers. A genetic algorithm can be considered as an iterative scheme, where each iteration cycle forms a generation of an evolutionary process.

### 2.1 Encoded representation of the design variables

As a main property of GAs the fact of using binary encoded individuals (genotype level) has to be noted. The components $x_i$ of the vector

$$X = (x_1, x_2, \cdots, x_i, \cdots, x_n)^T \tag{1}$$

are considered as genes. By using a binary coding:

$$(x_i)_{10} = \sum_{j=0}^{m-1} b_j \times 2^j, \qquad b_j = 1 \ or \ 0 \tag{2}$$

a component $x_i$ can be represented by a substring with a length of $m$. The encoded components can be considered as chromosomes. A vector of

design variables, usually called an individual, is characterized by a string of chromosomes:

$$\underbrace{0\ 1\ 1\ 0}_{x_1}\ \underbrace{1\ 0\ 1\ 1}_{x_2}\cdots\underbrace{1\ 0\ 1\ 0}_{x_n}.$$

In the following optimization process GAs work only with these coded binary strings.

For a discrete design variable, the length $m$ of the substring depends on the number of the feasible diskrete values of the variable. In case of $m = 4$, for example, 16 discrete values can be represented by the substrings (0 0 0 0) to (1 1 1 1). For a continuous design variable, the length $m$ of the substring depends on the required precision $A_c$, then the length $m$ of the binary substring may be estimated from the folloewing relationship [6]:

$$2^m \geq [(x_U - x_L)/A_c + 1]. \tag{3}$$

Here, $x_L$ and $x_U$ are the lower and upper bounds of a continuous variable $x_i$. It is obviously that a continuous variable is treated as a discrete one by dividing its feasible region $[x_L, x_U]$ into $2^m$ discrete values.

## 2.2 Mating, crossover and mutation

In the $\nu$-th generation, a population of $\mu$ individuals, characterized by their genes (chromosomes) exists:

$$\begin{array}{lcccccccccc}
1.: & 0 & 1 & 1 & 0 & \ldots & 1 & 0 & 1 & 1, \\
2.: & 1 & 0 & 0 & 0 & \ldots & 1 & 0 & 0 & 1, \\
 & & \cdots & & & & & \cdots & & & \\
\mu.: & 1 & 1 & 0 & 0 & \ldots & 1 & 0 & 1 & 0.
\end{array}$$

The $\mu$ individuals are mated at random. As a result of mating, a crossover of chromosomes takes place. The crossover of two individuals $A$ and $B$ results in $A^*$ and $B^*$. In a one-site crossover, the set of chromosomes is subdivided into two groups by one borderline (|) which is set at random. The chromosomes in one of the groups are exchanged as pointed out in the following example:
(before crossover)

$$\begin{array}{rcl}
A & = & (\mathbf{0\ 1\ 1\ 0} \mid \ldots \mathbf{\ 1\ 0\ 1\ 1}) \\
B & = & (1\ 0\ 0\ 1 \mid \ldots\ 1\ 1\ 0\ 1)
\end{array}$$

(after a one-site crossover)

$$\begin{array}{rcl}
A^* & = & (1\ 0\ 0\ 1 \mid \ldots \mathbf{\ 1\ 0\ 1\ 1}) \\
B^* & = & (\mathbf{0\ 1\ 1\ 0} \mid \ldots\ 1\ 1\ 0\ 1)
\end{array}$$

In a two-site crossover the chromosomes between two arbitrarily set borderlines are exchanged, e.g.
(before crossover)

$$\begin{array}{rcl} A & = & (\mathbf{0}\ \mathbf{1}\ \mathbf{1}\ |\ \mathbf{0}\ \ldots\ \mathbf{1}\ \mathbf{0}\ |\ \mathbf{1}\ \mathbf{1}) \\ B & = & (1\ \ 0\ \ 0\ |\ 1\ \ldots\ 1\ \ 1\ |\ 0\ \ 1) \end{array}$$

