Genetic crossover operator for partially separable functions

Nicolas Durand
Laboratoire d'Optimisation Globale
Centre d'Etudes de la Navigation Aérienne
7, av Edouard Belin
31055 Toulouse cedex France
durand@cenatls.cena.dgac.fr

Jean-Marc Alliot
Laboratoire d'Optimisation Globale
Centre d'Etudes de la Navigation Aérienne
7, av Edouard Belin
31055 Toulouse cedex France
alliot@recherche.enac.fr

ABSTRACT

Partial separation is a mathematical technique that has been used in optimization for the last 15 years. On the other hand, genetic algorithms are widely used as global optimizers. This paper investigates how partial separability can be used in conjunction with GA. In the first part of this paper, a crossover operator designed to solve partially separable global optimization problems involving many variables is introduced. Then, a theoretical analysis is presented on a test case, along with practical experiments on fixed size populations, with different kinds of selection methods.

Introduction

This paper deals with minimization problems for which the function to minimize is the sum of many positive terms, each term only involving a subset of variables. Griewank and Toint gave first definition of such functions (i.e partially separable functions) about 15 years ago (Griewank and Toint 1982). Properties of partial separability are commonly used to solve local optimization problems.

On the other hand, it is quite widely accepted that on real problems, when a crossover operator that makes sense can be defined, the efficiency of GA (compared to other stochastic optimization methods) is closely related to the use of the crossover operator: considering large size problems, the use of random crossover operator can penalize GAs performance. Classical crossover operators described in the literature (Goldberg 1989, Michalewicz 1992, Holland 1975) create two children from two parents chosen in the population. Initial operators on bit strings simply cut the two parents strings in two parts. The main drawback of these operators is that short schemes have a greater probability to survive than long ones. Consequently the encoding problem becomes very important. Multi-points crossovers and Gray codes are introduced to solve this problem. When using real variable coding, programmers very often use arithmetic crossover. However, none of these methods recognizes and favors good schemes. Heuristics are sometimes used to favor "good" crossovers and "good" mutations (Ravise *et al.* 1995) or try to reduce disruption of superior building blocks (Corcoran and Wainright 1996).

We will show in this paper that taking advantage of the structure of partially separable functions to define a new crossover operator improves the converging speed and converging rate of a genetic algorithm.

1 Partial separability

1.1 Definition

Partially separable problems considered in this paper have the following characteristics: the function F to minimize depends on n variables x_1, x_2, .., x_n (n large) and is a sum of m positive functions F_i involving only a subset of variables. Partially separable functions can be written:

$$F(x_1, x_2, .., x_n) = \sum_{i=1}^{m} F_i(x_{j_1}, x_{j_2}, .., x_{j_{n_i}})$$

1.2 Adapted crossover principle

The crossover operator introduced in this article does not require any particular coding. The chromosome is directly coded with the variables of the problem (these variables may be real, integer,...). The intuitive idea is the following: for a completely separable problem, optimizing the global function can be done by optimizing each variable separately. This strategy is adapted to partially separable functions. When creating a child from two parents, the idea is to take for each variable the one that locally fits better (more or less Δ, where Δ controls the determinism of the operator).

First, we define a *local fitness* $G_k(x_1, x_2, .., x_n)$ for variable x_k as follows:

$$G_k(x_1, x_2, .., x_n) = \sum_{i \in S_k} \frac{F_i(x_{j_1}, x_{j_2}, .., x_{j_{n_i}})}{n_i}$$

where S_k is the set of i such as x_k is a a variable of F_i and n_i the number of variables of F_i.

Intuitively, the local fitness associated to a variable isolates its contribution to the global fitness[1]. Furthermore, it has the following good property:

$$\sum_{k=1}^{n} G_k(x_1, x_2, .., x_n) = F(x_1, x_2, .., x_n)$$

When minimizing F, if:

$$G_k(parent_1) < G_k(parent_2) - \Delta$$

then child 1 will contain variable x_k of parent 1. Else, if:

$$G_k(parent_1) > G_k(parent_2) + \Delta$$

then child 1 will contain variable x_k of parent 2. If:

$$|G_k(parent_1) - G_k(parent_2)| \leq \Delta$$

then variable x_k of child 1 will be randomly chosen, or can be a random linear combination of the k^{th} variable of each parent when dealing with real variables. If the same strategy is applied to child 1 and to child 2, children may be identical, especially if Δ is small. This problem can be avoided by taking a new pair of parents for each child.

Let's consider the following completely separable function:

$$F(x_1, x_2, x_3) = x_1 + x_2 + x_3$$

for x_1, x_2 and x_3 integers include in $[0, 2]$. Variable k's local fitness is: $G_k(x_1, x_2, x_3) = x_k$. Let's cross parents $(1\ ,\ 0\ ,\ 2)$ and $(2\ ,\ 1\ ,\ 0)$ which have the same fitness $F = 3$. With $\Delta = 0$, child 1 will be $(1\ ,\ 0\ ,\ 0) : F = 1$. With $\Delta = 1$, child 2 may be $(2\ ,\ 1\ ,\ 0)$, $(2\ ,\ 0\ ,\ 0)$, $(1\ ,\ 1\ ,\ 0)$, or $(1\ ,\ 0\ ,\ 0)$. The children's fitness are always better than the parents fitness when $\Delta = 0$ which is not the case with a classical crossover operator.

As it is completely separable, this function is obviously too simple to show the interest of the adapted crossover operator. In the next paragraph, a simple partially separable function is introduced and the improvement achieved is theoretically measured.

