GAs for partially separable functions

Nicolas Durand, Jean-Marc Alliot

CENA*

Abstract. In this paper, a crossover technique for GAs is introduced to solve partially separable global optimization problems involving many variables. Therefore, the fitness function must be the sum of positive sub-functions involving only a subset of the variables. A "local fitness" is associated to each variable and a parameter Δ controlling the operator's determinism is introduced. Combined with sharing, this operator improves GAs efficiency to optimize combinatorial problems involving many variables. A polynomial function is first given as an example and the operator is then applied to different classical test functions for which the number of variables was increased. A practical application of this operator is given for solving conflicts involving many aircraft for air traffic control.

1 Introduction

In many optimization problems, the fitness function involves many variables. Sometimes, it is the sum of many positive terms, each term only involving a subset of the variables. GAs are good global optimization tools because they can avoid local minima. However, considering large size problems, performance of GAs can be penalized because they use random crossover and mutation operators. They are very efficient at the beginning to find good areas. Their efficiency can decrease rapidly when converging because of the random operators. This phenomenon particularly occurs for large size combinatorial problems.Genetic programmers sometimes use heuristics to favor "good" crossovers and "good" mutations [RSS95] or try to reduce disruption of superior building blocks [CW96]. This paper introduces a genetic operator for large size optimization problems in which the fitness function is the sum of positive functions involving only a subset of the variables.

In the first part, an adapted crossover to partially separable problems is defined. A simple example of its efficiency is given in the second part. In part three, the operator is tested on classical test functions and compared to classical methods. The last part gives a practical application of this operator. It is indeed very useful to solve problem such as conflict resolution involving many aircraft in Air Traffic Control.

* Centre d'Etudes de la Navigation Aérienne, durand@.cena.dgac.fr, alliot@dgac.fr

2 Adapted crossover to partially separable problems

In partially separable problems, the fitness 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. Many multiplicative functions can be transformed into additive function very simply. So partially separable functions as defined above can be written :

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

In order to define the crossover operator, a "local fitness" $G_k(x_1, x_2, .., x_n)$ will be introduced for each variable x_k as follow :

$$G_k(x_1, x_2, .., x_n) = \sum_{i \in E_k} F_i(x_{j_1}, x_{j_2}, .., x_{j_{n_i}})$$
$$E_k = \{i / x_k \; is \; a \; variable \; of \; F_i\}$$

The local fitness associated to a variable isolates its contribution to the global fitness. The goal of this paper is to use the local fitness to define an improved crossover operator. Combined with sharing this operator is very efficient and allows to use only low population sizes for problems involving a large number of variables.

Classical crossover operators 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 drawback of these operators is that short schemes have more chance to survive than long ones. Consequently the problem coding becomes very important. Multi-points crossovers and Gray coding are very often used to solve this problem. However, none of these methods recognizes and favors good schemes. When using real variable coding, programmers very often use barycentric crossover (the children are linear combinations of the 2 parents). These techniques do not distinguish good crossovers and bad crossovers.

The adapted crossover does not require any particular coding. The chromosome is directly coded with the variables of the problem (these variables may be real, integer or both).The intuitive idea is the following : for a totally separable problem, optimizing the global function can be done by optimizing each variable separately. The idea is to adapt this strategy 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). For example, if we minimize F, for the k^{th} variable.

If $G_k(parent_1) < G_k(parent_2) - \Delta$, then child 1 will contain variable x_k of father 1. If, on the contrary $G_k(parent_1) > G_k(parent_2) + \Delta$, child 1 will contain variable x_k of father 2. If $|G_k(parent_1) - G_k(parent_2)| \leq \Delta$, variable x_k of child 1 will be randomly chosen, or can be a random linear combination of variables x_k of the two parents if we are dealing with real variables. If the same strategy is applied to child 1 and to child 2, the two children may be identical, especially if Δ is small. Different strategies can be imagined, for example, different values of Δ can be chosen for the two children.