(after a two-site crossover)

$$\begin{array}{rcl} A^* & = & (\mathbf{0}\ \mathbf{1}\ \mathbf{1}\ |\ 1\ \ldots\ 1\ \ 1\ |\ \mathbf{1}\ \mathbf{1}) \\ B^* & = & (1\ \ 0\ \ 0\ |\ \mathbf{0}\ \ldots\ \mathbf{1}\ \mathbf{0}\ |\ 0\ \ 1) \end{array}$$

As rare and secondary events, mutations occur: With a low probability, certain chromosomes in the genetic code of individuals, selected at random, are exchanged from 1 to 0 or inversely, e.g.:

$$\text{from} \quad A' = (0\ \ 1\ \ 1\ \ \mathbf{1}\ \ldots\ 1\ \ 1\ \ 1\ \ 1)$$

$$\text{to} \quad A'' = (0\ \ 1\ \ 1\ \ \mathbf{0}\ \ldots\ 1\ \ 1\ \ 1\ \ 1)$$

After crossover and mutation $\mu$ new individuals are generated. The $\mu$ old individuals die out.

### 2.3 Reproduction

For a specific optimization problem of the form

$$\begin{array}{ll} \text{minimize} & f(X) \\ \text{subject to} & g_i(X) \leq 0 \quad i = 1, 2, \ldots, m \end{array} \tag{4}$$

the fitness of each individual must be defined. For an admissible point $X$ the value of the objective function can be used for the definition of its fitness. In general it is difficult to generate admissible points in a random process. For this reason, a penalty transformation

$$\text{minimize} \quad f(X) + \rho \sum_{i=1}^{m} \Phi[g_i(X)] \tag{5}$$

can be used, where $\Phi$ is an appropriate penalty function [7] and $\rho$ is a penalty coefficient.

With considering Eq. (4), the fitness of a individual $X_j$ can be defined as follows[8]:

$$F_j = (f_{max} + f_{min}) - f_j, \tag{6}$$

where $f_{max}$ is the maximal value of the objective function between the $\mu$ individuals and $f_{min}$ is the minimal one. According to their fitness values,

each of the $\mu$ new individuals gets $\alpha$ copies into the mating pool of the next generation, where $\alpha$ is calculated as follows, with rounding off:

$$\alpha_j = \frac{F_j}{\sum_{j=1}^{\mu} F_j/\mu}. \tag{7}$$

A good individual $X_j$ (with $\alpha > 1$) will get more than one copy in the next generation and a bad one gets no copy. This process is called reproduction following the Darwinian principle "survival of the fittest". After reproduction a meting pool of the $(\nu+1)$-th generation with $\mu$ individuals is formed, which has a higher average fitness value than that of the $\nu$-th generation.

In general, the population is kept constant and the evolutionary process is carried on until some stopping criteria are met.

Modifications and extensions are known from literature, e.g. [5] and will not be discussed here.

## 3 The basic multi-membered evolution strategy

Evolution strategies are often presented and discussed as a technique competing with genetic algorithms. A concluding decision, which one is superior, is certainly impossible, a comparison of similarities and differences is given in next section.

The evolution strategies were first proposed by Rechenberg [9] in 1964. Applications to optimization of technical systems were proposed by Rechenberg in 1973 [10]. A comprehensive description is given by Schwefel [11] and Hartmann [12]. Applications to structural analysis and design can also be found in [13]. Both, genetic algorithms and evolutions strategies, are directed or controlled statistical search techniques.

In the following we first restrict ourselves to a basic form of the multi-membered evolution.

### 3.1 Recombination and mutation

A main property of ESs that differs from GAs is that ESs work with real values of the variables (phenotype) instead of encoded binary strings.

