*Belgian Journal of Operations Research, Statistics and Computer Science Vol. 32 n° 1-2*

# Construction of exact D-optimal designs for linear regression models using genetic algorithms

B. Govaerts

D.C.T. Automation, Solvay S.A.
rue de Ransbeek 310
B-1120 Bruxelles, Belgium

P. Sanchez Rubal

avda Ramon y Cajal 81
E-28016 Madrid TLF, Spain

**Abstract**

This paper reports the application of genetic algorithms to the construction of exact D–optimal designs for linear regression models. D–optimal designs and the 2–exchange Federov algorithm are introduced. A modification of this algorithm, based on simulated annealing, is then discussed. This procedure is taken as a basis for the cross–over and mutation operators used by the genetic algorithm that is proposed in this paper. Finally, the performance of this algorithm is illustrated by applying it to the construction of exact D–optimal designs for the 2nd order model with 2 factors on the domain $[-1.0, 1.0] \times [-1.0, 1.0]$.

**Keywords** : experimental design, Federov algorithm, genetic algorithms, optimal design, simulated annealing

# 1 Introduction

In industries and research laboratories experiments are organized daily in order to design new products, improve existing ones or optimize industrial processes. Experimental design deals with determining, before any experiment is carried out, the set of experimental runs to be performed in order to get a targeted objective as efficiently as possible, which generally means at little cost.

For example, it may be desired to find the proportions of 2 additives $A$ and $B$ to be mixed to a polymer that maximize the thermic stability of the final product. The quantities of additives $x_A$ and $x_B$ are usually called *control factors* and the thermic stability $y$, the *response*. Many experiments would solve this problem by using a "one factor at a time" experimental strategy which means optimizing $y$ sequentially with respect to $x_A$ for a fixed value of $x_B$, then with respect to $x_B$ for a fixed value of $x_A$, and so on.

This approach however is usually expensive, poorly informative and may not lead to an optimum if $A$ and $B$ act in synergy. Experimental design techniques advocate a more global approach for such a problem. They suggest to assume, *a priori*, that the relation between the control factors and the response obeys a mathematical model of the form $y = f(x_A, x_B, \theta)$ and to select in the domain of interest, a particular set of experimental runs, usually called a *design*, which will permit to estimate precisely the *parameters* $\theta$ of the model and make predictions. Since the true relation is generally not known, an approximation model must be used. It is usually chosen in the class of *polynomial models* which are linear in their parameters. In the case of the example discussed above, a second order polynomial of the form

$$y = \beta_0 + \beta_A x_A + \beta_B x_B + \beta_{AA} x_A^2 + \beta_{BB} x_B^2 + \beta_{AB} x_A x_B + \epsilon \qquad (1)$$

can be used, where $\epsilon$ stands for an error term.

Many types of experimental designs exist each of them being devoted to a particular class of problems. They can be classified in two main categories: *classical designs* and *optimal designs*. A wide range of classical designs can be found in the literature (see for example Box *et al.* 1978, Box and Draper 1987, Khuri and Cornell 1987). Among the best known, we have the factorial, fractional factorial, Doehlert, Plackett-Burman, central composite, simplex,

Box-Behnken and Taguchi designs. These designs have very good and well known statistical properties. They have different shapes, usually symmetrical, and are very elegant tools to solve a lot of experimental design problems. If we assume, in our example, that the proportion of each additive may vary from 0 to 5 %, three suitable classical designs are shown in Figure 1. Each of

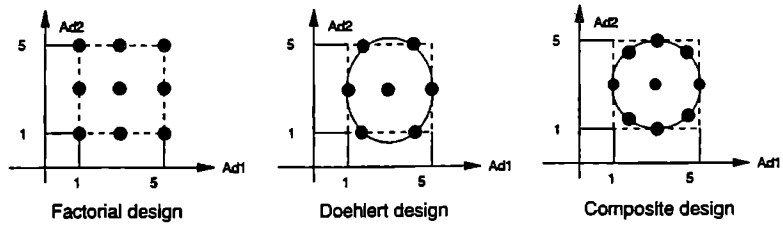


Figure 1: Three classical designs.

them is liable to solve the problem, which means estimate the model given in (1). The choice between them will depend on the budget and the objective of the experimenter.

Unfortunately, it turns out that classical designs are limited to "classical problems" which are not often met in industry. They are most of the time expensive, need cubic or spherical domains of interest and do not accept qualitative control factors, as for example the type of catalyst used in the experiment.

Optimal designs, by their flexibility, can be used in such situations. Such designs are built by choosing, in the domain of interest, a set of points which maximizes or minimizes a given optimality criterion. This criterion is generally expressed under the form of a functional of the variance-covariance matrix of the model parameters like the D-criterion which amounts to minimize the determinant of this matrix. The algorithms used to solve the resulting combinatorial problem are those of mathematical programming such as exchange algorithms, branch-and-bound, ..., and so on (see St. John and Draper 1975, Cook and Nachtsheim 1980, Atkinson and Donev 1992). Recently, however, some emphasis has been devoted to the application of

new heuristic algorithms like simulated annealing (Haines 1987, Meyer and Nachtsheim 1988, Crary *et al.* 1992).