It can be noted that it is easy to design a mutation operator following the same idea. This operator would mutate with a greater probability variables with a bad local fitness. This assumes to be able to compare local fitness between variables.

In this paper, classical GAs such as described in the literature [Gol89, Mic92, Hol75] are used. A sharing process may be very useful when the population size is not important enough regarding to the size of the problem. The main drawback of sharing methods is that they slow down the GA. Yin and Germay [YG93] have created a clustering method that is less expensive (proportional to $n\log(n)$) and will be used.

3 Polynomial example

To show the crossover operator efficiency, we build a very simple polynomial example as follow :

$$F(x_1, x_2, .., x_n) = \sum_{i \neq j} (x_i - x_j)^2$$

Local fitness is defined by : $G_k(x_1, x_2, .., x_n) = \sum_{i \neq k} (x_i - x_k)^2$. We are looking for the minimum of the polynomial for integers $x_i \in [0, 20]$.

In order to test the crossover operator efficiency, two test bench are done with $n = 50$. The two test bench do not use any sharing process. The first one uses a classical 1 point crossover operator. The second one uses the crossover operator described in part three. Each test is repeated a hundred times with different random number generators[2]. For practical reasons, instead of minimizing F, we maximized $\frac{1}{1+F}$.

Results are given in table 1 (computation done on HP 720) (table 1 gives the minimum, maximum, and average time to converge to an optimal solution, the minimum, maximum and average number of generations corresponding and the average fitness of the populations). They show that the adapted crossover operator makes the algorithm 3 times faster.

Another comparative test was done to show how this adapted operator can be used with sharing. In this test bench $n = 20$. The problem has 21 global optima. The research space size is 21^{20} (i.e. more than 10^{27}). Two simulations are done. The first one uses a classical crossover with sharing and the second one uses the adapted crossover with sharing[3].

Figure 1 gives the evolution of the best element in the two simulations. Using the adapted crossover operator makes the algorithm converge much more rapidly to an optimal solution despite the use of sharing which usually reduces the

[2] Population size : 100, $P_c = 60\%$, $P_m = 15\%$

[3] 100 generations, 600 population elements, $P_c = 60\%$, $P_m = 15\%$

Crossover	classical	adapted
Min generations	94	20
Average generations	131.6	38.27
Max generations	180	55
Min time (in sec)	55	12
Average time (in sec)	78.08	21.76
Max time (in sec)	110	30
Average fitness	0.3458	0.3427

Table 1. Numerical Results without sharing, $n = 50$.

converging speed. The optimum is obtained at generation 22 instead of 63 for the classical crossover operator.

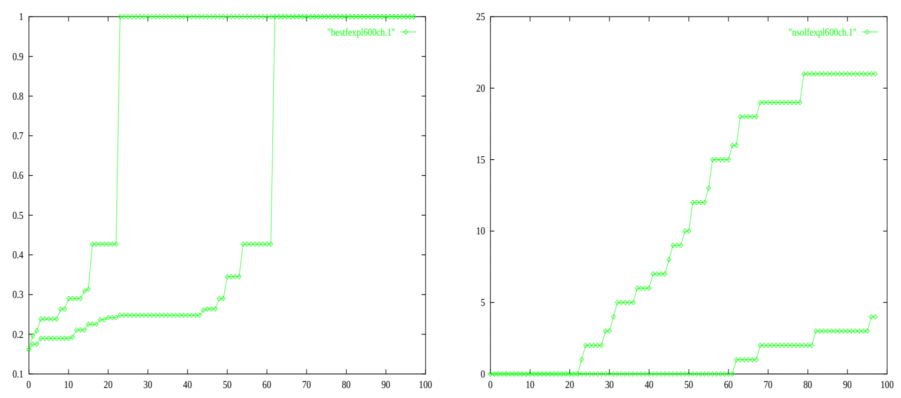

Fig. 1. Maximum fitness (left) and Optimal solution numbers (right), function of the generation number, adapted crossover with sharing (up), classical crossover with sharing (down).