In the $\nu$-th generation a population of $\mu$ design points, called $\mu$ parent vectors, are given:

$$P^{\nu} = (X_1^P,\ X_2^P,\ \cdots\ X_{\mu}^P) \tag{8}$$

with

$$\begin{aligned} X_1^P &= [x_{1,1},\ x_{1,2},\ \cdots\ x_{1,n}] \\ X_2^P &= [x_{2,1},\ x_{2,2},\ \cdots\ x_{2,n}] \\ &\cdots \\ X_{\mu}^P &= [x_{\mu,1},\ x_{\mu,2},\ \cdots\ x_{\mu,n}] \end{aligned} \tag{9}$$

From these $\mu$ parent vectors, $\lambda$ new design points, called offspring vectors, will be generated. For every offspring vector a temporary parent vector

$\tilde{X}^P = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n]^T$ should be first built by means of recombination. For a continuous problem five recombination cases which can be used selectively are given by Hoffmeister and Baeck [15]:

$$\tilde{x}_i = \begin{cases} x_{a,i} \text{ or } x_{b,i} \text{ randomly} & \text{(A)} \\ 1/2(x_{a,i} + x_{b,i}) & \text{(B)} \\ x_{bj,i} & \text{(C)} \\ x_{a,i} \text{ or } x_{bj,i} \text{ randomly} & \text{(D)} \\ 1/2(x_{a,i} + x_{bj,i}) & \text{(E)} \end{cases} \tag{10}$$

where $\tilde{x}_i$ is the $i$-th component of the temporary parent vector $\tilde{X}^P$, $x_{a,i}$ and $x_{b,i}$ are the $i$-th components of the vectors $X_a^P$ and $X_b^P$ which are two parent vectors randomly chosen from the population. In case (C) of Eq. (10), $\tilde{x}_i = x_{bj,i}$ means that the $i$-th component of $\tilde{X}^P$ is chosen randomly from the $i$-th components of all $\mu$ parent vectors.

For discrete optimization problems the following recombinations are employed [14]:

$$\tilde{x}_i = \begin{cases} x_{a,i} \text{ or } x_{b,i} \text{ randomly} & \text{(A)} \\ x_{bj,i} & \text{(B)} \\ x_{a,i} \text{ or } x_{bj,i} \text{ randomly} & \text{(C)} \\ x_{m,i} \text{ or } x_{b,i} \text{ randomly} & \text{(D)} \\ x_{m,i} \text{ or } x_{bj,i} \text{ randomly} & \text{(E)} \end{cases} \tag{11}$$

Where the vector $X_m^P$ is not chosen at random but as the best of the $\mu$ parent vectors in the $\nu-$th generation. In cases (D) and (E) of Eq (11) the information from the best parent can be used which results in a better convergence for many problems.

From the temporary parent $\tilde{X}^P$ an offspring vector $X_j^O$ can be created by means of mutation as follows:

$$X_j^O = \tilde{X}_j^P + Z_j \,, \tag{12}$$

where $Z_j = [z_{j,1}, z_{j,2}, \ldots, z_{j,n}]^T$ is a vector of random change. For continuous problems the components $z_{j,i}$ are random numbers from a normal distribution [11]:

$$p(z_{j,i}) = \frac{1}{\sqrt{(2\pi)}\,\sigma_i} \exp\left(-\frac{(z_{j,i} - \xi_i)^2}{2\sigma_i^2}\right), \tag{13}$$

where $\xi_i$ is the expectation, which should have the value zero, and $\sigma_i^2$ is the variance, which should be small.

For a discrete problem, the components of the random change vector $Z_j$ have the form [14]

$$z_{j,i} = \begin{cases} (\kappa + 1)\delta x_i & l\,(l < n) & \text{randomly chosen} \\ & & \text{components,} \\ 0 & n - l & \text{other components,} \end{cases} \tag{14}$$

Where $\delta x_i$ is the current difference between two adjacent values in the discrete set and $\kappa$ is a Poisson-distributed integer random number with the distribution

$$p(\kappa) = \frac{(\gamma)^\kappa}{\kappa!} e^{-\gamma}, \tag{15}$$

where $\gamma$ is the deviation of the random number $\kappa$ and should range from 0.001 to 0.1. A uniformly distributed random choice decides which $l$ components should be changed for a mutation according to Eq. (14). For structural optimization problems, according to our research, a suitable $l$ value ranges from 8 to 12 [14]. The first line of Eq. (14) has a symbolic sense: a mutation should be taken from the current discrete value to the $(\kappa+1)$-th adjacent discrete value up or down randomly.