The goal of this paper is to apply another class of heuristic tools, called genetic algorithms, to the generation of exact D-optimal designs for linear regression models on finite design spaces. Below, we first introduce the notations and definitions that will be used throughout the paper. After having presented the exchange algorithm proposed by Federov to generate exact D-optimal designs, we then propose a modified Federov algorithm based on simulated annealing and discuss the results that have been obtained by other authors when applying the simulated annealing approach to this problem. Next, we introduce the principle of genetic algorithms and describe the procedure that we have implemented for the generation of D-optimal designs. Finally, we illustrate its performance on the example presented in this introduction before making some concluding remarks.

# 2 Definitions and notations

Throughout this paper, we will assume that each observation $y_i$ of the response can be written as a linear function

$$y_i = f'(x_i).\beta + \epsilon_i \qquad i = 1, 2, \ldots, N \tag{2}$$

of the control factors at a point $x_i = (x_{i1}, \ldots, x_{ik})'$ of the domain of interest $\chi \subseteq \mathbb{R}^k$. In (2), $f'(x_i)$ stands for a function from $\mathbb{R}^k$ to $\mathbb{R}^p$, continuous on $\chi$, which depends on the form of the assumed model, $\beta$ is a $p$ component vector of unknown parameters and the $\epsilon_i$ are independent and identically distributed random variables of zero mean and variance $\sigma^2$. Letting $f'(x_i)$ denote the $i$-th row of a matrix $X$, (2) may be rewritten

$$y = X\beta + \epsilon \tag{3}$$

where $y = (y_1 \ldots y_N)'$ and $\epsilon_i = (\epsilon_1 \ldots \epsilon_N)'$.

A $N$-point *exact design* on $\chi$, noted $\xi^N$ is a list of $N$ elements $x_1, \ldots, x_N$ of $\chi$ not necessarily different from each other. $\Xi_\chi^N$ denotes the space of $N$-point exact designs on $\chi$.

It is assumed that the parameters $\beta$ of the model (3) are estimated by least-squares techniques. Those estimates are given by

$$\hat{\beta} = (X'X)^{-1}X'y \tag{4}$$

with variance-covariance matrix

$$V(\hat{\beta}) = \sigma^2(X'X)^{-1} = \frac{\sigma^2}{N}M^{-1}(\xi^N) \tag{5}$$

where

$$M(\xi^N) = \frac{X'X}{N} \tag{6}$$

is called the *moment matrix.*

Then, the predicted response at a given point $x \in \chi$ is given by

$$\hat{y}(x) = f'(x)\hat{\beta} \tag{7}$$

with variance

$$\begin{aligned} v(\hat{y}(x)) &= \sigma^2 f'(x)(X'X)^{-1}f(x) \\ &= \frac{\sigma^2}{N}f'(x)M^{-1}(\xi^N)f(x) = \frac{\sigma^2}{N}d(x,\xi^N) \end{aligned} \tag{8}$$

where $d(x,\xi^N)$ is called the *variance function.* By analogy, we need also to define the *covariance function*

$$d(x_i,x_j,\xi^N) = f'(x_i)M^{-1}(\xi^N)f(x_j) \tag{9}$$

The design problem then consists of choosing $N$ experimental runs in $\chi$ in such a way that a specified criterion, which depends on $x_1,\ldots,x_N$, is maximized or minimized. In this paper, we will consider only the D-criterion. A $N$-point exact design $\xi_*^N$ is said to be D-optimal on $\chi$ if it is defined in the following manner

$$\xi_*^N = \arg \max_{\xi^N \in \Xi_\chi^N} |M(\xi^N)| \tag{10}$$

which means that it minimizes the determinant of the variance-covariance matrix of $\hat{\beta}$ or, equivalently, the volume of the confidence region for $\beta$.

Of course, other criteria related to $M(\xi^N)$ can be used: A-optimal designs minimize the trace of $M^{-1}(\xi^N)$, G-optimal designs minimize $d_{max} = \max_{x\in\chi} d(x,\xi^N)$, and I-optimal designs minimize $d_\mu = \int_{x\in\chi} d(x,\xi^N)d_\mu(x)$ where $\mu$ is a probability measure on $\chi$ to be defined by the experimenter. The A-criterion is proportional to the sum of the parameter variances, while the G and I-criteria are respectively proportional to the maximum and mean of the prediction variance over the experimental domain $\chi$.

Note that, for the sake of simplicity and since we will only consider $N$-point designs in this paper, $\xi^N$ will simply be noted $\xi$ in the sequel.