The most interesting results are presented in figure 1 that gives the number of optimal solutions in the two cases. The efficiency of combining the adapted crossover with a sharing process is very clear: with the adapted crossover and a sharing process the 21 global optima are found. With sharing and classical operators, only 4 optima are found.

The previous simulations showed that the crossover operator using local fitness can be very interesting when combined with a sharing process to find the different global optima.

4 Classical test functions

Global optimization algorithm are generally tested on classical test functions [IR92]. We have chosen two of them the size of which could be extended. These two functions have many local optima but the first one is completely separated and the second one is not. GAs using classical operators and adapted operators are compared on these two functions with VFSR[4] (or ASA[5])).

4.1 Corana's function

This fonction is detailed in [CMMR87]. Results obtained by Ingber [IR92] with VFSR are used to compare GAs to simulated annealing. The following function is being minimized on the compact $[-10000, 10000]^N$.

$$
\begin{aligned}
F(x_1, \ldots, x_N) &= \sum_{i=1}^{N} \begin{cases} 0.15\, d_i (0.05\, S(z_i) + z_i)^2 & \text{if } |x_i - z_i| < 0.05 \\ d_i\, x_i^2 & \text{otherwise} \end{cases} \\
z_i &= 0.2 \left\lfloor |x_i/0.2| + 0.49999 \right\rfloor S(x_i) \\
S(z_i) &= \begin{cases} 1 & \text{if } z_i > 0 \\ 0 & \text{if } z_i = 0 \\ -1 & \text{if } z_i < 0 \end{cases} \\
d_i &= \{1.0, 1000.0, 10.0, 100.0\}
\end{aligned}
$$

This function has 10^{5N} local optima. According to L. Ingber, this function is one of the best possible test for a global optimization algorithm. It seemed interesting to compare our results to those obtained with VFSR on this example. This function is completely separated.

Good results are obtained with VFSR and classical GAs on this function for $N = 4$ and $N = 8$. For $N = 12$, the problem becomes more difficult. Default parameters of VFSR do not make the algorithm converge.

With the help of L. Ingber, it was possible to find parameters making VFSR converge for $N = 12$ (A technique named "Quenching" was used to accelerate the reannealing scheme). 5 out of the 10 tests that were done gave the global minimum, 5 other tests gave local minima. The GA always gave the global minimum before generation 300 when using a population of 400 elements. When it converges, VFSR is faster than the GA.

Then we decided to increase the value of N to compare the behaviour of both algorithms. With VFSR and $N = 20$ the global minimum was never found (for 50 test done). With classical GA we could easily go to $N = 24$ but then it became difficult to find the global minimum.

In order to introduce the adapted operator in the GA, we defined the local fitness as follow :

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

[4] Very Fast Simulated Reannealing

[5] Adaptive Simulated Annealing

As the function is totally separated, we could expect that this operator would be very efficient. Using a population of 400 hundred elements on 100 tests confirmed that for $N = 1000$ the GA using the adapted crossover converges in less than 100 generations.

4.2 Griewank's function

Griewank's function is interesting because it can not be separated. For $N = 10$, it is defined as follows :

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

A local fitness can be defined by : $G(x_1, .., x_{10}) = \frac{1}{4000} x_i^2 - \prod_{i=1}^{10} \cos(\frac{x_i}{\sqrt{i}})$. 2 sets of tests have been done on this function using classical and adapted crossover. In each case, the GA was executed 100 times with a population of 100 elements and 600 generations. Table 2 gives the minimal, maximal and mean number of generations necessary to make the GA find the global minimum. This table also gives the standard deviation of the number of generations required.

N=10	min	max	moy	σ	N=20	min	max	moy	σ
classical cross	76	294	179.44	178.47	classical cross	204	554	440.10	220.2
adapted cross	42	204	92.33	91.81	adapted cross	90	420	230.3	151.2

Table 2. $N = 10$, population :100 - $N = 20$, population : 200

The results show that using the adapted crossover makes the algorithm converge twice as fast as using a classical barycentric crossover. Tests that were done using $VFSR$ did not converge to the optimum[6].