2 Theoretical study

Let's define the following function:

$$F(x_1, x_2, .., x_n) = \sum_{0<i\neq j\leq n} \delta(x_i, x_j) \qquad (1)$$

$(x_1, x_2, .., x_n)$ is a bit string and $\delta(x_i, x_j) = 1$ if $x_i \neq x_j$ and 0 if $x_i = x_j$. It must be noticed that the function is only partially separable and has 2 global minima, $(1, 1, 1, .., 1)$ and $(0, 0, 0, .., 0)$.

[1]There are often many different ways to define a local fitness that can favour more or less competing optima. Discussing this subject in the present paper would be too long.

For $x = (x_1, x_2, ..x_n)$, we define the local fitness $G_k(x)$ by:

$$G_k(x) = \frac{1}{2} \sum_{i=1}^{n} \delta(x_k, x_i)$$

We define $I(x)$ as the number of bits equal to 1 in x. Then, it is easy to establish that:

$$\begin{aligned} F(x) &= I(x)(n - I(x)) \\ G_k(x) &= \frac{I(x)}{2} \quad if\ x_k = 0 \\ &= \frac{n - I(x)}{2}\ if\ x_k = 1 \end{aligned}$$

In the following paragraphs, we use a classical n point crossover operator; A_1 and A_2 represent 2 parents randomly chosen in a population and C represents their child.

In paragraph 2.1, the probability of fitness increase when using the adapted or the classical crossover operator are compared. Then, the adapted crossover converging rate is computed in a simple genetic algorithm, with no selection and no mutation (paragraph 2.2). In paragraph 2.3, a GA with selection is used to compare the two crossovers

2.1 Probability of fitness increase

For function (1), the probability of fitness increase when using the classical or the adapted operator can be mathematically computed for every possible couple of parents.

Let's define $P_{1-1}(i, j, k)$ as the probability to find k bits equal to 1 at the same position in both parents A_1 and A_2, with $I(A_1) = i$ and $I(A_2) = j$. As $P_{1-1}(i, j, k) = P_{1-1}(j, i, k)$, we will suppose that $i \leq j$ in the following. It can be shown that:

- if $k > i$, then:
$$P_{1-1}(i, j, k) = 0$$
- if $k \leq i$, then:
$$P_{1-1}(i, j, k) = C_i^k \prod_{l=0}^{k-1} \frac{j-l}{n-l} \prod_{l=k}^{i-1} \frac{(n-l)-(j-k)}{n-l}$$

The classical crossover used is the n point crossover that randomly chooses bits from A_1 or A_2 (the order of the bit string has no influence on the fitness).

For the adapted crossover (respectively for the classical crossover), let's define $P_a(i, j, k)$ (resp. $P_c(i, j, k)$) as the probability that if $I(A_1) = i$ and $I(A_2) = j$ then $I(C) = k$. As $P_a(i, j, k) = P_a(j, i, k)$ and $P_c(i, j, k) = P_c(j, i, k)$, we will suppose that $i \leq j$ in the following. Then, it can be shown that for the classical crossover:

$$P_c(i, j, k) = \sum_{l=\max(0, \frac{i+j+1-n}{2})}^{\min(k, i+j-k)} P_{1-1}(i, j, l)\, \frac{C_{i+j-2l}^{k-l}}{2^{i+j-2l}}$$

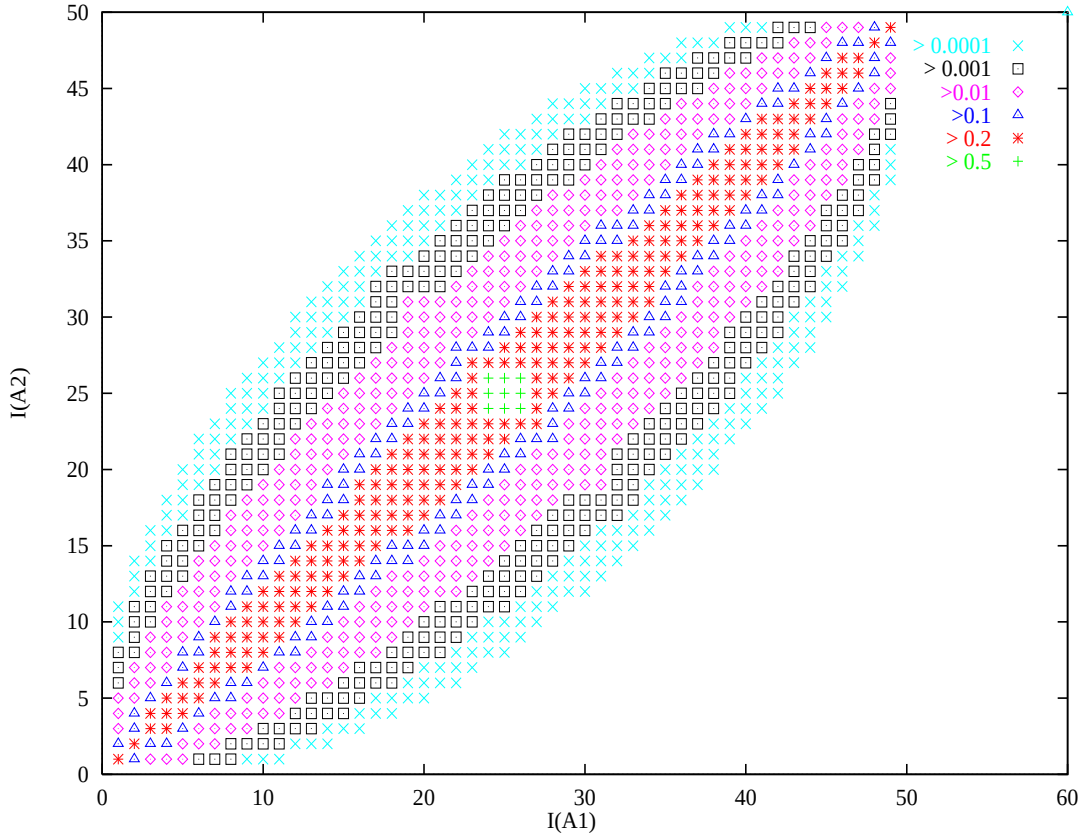

Figure 1 $Prob(F(C) > \max[F(A_1), F(A_2)])$ **function of** $[I(A_1), I(A_2)]$ **- classical crossover -** $n = 50$ **.**