# 3 The Federov algorithm

Several algorithms have been proposed to generate exact D-optimal designs. Up to now, the most successful ones are exchange algorithms. They begin with a $N$-point design $\xi_0$ and then add and delete one or more points of the design in order to achieve an increase of the determinant. An extensive literature is available on those algorithms (see for example Federov 1972, Mitchell 1974, St. John and Draper 1975, Cook and Nachtsheim 1980). The first of them to be developed, introduced by Federov, is presented below.

The Federov algorithm (Federov 1972) starts with a $N$-point non singular design $\xi_0$, which means a design for which $M(\xi_0)$ is a non singular matrix. The $i$-th iteration consists of choosing a point of $\xi_i$ to be deleted, say $x$, and a point $z$ of $\chi$ to be added to $\xi_i$ in such a way that the increase of the determinant of $M(\xi_i)$ is maximal. The procedure terminates when no exchange which significantly improves the D-criterion exists.

Federov has shown that (Federov 1972)

$$|M(\xi_{i+1})| = |M(\xi_i)|.(1 + \Delta(x,z,\xi_i)) \tag{11}$$

where

$$\begin{aligned}\Delta(x,z,\xi_i) &= -d(x,\xi_i) + d(z,\xi_i) - d(x,\xi_i).d(z,\xi_i) \\ &\quad +d^2(x,z,\xi_i)\end{aligned} \tag{12}$$

This formula shows that a new determinant can be calculated without having to update the matrix $M(\xi_i)$ and compute its determinant from scratch. In

practice, the computations can be performed by having recourse to a QR decomposition of $X/\sqrt{N}$ which can be updated at low cost when a point is deleted from or added to $\xi$. The use of this decomposition also allows a quick computation of the matrix $M^{-1}(\xi)$. See Golub and Van Loan (1983) for further details about the QR decomposition and its updating.

In what concerns the domain of interest $\chi$, it is generally continuous and optimizing on such a space is tedious. For this reason, it is usually replaced in practice by a discretization of it that will be referred to as the *candidate set*, noted $\chi_c$. In this paper, we will assume that the domain of interest has been discretized. According to our experience, this fulfils most of the experimenter's needs. Indeed, even when domain factors are continuous, experimenters usually have in mind a discrete set of possible levels for each factor and want to minimize the number of changes in the factor setting over the whole experiment.

Note also that the Federov algorithm does not guarantee the achievement of a global optimum. It is thus advisable to start it from different initial designs chosen randomly in $\Xi_{\chi_c}^N$. Another way to avoid this problem is to use a simulated annealing approach. Such algorithms have the "ability to migrate through a sequence of local extrema in search of the global solution and to recognize when the global extremum has been located" (Bohachevsky *et al.* 1986). In Section 4, we report the results that have been obtained by different authors which have tried to apply this approach to optimal design.

However, before continuing, we first introduce here a modified Federov algorithm, based on simulated annealing, that will be used later in the implementation of our D-optimal algorithm. For this purpose, two modifications have been made to the Federov algorithm:

1. Rather than trying systematically all exchanges between all points of the design $\xi_i$ and all points in the candidate set $\chi_c$, a subset of points to be tested is chosen randomly in each set.

2. An exchange between two such points $x$ and $z$ may be accepted even if the function $\Delta(x, z, \xi_i)$ is negative. However, such an exchange is only accepted with a probability which is inversely proportional to the decrease of the determinant.

As a consequence, the simulated annealing variant of the Federov algo-

rithm starts with a $N$-point non singular design $\xi_0$ while its $i$-th iteration is performed as follows:

- Pick randomly $m_1$ points in $\xi_i$, $m_2$ points in $\chi_c$, and try all available exchanges between these points. Keep the pair $(x, z)$ which maximizes $\Delta(x, z, \xi_i)$.
- Then apply the following:
  - If $0 \leq \Delta(x, z, \xi_i)$, exchange $x$ with $z$ in $\xi_i$.
  - If $-1 < \Delta(x, z, \xi_i) < 0$, exchange $x$ with $z$ in $\xi_i$ with probability

$$p = (1 + \Delta(x, z, \xi_i))^{\frac{1}{T}} = \left(\frac{M(\xi_{i+1})}{M(\xi_i)}\right)^{\frac{1}{T}}$$

    where the parameter $T$, called the *temperature*, should be decreased by steps as $i$ increases, following a negative exponential function for example. The decision to perform the exchange is thus taken simply by generating a random number $u$ in $[0.0, 1.0]$. The exchange is then accepted if $u \leq p$. Otherwise, it is rejected.
  - If $\Delta(x, z, \xi_i) = -1$, do nothing since $M$ becomes singular.

The process is stopped when no significant change in $M(\xi_i)$ can be observed.