When increasing the size of the problem, the GA still converges up to $N = 20$ (see table 2) and using the adapted crossover still makes it twice as fast as using the classical crossover.

These two examples seem to show that the adapted operator is more efficient when the problem is "more separated".

5 Air traffic control problem

This example shows that we can find real problems on which the crossover operator we defined is very efficient. Studies on the use of GAs for conflict resolution and air space sectoring have given promissing results [AGS93, DASF94a, DASF94b, DAAS94, DAN96, vKHHK95].

[6] Ajusting parameters for $VFSR$ is difficult which may explain why these tests failed

5.1 Problem description

The problem takes place in an horizontal plane. n aircraft are flying at constant altitude and constant speed from an original point to a destination point. Two aircraft are said in conflict if at some time t the distance between these two aircraft is less than d.They are separated if they are not in conflict. The problem is to simultaneously minimize the aircraft trajectory length and separate aircraft. This problem is very complex [DAAS94] and cannot be solved using classical optimization methods because it is too combinatorial.

The model we chose to solve conflict is the following (figure 2) : an aircraft can be ordered to modify its heading at time t_0. This modification can take different values and will be noted dir (dir = -30, -20, -10, 10, 20, or 30 degrees). At time t_1, the aircraft turns to its initial heading. At time t_2, the aircraft heading is modified in the opposite direction of dir degrees and at time $t_3 = t_2 + (t_1 - t_0)$, the aircraft turns to its initial heading. An example of chromosome is given in figure 2. Each aircraft represents a $4D$ variable.

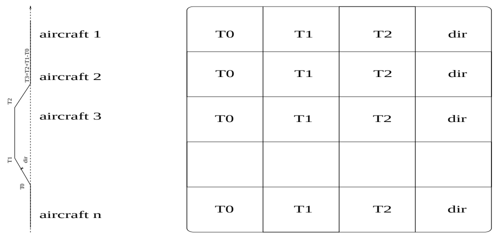

Fig. 2. The Model (left) - Structure of the chromosome.

5.2 Local and global fitness

Instead of considering a global fitness value that takes into account the different lengthening of the trajectories and the conflicts between the aircraft, we keep in a n^2 sized matrix F (where n is the number of aircraft) these values : if $i \neq j$, $F_{i,j}$ measures the conflict between aircraft i and j. It is set to 0 if no conflict occurs and increases with the duration of the conflict. $F_{i,i}$ measures the lengthening of aircraft i trajectory.

The global fitness is calculated as follow :

$$\exists (i,j),\ i \neq j,\ F_{i,j} \neq 0 \Rightarrow F = \frac{1}{2 + \sum_{i \neq j} F_{i,j}}$$
$$\forall (i,j),\ i \neq j,\ F_{i,j} = 0 \Rightarrow F = \frac{1}{2} + \frac{1}{2 + \sum_i F_{i,i}}$$

This fitness function guarantees that if a chromosome value is larger than $\frac{1}{2}$, no conflict occurs. If a conflict remains the fitness does not take into account the delays induced by maneuvers.

We can define for each aircraft its local fitness by : $G_i = \sum_{j=1}^{n}(F_{i,j})$.

5.3 Numerical results

To show the efficiency of the adapted operator combined with a sharing process, we will solve two conflicts, each one involving 20 aircraft. The first one (figure 3) deals with 20 aircraft on a circle. It has very few solutions because all the aircraft are conflicting with each other. Only solutions where all the aircraft turn in the same direction are conflict free. Figure 3 gives the result of the optimization[7].