During the optimization process, not only the variables, but also the optimization parameter like $\sigma$ and $\gamma$ will be changed for improving the convergence rate. This is called self-adaptation of the optimization parameters [11, 14].

### 3.2 Selection

Now we have a population of $(\mu+\lambda)$ individuals. Following the Darwinian principle "survival of the fittest" $\mu$ best individuals will be selected according to their fitness for surviving to the next generation. The fitness of a individual is defined as its value of objective function:

$$F_j = f(X_j). \tag{16}$$

There are two variants of the multi-membered evolution strategy: $(\mu+\lambda)$-ES and $(\mu, \lambda)$-ES. In the case of $(\mu+\lambda)$-ES, all the $\mu+\lambda$ individuals are ordered according to their fitness values:

$$F_1 \leq F_2 \leq \cdots F_{\mu+\lambda} \tag{17}$$

The first set of the $\mu$ elements are chosen as the parent vectors of next generation

$$P^{\nu+1} = (X_1^P, X_2^P, \cdots X_\mu^P) \tag{18}$$

In the case of $(\mu, \lambda)$-ES $(\lambda > \mu)$, only the $\lambda$ offspring vectors of the $\nu$-generation are ordered according to their fitness values:

$$F_1 \leq F_2 \leq \cdots F_\lambda \tag{19}$$

The first set of the $\mu$ elements are chosen as the parent vectors of next generation. The $\mu$ parent vectors of the $\nu$-generation die out.

For a continuous problem the search will be terminated [11],

1) if the absolute or relative difference between the best and the worst objective function values is less than a given value $\varepsilon_1$, or

2) if the mean value of the objective function values of all parent vectors in the last $K$ ($\geq 2n$) generations has been improved by less than a given value $\varepsilon_2$.

For discrete problems the following termination criteria can be used [14]. The search will be terminated,

*a*) if the best value of the objective function in the last $K_I$ ($\geq 4n\mu/\lambda$) generations has not been improved, or

*b*) if the mean value of the objective function values from all parent vectors in the last $K_{II}$ ($\geq 2n\mu/\lambda$) generations has been improved by less than a given value $\varepsilon_b$, or

*c*) if the relative difference between the best objective function value and the mean value of objective function values from all parent vectors in the current generation is less than a given value $\varepsilon_c$, or

*d*) if the ratio $\mu_b/\mu$ has reached a given value $\varepsilon_d$, where $\mu_b$ is the number of the parent vectors in the current generation with the best objective function value.

## 4 Parallelization

Both ESs and GAs imitate biological evolution and work simultaneously with a population of design points in the space of variables. This inherent parallelism allows for an implementation of in a parallel computing environment.

With an increasing size of the population, the probability to obtain a global optimum increases almost proportionally. However, large scale problems with an increasing size of population also require much computing time. Following an idea of natural evolution, a parallel sub-evolution-strategy (PSES) was suggest by the authors [14, 17]. A similar parallelization called "islands model" is discussed by Surry and Radcliffe [16]. The population can be divided into several smaller subpopulations which can undergo their evolution separately and in parallel. In order to prevent the development of evolutionary niches, migration between the subpopulations must be allowed.

In the parallel implementation on a $n$-processor computer, the whole population is divided into $n$ subpopulations, each processor runs the GAs or the ESs on its own subpopulation. Periodically some good individuals will be selected and copies of them will be sent to one of its neighbors (migration). Every subpopulation also receives copies from its neighbors, which replace its own "bad" individuals.