We have observed that this modified Federov algorithm is able to find global optima which are ignored by the standard Federov algorithm, but also that it has the default of being quite slower than the latter.

This simulated annealing variant of the Federov algorithm can also be seen as a non deterministic variant of the $KL$ exchange algorithm presented in the literature (see e. g. Atkinson and Donev 1992). The $KL$ exchange algorithm suggests to reduce the large number of exchanges tested at each iteration of the Federov algorithm ($N \times (N_c - 1)$) to $K \times L$ exchanges: at each iteration, the $K$ points of $\xi_i$ with the lowest variance $d(x, \xi_i)$ $(1 \leq K \leq N)$ are considered for exchange with the $L$ points of $\chi_c$ having the highest variance $d(z, \xi_i)$ $(1 \leq L \leq N_c - 1)$.

# 4 From simulated annealing to genetic algorithms

Several authors have applied the simulated annealing technique to search for approximately optimal designs (see for example Haines 1987, Meyer and Nachtsheim 1988, Crary *et al.* 1992). They experimented on both discretized and continuous design spaces. They did not only investigate D-optimality but also G and I-optimality. The results were compared with those obtained by currently used exchange algorithms.

By experimenting on continuous design spaces for 1st and 2nd order polynomials, Haines (1987) shows that simulated annealing is much slower than traditional heuristics, though as effective, for the D-criterion. On the contrary, it is judged equivalent on the G-criterion and much faster and better on the I-criterion. More extensive experimentations by Meyer and Nachtsheim (1988) yield the conclusion that simulated annealing can be "simply implemented and cheaply used to search for globally (D-) optimal designs on as many as $N = 1000$ candidate points" on a discretized design space. However, their results for continuous convex design spaces are not encouraging.

Usually, one run of simulated annealing is compared with several runs of a deterministic algorithm, like Federov's, starting from different initial designs (10 trials in Haines 1987). In spite of theoretical convergence results, which are irrelevant in practice, the final solution yielded by simulated annealing is subject to non-negligible variations when the starting point and the random number sequence are varied (see for example Figure 9, p. 873, in Johnson *et al.* 1989). Hence, it is advisable, though not common practice, to restart simulated annealing several times. This fact is a first motivation for trying an approach where a *population* of initial designs would be transformed in parallel. Another attractive feature of such an approach is that it is conceivable to try to mix together some designs in order to retain their "good characteristics" and eliminate "bad" experimental points. This idea led us to try *genetic algorithms* (Holland 1975, Goldberg 1989).

# 5 Genetic algorithms

The principle of genetic algorithms is the following. Consider the problem of maximizing a function $F$ on a solution space which is a finite set. The algorithm does not deal directly with the solutions but with an appropriate encoding of them. Usually, each solution will be represented by a *word* composed of *letters* chosen in a given *alphabet*. The simplest and commonest example is a binary encoding in bitstrings (0-1 vectors). As the algorithm essentially works by recombining pairs of words and altering some letters, it is recommended to choose meaningful encodings in which letters or groups of letters are associated with characteristics of the solution that are relevant to the optimization problem.

The algorithm starts with a population $\xi^{(0)}$ of $m$ encoded solutions $(\xi_1^{(0)}, \ldots, \xi_m^{(0)})$ where each solution $\xi_j^{(i)} \in \xi^{(i)}$ is evaluated through its *fitness* which is simply the value of $F(\xi_j^{(i)})$.

At step $i$, a certain number of "good" solutions are first *selected* from $\xi^{(i)}$: solutions are drawn at random from $\xi^{(i)}$ with probabilities increasing with their fitness.

The selected solutions are then mixed together by pairs through an operation called *cross-over*. The simplest version, called 2-point cross-over, consists of exchanging the letters appearing in two words at two arbitrarily chosen positions. Each cross-over operation usually yields two *children*, two new solutions, that will substitute "bad" elements in $\xi^{(i)}$ after being possibly submitted to *mutation*. This last operation performed on a small proportion of the normal children consists of changing arbitrarily some letters chosen at random.

The "bad" solutions to be removed and substituted by the children are chosen in $\xi^{(i)}$ with probabilities decreasing with their fitness. The remaining elements from $\xi^{(i)}$ together with the children obtained by cross-over and mutation constitute the new population $\xi^{(i+1)}$.

The whole procedure stops after a predefined number of generations $N_g$. Normally, the average fitness of the population tends to increase at each step due to the "selection pressure". Figure 2 below illustrates a typical step of the algorithm although many variants departing from this basic scheme can

be found in the literature.

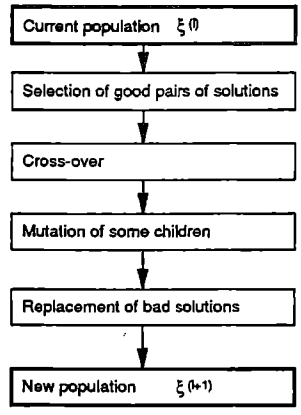


Figure 2: A typical step of the genetic algorithm.