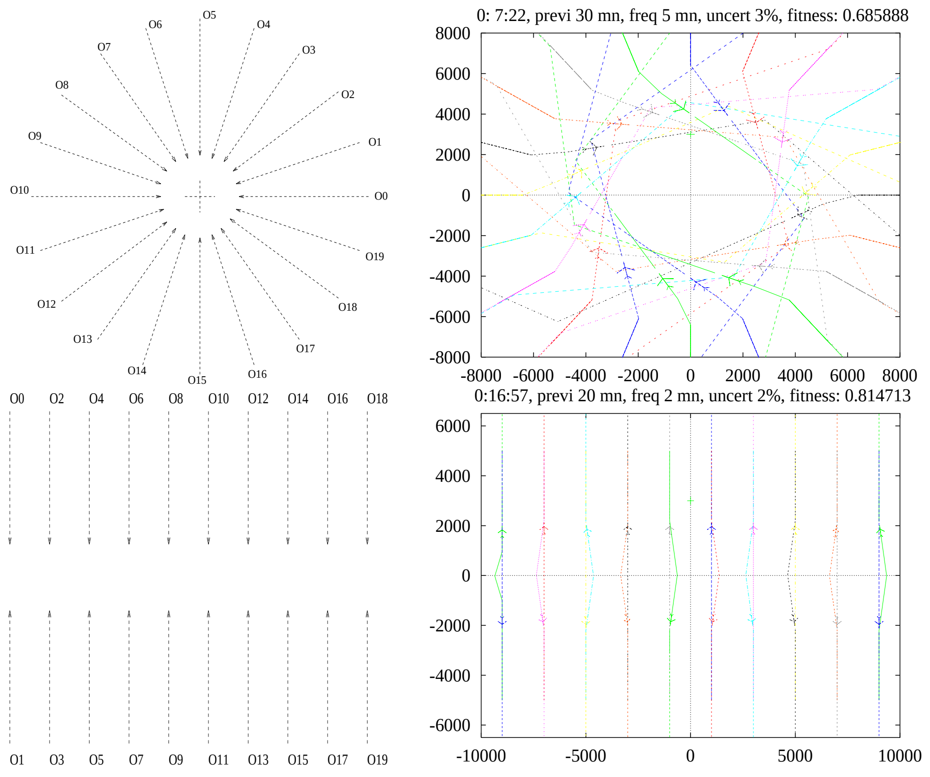

Fig. 3. 20 aircraft conflict (up : aircraft on a circle, down : 10 different conflicts)

The second conflict deals with 10 independent conflicts between two aircraft. In this example there are many available solutions which can easily be found by a human being because of the symmetries of the problem, but for the GA, the complexity remains. Figure 3 gives the result of the optimization with the same parameters as previously.

[7] 20 generations, 100 population elements, $P_c = 60\%$, $P_m = 15\%$, sharing used

In order to validate the use of the adapted operators combined with a sharing process on this problem, 4 different tests have been done on this two examples using classical and adapted crossover with and without sharing. For these 4 tests, we have measured on each example the value of the best fitness (figure 4) on the 20 generations of our GA.

First, we can check that adapted operators are very efficient because with classical operators, no conflict free solution is found before generation 20 (tests have shown that the first conflict free solution is found after generation 200) whereas with adapted operators a conflict free trajectory is always found before generation 10 (A solution is conflict free if its fitness is more than 0.5).

Figure 4 shows that the sharing process make the best fitness grow slower. However, the final best fitness is as good with a sharing process as it is without. It is even better with a sharing process when the conflict is very difficult to solve. The sharing process has the great advantage to help the GA to avoid local minima and to give different conflict free solutions. These tests justify the use of adapted operators in this partially separable problem.