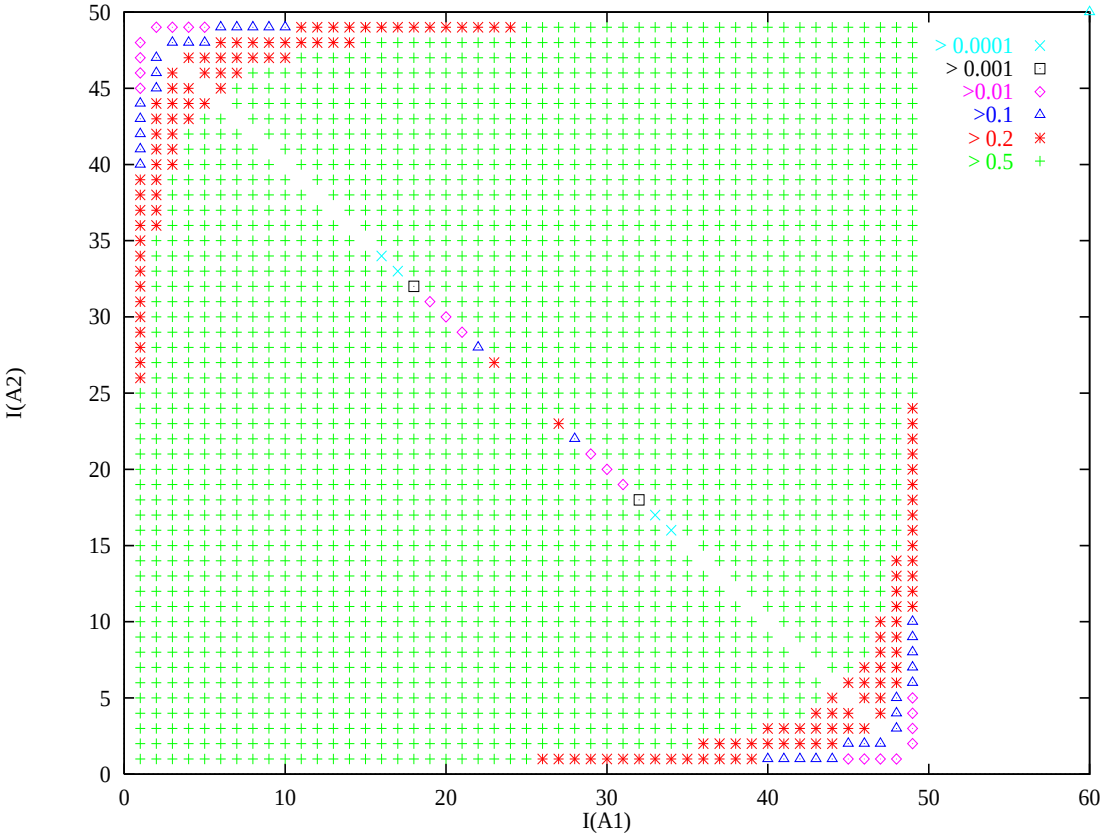

Figure 2 $Prob(F(C) > \max[F(A_1), F(A_2)])$ **function of** $[I(A_1), I(A_2)]$ **- adapted crossover -** $n = 50$ **.**

For the adapted crossover(with $m = \min(k, n-k)$):

$$\begin{aligned} i+j<n: \quad & P_a(i,j,k) = P_{1-1}(i,j,k) \\ i+j>n: \quad & P_a(i,j,k) = P_{1-1}(n-i,n-j,n-k) \\ i+j=n: \quad & P_a(i,j,k) = \sum_{l=0}^{m} P_{1-1}(i,j,l)\,\frac{C_{i+j-2l}^{k-l}}{2^{i+j-2l}} \end{aligned}$$

As $P_{1-1}(i,j,k) = 0$ if $k > \min(i,j)$, then:

- if $i+j<n$ and $k > \min(i,j)$, then $P_a(i,j,k) = 0$
- if $i+j>n$ and $k < \max(i,j)$, then $P_a(i,j,k) = 0$

Consequently:

- if $i+j<n$ and $P_a(i,j,k) > 0$, then $k < \min(i,j,n-i,n-j)$
- if $i+j>n$ and $P_a(i,j,k) > 0$, then $k > \max(i,j,n-i,n-j)$

Thus, if $i+j \neq n$, then $F(C) \geq \max[F(A_1), F(A_2)]$. If $i+j=n$, local fitness of variables of each parent are equal and the adapted crossover behaves like a classical n points crossover.

Figures 1 and 2 give the probability for a child to have a better fitness than its parents (for all the possible combinations of parents). On this example, the adapted crossover widely improves crossover efficiency. The small square in the center of figure 1 represents a probability of improvement larger than 0.5. It becomes a very large square on figure 2.