# 6 Description of the algorithm

Our algorithm follows the general genetic algorithm scheme with some peculiarities described below. The solution space is the set of all possible $N$-point designs $\Xi_{\chi_c}^N$. The chosen alphabet is $\chi_c$ which means that each point of the domain of interest is a letter. Each solution $\xi$ is then coded simply as a list of points in $\chi_c$. The function $F$ to be maximized is the D-criterion $|M(\xi)|$. A description of each step of the general genetic algorithm introduced above follows.

## 6.1 Generation of an initial population $\xi^{(0)}$

The initial population $\xi^{(0)}$, as well as the subsequent ones, is made of $m$ non singular designs $(\xi_1^{(0)}, \ldots, \xi_m^{(0)})$. Each design $\xi_j^{(0)}$ is composed of $N$ points of

$\chi_c$ and is generated by first reordering $\chi_c$ at random and by choosing the first $N$ points in this list which ensure the non singularity of the matrix $M(\xi_j^{(0)})$. Each design $\xi_j^{(0)}$ of $\xi^{(0)}$ is also characterized by its fitness $F_j^{(0)} = |M(\xi_j^{(0)})|$.

## 6.2 Selection

The selection step consists of drawing at random $l$ pairs ($l \leq m$) of designs from the current population $\xi^{(i)}$. This drawing is performed with replacement and the probability $p_j$ of choosing a design is made proportional to its fitness through the following expression

$$p_j = \frac{F_j^{(i)} - F_{min}^{(i)}}{\sum_{k=1}^{m} F_k^{(i)} - m F_{min}^{(i)}} \tag{13}$$

where $F_{min}^{(i)} = \min\{F_k^{(i)} | \xi_k^{(i)} \in \xi^{(i)}\}$.

## 6.3 Cross-over

The cross-over operator is applied to each pair of designs, called parents, drawn during the selection procedure. The choice of the cross-over operator is crucial. Ideally, it should be able to generate new solutions that retain on the average the "good" characteristics of the parents. In experimental design, a dream would be an operator that could select and paste together two sub-designs from the parents and obtain a better design. In order to be able to use formula (11), however, one is always forced to alter the parents step by step through a sequence of 2-exchanges.

On the basis of 2-exchanges, there are a lot of possibilities to cross-over 2 designs. On the experience gained when using both the standard and modified Federov algorithms presented in Section 3, we have chosen a single, asymmetric operator with loss of information: it yields a single child (from two parents), parents play an asymmetric role, and some of the design points which are present in the parent designs are lost by the unique child.

More precisely, suppose that a pair $(\xi_1, \xi_2)$, where $\xi_1$ is the first drawn, must be crossed over. The chosen operator consists of applying $N_c$ iterations of the modified Federov algorithm introduced in Section 3 by considering $\xi_1$

as the design to be improved, $\xi_2$ as the candidate set, and taking $m_1 = 1$ and $m_2 = N$. This means that the following statements are executed $N_c$ times:

- Draw a point $x$ in $\xi_1$ at random.
- Try all possible exchanges of $x$ with points $z$ of $\xi_2$ and keep $z$ such that $\Delta(x, z, \xi_1)$ is maximized.
- Then do the following:
  - If $0 \leq \Delta(x, z, \xi_1)$, replace $x$ by $z$ in $\xi_1$.
  - If $-1 < \Delta(x, z, \xi_1) < 0$, replace $x$ by $z$ in $\xi_1$ with probability
$$p = (1 + \Delta(x, z, \xi_1))^{\frac{1}{T_c}}$$
  - If $\Delta(x, z, \xi_1) = -1$, do nothing.

The design $\xi_1$ obtained after those $N_c$ steps of cross-over is the child of $\xi_1$ and $\xi_2$.

Note that, contrary to the simulated annealing algorithm, the temperature $T_c$ is kept constant during all the search procedure.

## 6.4 Discussion of the choice made for the cross-over operator

The choice of our cross-over operator has been guided by the techniques which are accepted to be performant in the statistical literature (Federov algorithm, $KL$ exchange algorithm, simulated annealing, ...). Our goal was a very pragmatical one: find an algorithm able to find a D-optimal design from a population of initial designs without having necessarily a "pure" application of the genetic algorithms. For this reason, the final product may seem quite unorthodox with respect to the usual genetic algorithm standards. In this section, we try however to argue that what we propose is well in the spirit of genetic algorithms.

Our cross-over operator is probably closest to a *uniform* cross-over (Ackley 1987, Syswerda 1989): it gives in principle the same chances of survival

or disruption to all substrings (of equal length). Such an operator seems appropriate here as a design is just a *list* of points (in our coding) without any intrinsic order. Moreover, in a D-optimal design, the contribution of a point or a subset of points *essentially* depends on the other points in the design. This means that a good design is not only a *linear* juxtaposition of good sub-designs and that usual theory of schemata (Holland 1975) does not apply.