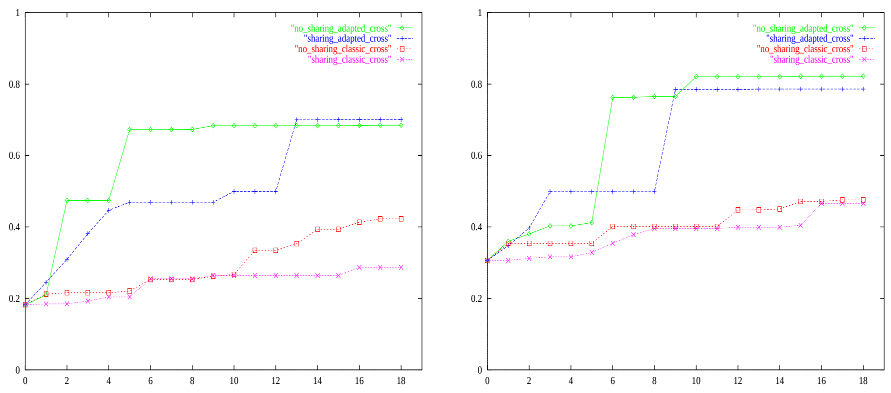

Fig. 4. Left : best fitness (aircraft on a circle) - Right : best fitness (10 different conflicts)

6 Conclusion

The crossover operator introduced in this paper can be adapted to various global optimization problems. The last combinatorial problem has shown that using local optimization operators was very helpful to find available solutions. The crossover operator can be used when the fitness function is partially separable and when a local fitness associated to variables can be defined. The operator determinism can be controlled by the parameter Δ. Most of the time, this parameter should be decreased while the algorithm converges so that at the beginning the space is explored randomly and at the end the algorithm becomes more determinist.

References

[AGS93] J. M. Alliot, Hervé Gruber, and Marc Schoenauer. Using genetic algorithms for solving ATC conflicts. In *Proceedings of the Ninth Conference on Artificial Intelligence Application*. IEEE, 1993.

[CMMR87] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal unctions of continuous variables with the "simulated annealing" algorithm. In *Proceedings of the ACM Transaction and Mathematical Software*. ACM, 1987.

[CW96] Arthur L. Corcoran and Roger L. Wainright. Reducing disrupton of superior building blocks in genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.

[DAAS94] Nicolas Durand, Nicolas Alech, J. M. Alliot, and Marc Schoenauer. Genetic algorithms for optimal air traffic conflict resolution. In *Proceedings of the Second Singapore Conference on Intelligent Systems*. SPICIS, 1994.

[DAN96] Nicolas Durand, J. M. Alliot, and Joseph Noailles. Automatic aircraft conflict resolution using genetic algorithms. In *Proceedings of the Symposium on Applied Computing, Philadelphia*. ACM, 1996.

[DASF94a] Daniel Delahaye, J. M. Alliot, Marc Schoenauer, and Jean-Loup Farges. Genetic algorithms for partitioning airspace. In *Proceedings of the Tenth Conference on Artificial Intelligence Application*. IEEE, 1994.

[DASF94b] Daniel Delahaye, J. M. Alliot, Marc Schoenauer, and Jean-Loup Farges. Genetic algorithms for air traffic. In *Proceedings of the Conference on Artificial Intelligence Application*. CAIA, 1994.

[Gol89] David Goldberg. *Genetic Algorithms*. Addison Wesley, 1989. ISBN: 0-201-15767-5.

[Hol75] J.H Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan press, 1975.

[IR92] Lester Ingber and Bruce Rosen. Genetic algorithm and very fast simulated reannealing: a comparison. *Mathematical and Computer Modeling*, 16(1):87–100, 1992.

[Mic92] Zbigniew Michalewicz. *Genetic algorithms+data structures=evolution programs*. Springer-Verlag, 1992. ISBN: 0-387-55387-.

[RSS95] C. Ravise, Michele Sebag, and Marc Schœnauer. Optimisation guidée par l'induction. In *Proceedings of the Journees Evolution Artificielle Francophones*. EAF, 1995.

[vKHHK95] C.H.M. van Kemenade, C.F.W. Hendriks, H.H. Hesselink, and J.N. Kok. Evolutionnary computation in air traffic control planning. In *Proceedings of the Sixth International Conference on Genetic Algorithm*. ICGA, 1995.

[YG93] Xiaodong Yin and Noel Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In C.R. Reeves R.F.Albrecht and N.C. Steele, editors, *In proceedings of the Artificial Neural Nets and Genetic Algorithm International Conference, Insbruck Austria*. Springer-Verlag, 1993.

This article was processed using the LaTeX macro package with LLNCS style