2.2 Adapted crossover converging rate

As P_a and P_c are known, it is possible to evaluate the probability to find an optimal solution after m generations. The GA only uses a crossover operator without any selection, or mutation operator. Let's define $Q_a(m,k)$ as the probability that $I(C) = k$ if C is a randomly chosen element at generation m. At generation 0:

$$Q_a(0,k) = \frac{C_n^k}{2^n}$$

At generation $m+1$:

$$Q_a(m+1,k) = \sum_{i=0}^{n}\sum_{j=0}^{n} Q_a(m,i)\,Q_a(m,j)\,P_a(i,j,k)$$

Figure 3 gives the evolution of $Q_a(m,0) = Q_a(m,n)$ for different values of n. It appears that the probability to find $(0,0,0..,0)$ or $(1,1,1..,1)$ stabilizes around $\frac{1}{3}$ after generation 7. It was decided that when local fitness are equal, the crossover used was the classical crossover. If it had been decided that each time local fitness are equal, the same parent would give its value to the child, then probabilities to find $(0,0,0..,0)$ or $(1,1,1..,1)$ would have stabilized around $\frac{1}{2}$. It is important to note that n has little influence on the convergence speed.

It is obviously useless to expect that the same GA using the classical crossover converges. In fact, we have:

$$\forall m \in N,\ Q_c(m,k) = \frac{C_n^k}{2^n}$$

2.3 Selection probability

As the fitness function takes its values in $[0, n^2/4]$, we will select chromosomes proportionally to $n^2/4 - F(A)$. Let's study the simple GA that selects individuals according to the previous criteria and then uses the adapted or classical crossover. Therefore, let's define $S_a(m,k)$ and $S_c(m,k)$ as the probability that $I(C) = k$ if C is a randomly chosen element at generation m, just after the selection operator.

Then, it is easy to prove that:

$$\begin{aligned} S_a(m+1,k) &= \frac{s_a(m+1,k)}{\sum_{k=0}^{n} s_a(m+1,k)} \\ S_c(m+1,k) &= \frac{s_c(m+1,k)}{\sum_{k=0}^{n} s_c(m+1,k)} \end{aligned}$$

with

$$H(k) = \frac{n^2}{4} - k\,(n-k)$$

$$s_a(m+1,k) = \sum_{i=0}^{n}\sum_{j=0}^{n} H(k)\,S_a(m,i)\,S_a(m,j)\,P_a(i,j,k)$$

$$s_c(m+1,k) = \sum_{i=0}^{n}\sum_{j=0}^{n} H(k)\,S_c(m,i)\,S_c(m,j)\,P_c(i,j,k)$$

On figure 4, the four curves on the top represent $S_a(m,0)$ for chromosomes of size 20, 50, 100 and 200 bits. The four curves on the bottom represent $S_c(m,0)$ for chromosomes of sizes 20, 50, 100 and 200.

The efficiency of the adapted crossover appears clearly, particularly for large sized chromosomes. The converging rate is not very dependent on the size of the chromosome with the adapted crossover, whereas it is very dependent on the size of the chromosome with the classical crossover.

3 Experimental tests

In the previous study, the size of the population was not limited. However, population size does have an influence on the converging rate. Moreover, the previous example has only 2 global optima and we did not measure the ability of the GA to find every global optimum. We are going to tackle these two issues in this section.

Let's consider the same function as previously:

$$F(x_1,..,x_n) = \sum_{1 \leq i \neq j \leq n} \delta(x_i, x_j) \qquad (2)$$

but with x_i integer in $[0,4]$.

This function has 5 optima defined by $\forall (i,j)\ x_i = x_j$. In paragraph 3.1, classical and adapted crossover operators are compared in a simple GA, similar to the theoretical model.