A second feature of our operator is that it involves local optimization: the randomized selection of exchanges between the parents is biased towards the good exchanges. The mutation operator (see below) also presents such a feature. This is not unusual in genetic algorithms especially when dealing with hard combinatorial problems. For the travelling salesman problem, Grefenstette *et al.* (1985) devised such a locally optimized operator. Local optimization is also an important element of parallel genetic algorithms of Mühlenbein (1990).

The main argument for including local optimization in our cross-over is the heavy combinatorial aspect: unlike in 0-1 codings, each "site" of a chromosom can take a huge variety of different values ($N_c$). Moreover, previous experiments with simulated annealing show that it is extremelly time consuming to make completely random exchanges. This is, among all, because the computing time needed to *do* an exchange is much larger than the time needed to *test* an exchange. As large instances of the problem are to be treated in practice (e. g. designs of 50 points from 1000 points candidate points) it is very unlikely that pure random exchange could be of practical interest. Note finally that our local optimization feature is carefully designed for avoiding being trapped in local optima has they involve a part of randomness inspired by simulated annealing.

To finish this discussion, one word about the asymetry of our cross-over: in our view, the fact that the unique child is produced with major contribution of one of the parents is of secondary importance as all individuals have a chance of being the important parent. Alternatively, the asymetry can be viewed as an additional local optimization feature.

## 6.5 Mutation

The mutation operator is applied to a certain proportion of child designs. It consists of trying to modify each chosen child by using information which is not in the current population. In our case, we modify a child design $\xi$ by performing 2-exchanges between $\xi$ and randomly chosen subsets of the candidate set $\chi_c$.

For each child design $\xi$, the mutation algorithm first decides if $\xi$ must be mutated, each design having a probability $P_m$ to be mutated. If $\xi$ has to be mutated, $N_m$ points are drawn in $\xi$ and $N_\chi$ in $\chi_c$. For each point chosen in $\xi$ one then searches for the best 2-exchange with one of the points of the reduced candidate set. The decision to apply the exchange is taken along the same rule as the one used during cross-over with a temperature $T_m$ replacing the temperature $T_c$. Note that the candidate set is reduced to a subset ($N_c$ elements of $\chi_c$) in order to diminish the amount of computations.

## 6.6 Replacement

This procedure consists of choosing $l$ "bad" designs in the population $\xi^{(i)}$ and substituting them with the $l$ generated children. A design $\xi_j^{(i)}$ is chosen for removal with probability

$$\overline{p}_j = \frac{F_{max}^{(i)} - F_j^{(i)}}{mF_{max}^{(i)} - \sum_{k=1}^{m} F_k^{(i)}} \tag{14}$$

where $F_{max}^{(i)} = \max\{F_k^{(i)} | \xi_k^{(i)} \in \xi^{(i)}\}$.

## 6.7 Parameter choice

From the steps described above, it appears that the following parameters have to be defined:

$m$ Number of designs in a generation or population.

$l$ Number of pairs selected for cross-over.

$N_c$ Number of points to be exchanged from the best parent designs during cross-over.

$T_c$ Cross-over temperature parameter.

$P_m$ Child probability of mutation.

$N_m$ Number of points to be exchanged from the mutated design.

$N_\chi$ Number of points in the reduced candidate set for mutation.

$T_m$ Mutation temperature parameter.

$N_g$ Number of generations.

Note that unlike simulated annealing, the temperatures $T_c$ and $T_m$ remain fixed. These 2 parameters only tune the facility with which a worsening exchange is accepted. The selection is taken in charge by the genetic procedure. The choice of this set of parameters is discussed in Sanchez (1992) where their influence on the computing time and the fitness of the final solution have been illustrated on some simple experimental design problems.

# 7 Example

The example given in the introduction will be used to illustrate the present application of genetic algorithms to the generation of D-optimal designs on discrete design spaces. If we consider that the experimenter is allowed to run a 9 points design, the problem consists of finding which points of the square $[0.0, 5.0] \times [0.0, 5.0]$ will maximize the D-criterion for the second order polynomial given in (1). In this case, the exact D-optimal design is the $3^2$ factorial design given in Figure 1.

The standardized domain $[-1.0, 1.0] \times [-1.0, 1.0]$ has been discretized following a grid of $11 \times 11$ points going from $-1$ to 1 by steps of 0.2. Note that the best value of the D-criterion when the domain is reduced to $[-1.0, 1.0]^2$ is 0.00976. The algorithm has been run with the following set of parameters: $m = 10$, $l = 6$, $N_c = 5$, $T_c = 0.7$, $P_m = 0.4$, $N_m = 5$, $N_\chi = 25$, $T_m = 0.2$

and $N_g = 50$. These values have been chosen on the basis of the experiments accounted for in Sanchez (1992).