The information exchange will be carried out in a determined sequence, For example, from processor i-1 to i and from i to i+1 [14]. During a

optimization process, except for the information exchange, the job on every processor runs independently and a local search can be stopped according to its own termination criterion. If a job on a processor terminated normally, it sends a signal to its neighbours. To keep the cyclic exchange working, exchange between the processor and its neighbours is carried out until all computations are terminated.

## 5 Conclusions and Acknowledgements

Heuristic methods like the Darwinian methods show important advantages compared with conventional methods from mathematical programming: First, they are easily parallelized and second, discrete optimization problems and mixed discrete-continuous problems can be solved approximately. The methods developed for using mathematical programming, e.g. fast techniques for redesign and approximate evaluations of gradients, could be incorporated in the future. An extension to include chance-constraints and the non-deterministic nature of material properties and loading seems to be straightforward. Solutions in this field might give better insight into the optimal design and into the safety of optimized structures.

The research described in this paper was partially supported by German Research Association (DFG) under grant TH-218/12-1/2. The authors would like to thank DFG for the support.

## References

[1] Metropolis, N. and Ulam, S.: The Monte Carlo method. *J. Amer. Statistical Assoc.*, Vol. 44, No. 247, pp 335-341.

[2] Thierauf, G.: An application of Monte-Carlo-Method for computation of failure probabilities of structures (in German). Research report on reliability of structures, Laboratory for Structural Engineering, Technical University of Munich, Sept. 1972.

[3] Holland, J.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, Mich., 1975.

[4] Dejong, K. A.: Analysis of the behaviour of a class of genetic adaptive systems. Ph. D. Thesis, University of Michigan, Ann Arbor, MI. 1975

[5] Goldberg, D. E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley Publishing Co., Inc., Reading, Mass., 1989.

[6] Lin, C.-Y. and Hajela, P.: Genetic algorithms in optimization problems with discrete and integer design variables. *Eng. Opt.* Vol. 19, 1992, pp 309-327.

[7] Kirsch, U.: Structural Optimization - Fundamentals and Applications. Springer-Verlag, Berlin, 1993.

[8] Rajeev, S. and Krishnamoorthy, C. S.: Discrete Optimization of Structures Using Genetic Algorithms. *J. Struct. Engrg., ASCE*, Vol. 118, No. 5, 1992, 1233-1250.

[9] Rechenberg, I.: Cybernetic solution path of an experimental problem, Royal Aircraft Establishment, Library Translation 1122, Farnborough, England, 1965 (in German: 1964).

[10] Rechenberg, I.: Evolution strategy: Optimization of technical systems according to the principles of biological evolution (in German). Frommann-Holzboog, Stuttgart, 1973.

[11] Schwefel, H.-P.: Numerical optimization of computer models. Wiley & Sons, Chichester, 1981 (translated from German, 1977).

[12] Hartmann, D.: Optimization of beamlike cylindrical shells in reinforced concrete including elastic and plastic material behaviour (in German). Doctoral thesis, University of Dortmund, 1974

[13] Hartmann, D.: Computer aided structural optimization by means of evolution strategies. Report No. NCB/SESM-84/8, University of California, Berkeley, USA, June 1984.

[14] Cai, J. 1995. Discrete optimization of structures under dynamic loading by using sequential and parallel evolution strategies (in German). Doctoral Dissertation, Department of Civil Engineering, University of Essen, Germany.

[15] Hoffmeister, F. and Baeck, T. 1992. Genetic algorithms and evolution strategies: similarities and differences. Technical Report No. SYS-1/92, University of Dortmund.

[16] Surry, P. D. and Radcliffe, N. J. 1995. RPL2: A language and parallel framework for evolutionary computing. Proceedings of the Structural Engineering Computational Technology Seminar, Heriot-Watt University, Edinburgh, 1st-2nd May, 1995.

[17] Thierauf, G. and Cai, J. 1994. Evolution strategy, its parallelization and application to discrete optimization problems. NATO Advanced Research Workshop: Emergent Computing Methods in Engineering Design, Nafplio, Greece, August 25-27, 1994.