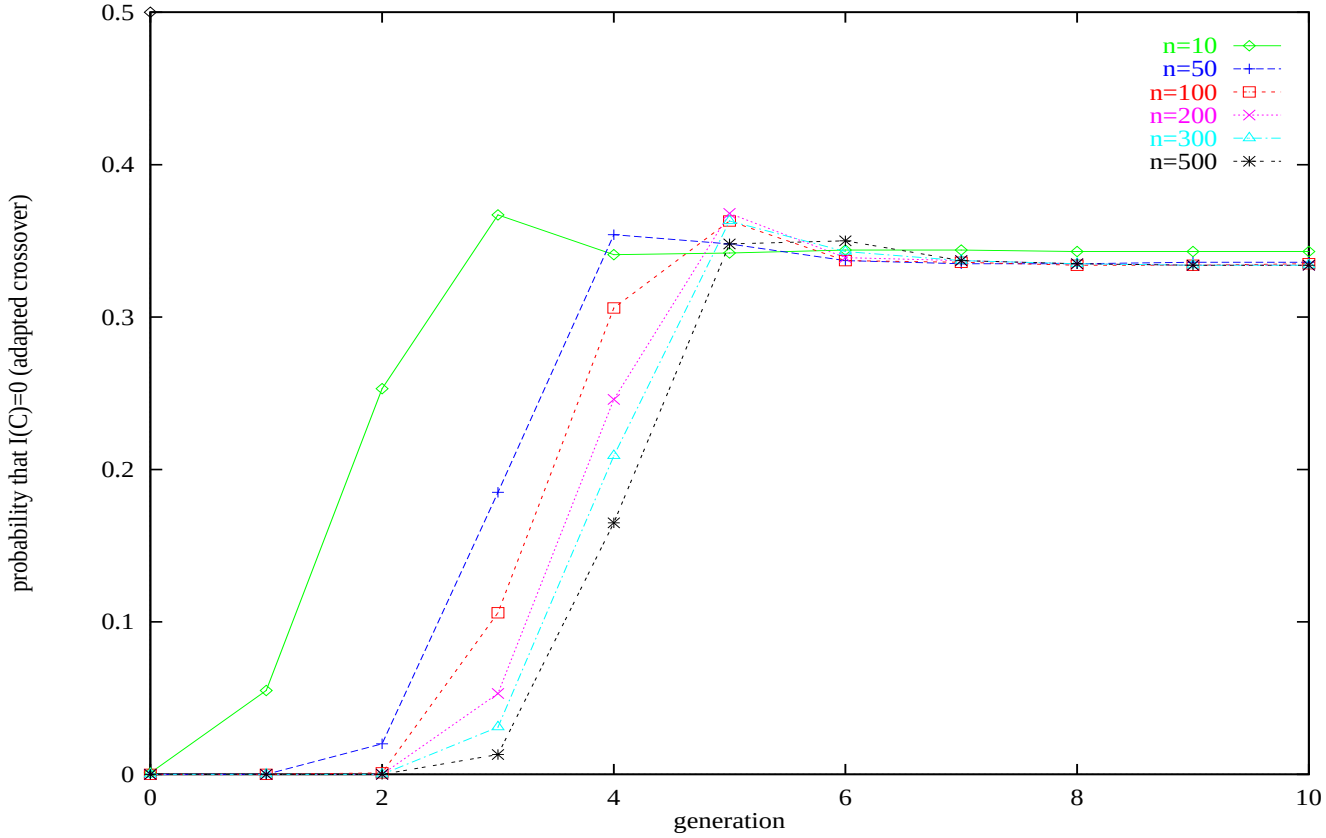

Figure 3 $Q_a(m,0)$ **as a function of the generation** m **for different values of** n**.**

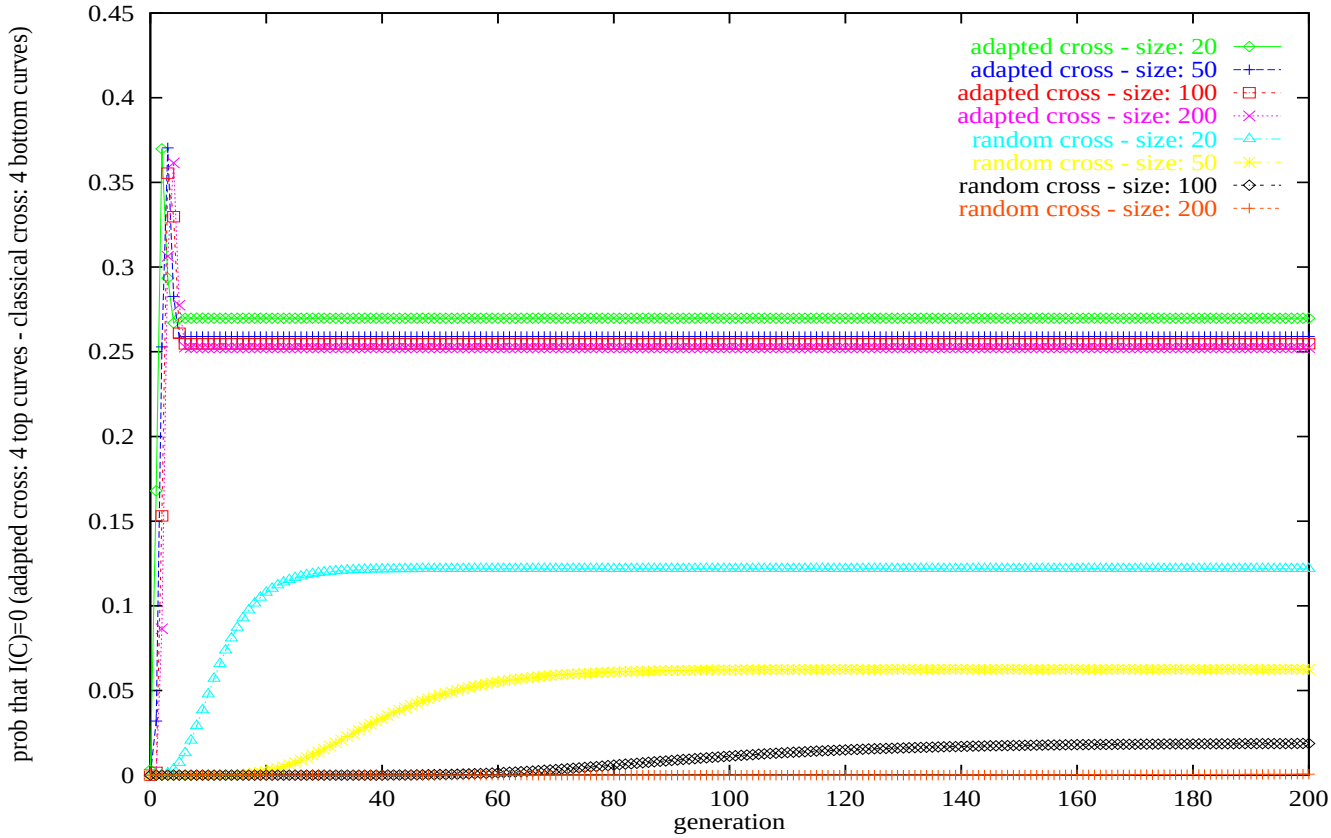

Figure 4 $S_a(m,0)$ **and** $S_c(m,0)$ **as a function of the generation** m **for different values of** n**.**