Figure 3 shows the evolution of the minimum, mean and maximum value of the D-criterion over the 10 designs for each generation.

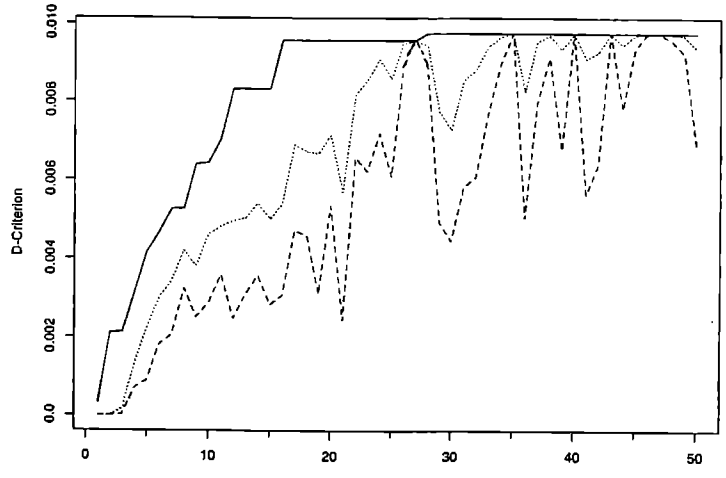


Figure 3: The evolution of the minimum, mean and maximum value of the D-criterion.

Figure 4 gives the 10 designs for generation 1, 5, 10, 15, 20, 25, 30 and 35. The designs are ranked following their fitness. The optimal designs are surrounded with double line frames. Nearly optimal designs are surrounded with double dashed frames.

Those figures are self-comprehensible. However, some remarks are in order here:

1. An optimal design is obtained after 28 generations.

2. At generation number 35, all designs are optimal (see Figure 4) but,

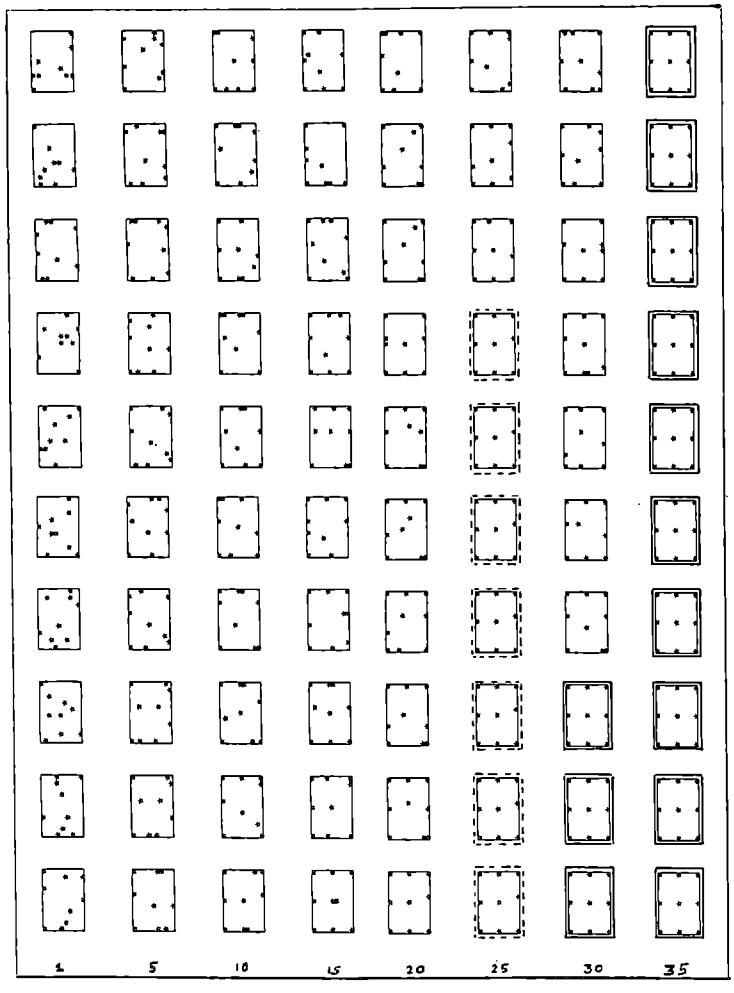


Figure 4: The 10 designs for generation 1, 5, 10, 15, 20, 25, 30 and 35.

after it (see Figure 3), the mutation operator introduces new design points in the population which decrease the D-criterion of some designs.

3. Figure 3 shows that at step 27, all the designs have the same fitness, $F = 0.00957$, which is not optimal. At this step, all the designs are the same (see the first seven designs of generation number 25). No improvement could be obtained there by cross-over. One can see from Figure 3 that from generation number 27 to generation number 35, the mean fitness function begins to decrease due to mutations and then increases towards a better population.

# 8 Conclusions

This study should be considered as a first attempt to apply genetic algorithms to the generation of optimal designs. Additional efforts are needed to experiment it with a wider variety of models and master the choice of the parameters. Nevertheless, in the few situations we have worked on, the results have proven to be reasonable when compared with those obtained by simulated annealing or with the Federov algorithm. From a computational point of view, we thus observed that our genetic algorithm approach compared favourably with simulated annealing and Federov method.

An interesting feature of a genetic approach is its intrinsic parallelism: several solutions evolve iteratively which is presumably at least as good as the multi starting mechanism used by Federov or simulated annealing algorithms. Since a population of designs is finally obtained, instead of a single one, and when diversity remains sufficient, one can imagine to choose between the most interesting designs following other criteria. For this purpose, there exist specialized genetic algorithms techniques called *niching* (Gold 1992) that aim at retaining as many different global optima or, more modestly, distinct good solutions, as possible in the final population. This is a potential direction for further research.

The most interesting development however would be the design of a cross-over operator that would enable to more deeply take into account the geometry of the parent designs during the construction of the child designs. Without any doubt, it is in this direction that the contribution of genetic

algorithm could be really substantial and therefore, future effort should concentrates on this particular point.

*Acknowledgements*: The authors are grateful to M. Pirlot and M. Haest for their substantial contribution in the writing of this paper. They also thank the reviewer for his useful remarks and suggestions.

# 9 References

Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hill-climbing.* Kluwer Academic Publishers.

Atkinson, A. C., and A. N. Donev (1992). *Optimum Experimental Designs.* Clarendon Press, Oxford.

Bohachevsky, I. O., M. E. Johnson and M. L. Stein (1986). Generalized simulated annealing for function optimization. *Technometrics*, 28, pp. 209-217.

Box, G. E. P., and N. R. Draper (1987). *Empirical Model Building and Response Surfaces.* John Wiley and Sons, New York.

Box, G. E. P., W. G. Hunter and J. S. Hunter (1978). *Statistics for Experimenters: an Introduction to Design, Data Analysis and Model Building.* John Wiley and Sons, New York.

Cook, R. D., and C. J. Nachtsheim (1980). A comparison of algorithms for constructing exact D-optimal designs. *Technometrics*, 22, pp. 315-324.

Crary, S. B., L. Hoo and M. Tennenhoure (1992). I-optimality algorithms and implementation. *Proc. 10-th Symposium on Computational Statistics.* Y. Dodge and J. Whittaker (Eds.), Springer Verlag.

Federov, V. V. (1972). *Theory of Optimal Experiments.* Translated and edited by W. J. Studden and E. M. Klimko. Academic Press, New York.

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

Golub, G. H., and C. F. Van Loan (1983). *Matrix Computations.* North Oxford Academic, Oxford.

Grefenstette, J. J., R. Gopal, B. J. Rossmalta and D. Vand Gucht (1985).

Genetic algorithms for the travelling salesman problem. *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, J. J. Grefenstette (Ed.), Lawrence Erlbaum Associates, Hillsdale, NJ, USA.

Haines, L. M. (1987). The application of the annealing algorithm to the construction of exact optimal designs for linear regression models. *Technometrics*, 29, pp. 439-447.

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

Johnson, D. S., C. R. Aragon, L. A. McGeogh and C. Schevon (1989). Optimization by simulated annealing: an experimental evaluation - Part 1: graph partitionning. *Operations Research*, 37, 6, pp. 865-892.

Khuri, A. I., and J. A. Cornell (1987). *Response Surfaces Designs and Analyses.* Marcel Dekker, Inc., New York.

Meyer, R. K., and C. J. Nachtsheim (1988). Constructing exact D-optimal experimental designs by simulated annealing. *American Journal of Mathematical and Management Sciences*, 8, pp. 329-359.

Mitchell, T. J. (1974). An Algorithm for the construction of D-optimal experimental design. *Technometrics*, 16, pp. 203-211.

Mühlenbein, H. (1989). Parallel genetic algorithms, population genetics and combinatorial optimization. *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, F. D. Schaffer (Ed.), Morgan Kaufmann Publ., San Mateo, USA.

St. John, R. C., and N. R. Draper (1975). D-optimality for regression designs: a review. *Technometrics*, 17, pp. 15-23.

Sanchez, R. P. (1992). *Application des Algorithmes Génétiques à la Recherche de Plans d'Expériences D-Optimaux.* Graduating Dissertation, Université Libre de Bruxelles, Brussels.

Syswerda, G. (1989). Uniform cross-over in genetic algorithms. *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, F. D. Schaffer (Ed.), Morgan Kaufmann Publ., San Mateo, USA.