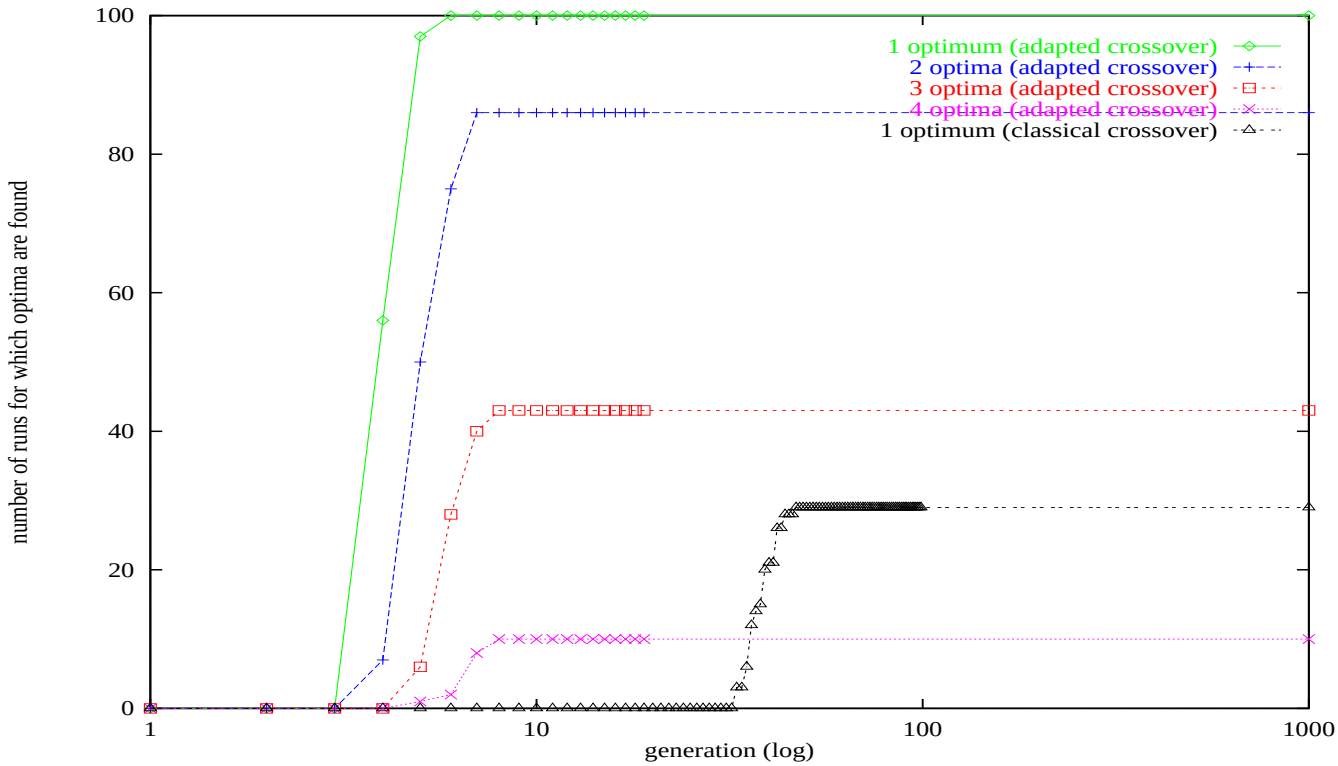

Figure 5 Number of runs for which 1, 2, 3 or 4 optima are found with the adapted crossover, 1 optimum is found with the classical crossover, as a function of the generation.

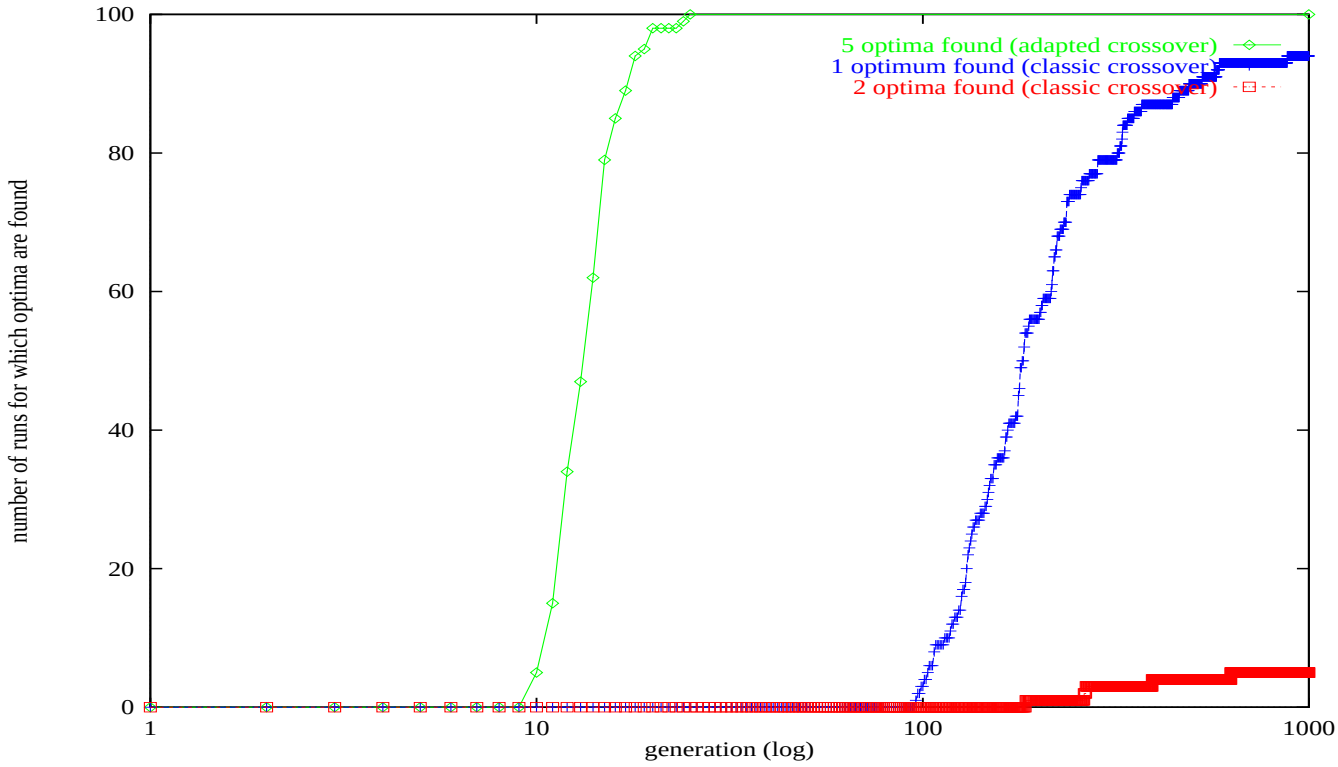

Figure 6 Number of runs for which 5 optima are found with the adapted crossover, 1, or 2 optima are found with the classical crossover, as a function of the generation.

The number of optima found and the converging speeds are compared. In paragraph 3.2, a sharing operator is added to find as many optima as possible. The efficiency of classical and adapted crossover are also measured.

3.1 GA without sharing

Function (2) is optimized with a GA on 200 runs (generations: 1000, population size: 500, proportion crossed: 50%, selection: stochastic reminder without replacement, elitism: yes, sharing: no, mutation:no). The first 100 tests are done with a classical n point crossover[2]. The 100 other tests are done with the adapted crossover previously described.

Figure 5 gives the number of runs for which 1, 2, 3 or 4 optima are found when using the adapted crossover. It is important to note that the 5 optima are never found. The figure also gives the number of runs for which 1 optimum is found when using the classical crossover. The GA with classical crossover never finds 2 optima.

3.2 GA with sharing

The GA is performed on the same function, but with a sharing process. Therefore, a distance between two chromosomes is defined. Let's define $h(C_1, C_2)$ as the Hamming distance between chromosome 1 and 2. If n is the size of the chromosome, then let's define:

$$Dist(C_1, C_2) = \begin{cases} 1\,, & \text{if } h(C_1, C_2) < \dfrac{n}{2} \\ 0\,, & \text{otherwise} \end{cases}$$

and $N(C_i)$ the number of chromosomes C_j in the population for which $Dist(C_i, C_j) = 0$. The sharing process used in the following divides the fitness of chromosome C_i by $N(C_i)$ before applying the selection process.

Figure 6 gives the number of runs for which the 5 optima are found when using the adapted crossover and sharing. The figure also gives the number of runs for which 1 or 2 optima are found when using the classical crossover and sharing. The GA with classical crossover and sharing never finds more than 2 optima.

3.3 Influence of sharing

A "good" crossover can be defined as a crossover for which the fitness of the child is better than the fitness of both parents. In order to measure the influence of sharing on the crossover's efficiency, figures 7 (resp 8) give the percentage of "good" crossovers with the classical (resp adapted) operator without sharing and with sharing.

Figure 7 shows that the sharing process delays the convergence to the optimum. Comparing figures 5 and 6 shows that the converging rate is penalized by the sharing process. However, it is difficult to say if the sharing process has an influence on the efficiency of the crossover.

Figure 8 shows that the adapted operator is all the more efficient because population is diversified. Figure 6 shows that the 5 optima are always found before generation 20. Its seems that the sharing process has little influence on the converging rate.

[2]The n point crossover takes into account the fact that the order of the variables has no importance

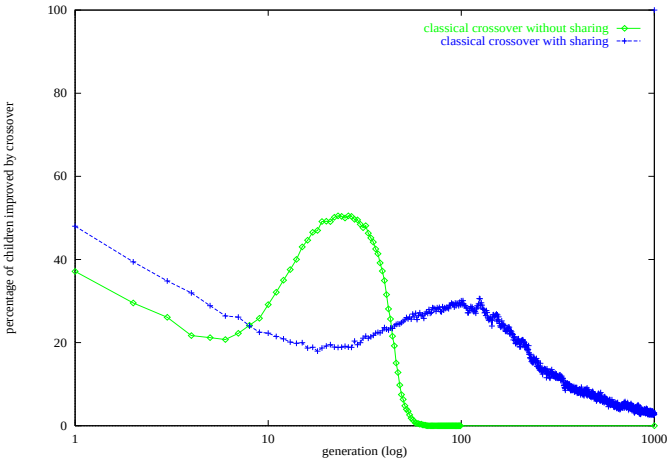

Figure 7 percentage of improved chromosome with the classical crossover without and with sharing.

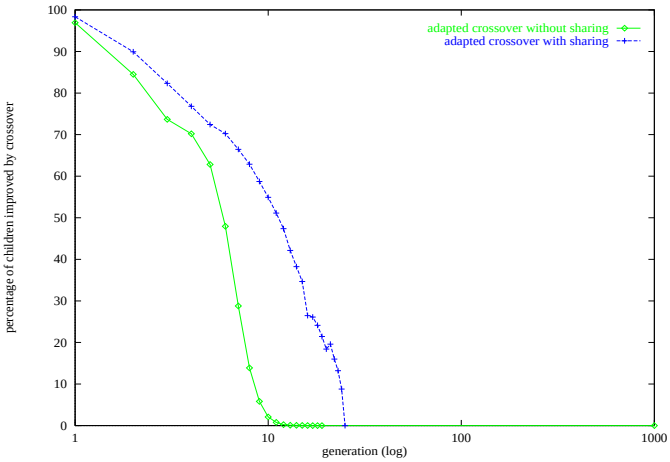

Figure 8 percentage of improved chromosome with the adapted crossover without and with sharing.

4 Classical test functions

4.1 Corana's function

This function is presented in (Corana *et al.* 1987). We use here the restriction used by Ingber in its article (Ingber and Rosen 1992). The function must be optimized on $[-10000, 10000]^N$. It is defined by :

$$f_0(x) =$$

$$\sum_{i=1}^{N} \left\{ \begin{array}{ll} 0.15\, d_i (0.05\, S(z_i) + z_i)^2 & \text{for } |x_i - z_i| < 0.05 \\ d_i\, x_i^2 & \text{otherwise} \end{array} \right\}$$

$$z_i = 0.2\, \lfloor |x_i/0.2| + 0.49999 \rfloor\, S(x_i)$$

$$S(z_i) = \begin{cases} 1 & \text{if } z_i > 0 \\ 0 & \text{if } z_i = 0 \\ -1 & \text{if } z_i < 0 \end{cases}$$

	min	max	mean	σ
classical	76	294	179.44	178.47
adapted	42	204	92.33	91.81
classical / sharing	79	357	248.87	247.32
adapted / sharing	42	186	96.57	95.95

Table 1 Convergence for Griewank's function

$$d_{i \bmod 4} = \{1.0, 1000.0, 10.0, 100.0\}$$

This function has 10^{5N} local optima and all points of $[-0.05, 0.05]^N$ are global optima. Ingber presents this function as an excellent test for all global optimization techniques, and it is interesting to compare GA with adapted crossover with VFSR, which is currently the best simulated annealing technique. It is easy to define a local fitness for adapted crossover as the function is completely separable.

$$G_i(x_i) = \begin{cases} 0.15\, d_i (0.05\, S(z_i) + z_i)^2 & , |x_i - z_i| < 0.05 \\ d_i\, x_i^2 & \text{otherwise} \end{cases}$$

As it could have been expected, the adapted crossover is of course very efficient: both classical GA and VFSR fail in finding an optimum for $N > 24$; the GA with adapted crossover finds the optimum up to $N = 1000$.

4.2 Griewank's function

This function is much more interesting than Corana's, as it is not completely separable. It is defined by:

$$F(x_1, .., x_n) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$$

We will use here $n = 10$. This function has only one global minimum, for $x = (0, \ldots, 0)$. We define the local fitness by:

$$G_i(x_1, .., x_{10}) = \frac{1}{4000}\, x_i^2 - \prod_{i=1}^{10} \cos(\frac{x_i}{\sqrt{i}})$$

4 series of 100 tests were conducted: (classical crossover, no sharing), (classical crossover, sharing), (adapted crossover, no sharing), (adapted crossover, sharing). We say that the GA has found the optimal solution when a local optimization function applied to the best element finds the optimum. Results are presented on table 1. They give the min, max, mean and standard deviation number of generations to find the optimum. The adapted crossover is clearly much more efficient than the classical one.

5 Conclusion

The crossover operator introduced in this paper can be adapted to various global optimization problems, and specially difficult problems with many variables. The method was in fact developed during a work conducted jointly with the French Civil Aviation Administration. The goal was to build a system able to solve automatically air traffic conflict involving up to 30 planes with real traffic. The problem involves up to 90 variables. The GA with adapted crossover outperformed every other method tried (simulated annealing, A^*, B&B based on interval programming). It was able to tackle the problem while classical GA were limited to problems involving less than 10 planes (30 variables) (Durand *et al.* 1996, Durand 1996).

In this paper, we hadn't the place to discuss the influence of parameter Δ. It can be used to control the determinism of the adapted crossover. Too small values can push the GA into local minima, while too large ones slow down the convergence. It seems that it should decrease while the algorithm converges as, at the beginning, the space is explored randomly and at the end, the algorithm becomes more deterministic.

References

Corana, A., M. Marchesi, C. Martini and S. Ridella (1987). Minimizing multimodal unctions of continuous variables with the "simulated annealing" algorithm. In: *Proceedings of the ACM Transaction and Mathematical Software*. ACM.

Corcoran, Arthur L. and Roger L. Wainright (1996). Reducing disrupton of superior building blocks in genetic algorithms. In: *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM.

Durand, Nicolas (1996). Optimisation de Trajectoires pour la Résolution de Conflits en Route.. PhD thesis. ENSEEIHT, Institut National Polytechnique de Toulouse.

Durand, Nicolas, J. M. Alliot and Joseph Noailles (1996). Automatic aircraft conflict resolution using genetic algorithms. In: *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM.

Goldberg, David (1989). *Genetic Algorithms*. Addison Wesley. ISBN: 0-201-15767-5.

Griewank, A. and Ph. L. Toint (1982). On the unconstrained optimization of partially separable functions. In: *Nonlinear Optimization 1981* (M. J. D. Powell, Ed.). Academic Press. London and New York. pp. 301–312.

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

Ingber, Lester and Bruce Rosen (1992). Genetic algorithm and very fast simulated reannealing: a comparison. *Mathematical and Computer Modeling* **16**(1), 87–100.

Michalewicz, Z. (1992). *Genetic algorithms+data structures=evolution programs*. Springer-Verlag. ISBN: 0-387-55387-.

Ravise, C., Michele Sebag and Marc Schœnauer (1995). Optimisation guidée par l'induction. In: *Proceedings of the Journees Evolution Artificielle Francophones*. EAF.