METHODS TO APPLY OPERATORS IN A STEADY STATE EVOLUTIONARY ALGORITHM

L.Gacôgne * **

* LIP6 - Université Paris VI 8 rue du Capitaine Scott 75015 Paris
tel : 01 44 27 88 07 fax : 01 44 27 88 02 mail : Louis.Gacogne@lip6.fr
** Institut d'Informatique d'Entreprise (CNAM) 18 allée J.Rostand 91025 Evry

Abstract
A particular steady-state strategy of evolution with a small sized population is studied in this paper. After comparison with GA and ES, we specially focus our attention on the choice of genetic operators, and the way to apply them. Knowing that it is not possible to reach a universal heuristic able to choose the genetic operators and to manage them, we present methods where the genetic operators themselves are evaluated according to their performance. Thanks to those methods, we observe improvements in order to optimize classical problems.

Keywords *evolutionary algorithms - steady state genetic algorithms - adaptive operators*

INTRODUCTION

Evolutionary computation comprises different techniques that have been inspired by biological mechanisms. Beyond the canonical genetic algorithm (GA) [Holland 75], [Goldberg 82, 89], many ways of research intend to accelerate evolution in view of optimization problems with a space of candidate solutions. GA are inspired by the natural evolution rules where a large population is updated, by selection and crossing-over, where mutations with very weak probabilities may bring a modification for the individual behaviour in any direction, and then, may facilitate or not the reproduction ability. This work focuses in investigation how a family of various genetic operators could be applied to master and to improve the technique of steady-state evolutionary algorithms. We start with a representation able to encode a wide field of problems, and a description of various operators, and next, we study different methods to which those operators can be applied and also different methods for population updating.

I ALGORITHMS WITH AN OPERATOR FAMILY

For the following comparisons, we always use the same representation and a set of operators defined with respect to this representation.

I-1 Representation

In any case, P_0 is a random population of μ individuals, the chromosome encoding is defined according to the problem, it could be character string, symbols list, tree as in genetic programming, any structure mixing numeric and symbolic items. Here, we encode all spaces like $[a, b]^{dim}$, with solutions represented by a vector of decimal digits list in order to code real numbers in $[0, 1[^{dim}$. For instance if dim = 1, among [7, 10], the integer list [3;3;3;3;3;3;3;3;3;3;3;3] means 1/3, and it will represent 8 by dilatation to the interval [7, 10].

I-2 Operators

Many particular works are looking for specific operators linked to a special representation. Following [Michalewicz 92], a lot of papers introduce different kinds of specific operators, and some are devoted to adaptive probabilities [Davis 91], [Tuson, Ross 96]. So, let OP_0 be a random

list picked (with eventually repetitions) among a list of genetic operators : mutations, transpositions for two genes, inversion of sequences, gaussian noise for numeric genes, migration, creation or suppression inside the chromosome, crossover over one or two sites, ... particular removing of a subtree by an other... All operators may be imagined according to the problem. The following operators are used to explore the neighbourhood (the "migation-0" is not only used to initialize the population, but also to replace double or similar individuals and moreover as an operator) :
migration-0 : gives a completely new individual (Monte-Carlo method)
migration-1 : everything is replaced except the only first element of each component
migration-2 : a random half of components is replaced
migration-3 : a component is completely removed and replaced by a random one
mutation-1 : mutation for a randomly chosen digit in a component
mutation-2 : "small" mutation adding ±1 to one digit.
addition : a new digit is added at the end of one component
transposition-1 : two digits are swapped inside a same component
transposition-2 : two components are swapped (for a dimension greater than 2)
copy : a component is removed by an other duplicated component (for dim > 2)
crossover-0 : uniform crossover with a random other parent [Michalewicz 1992]
crossover-1 : one site crossover inside the list of components
crossover-2 : two sites crossover
crossover-3 : crossover only with one the 10% best individuals ("coral-fish" idea inspired by the breeder genetic algorithm [Schierkamp-Vosen 1993]
symmetry : the individual x produces a child x' where x+x' = 2p where p is another parent randomly chosen (computing is here on digits modulo 10)

differential operator : inspired by [Storn, Price 96, 97], [Lampinen 99], x may produce x' by a kind of "tri-crossover", where two other parents y, z are randomly chosen with x, y, z all distinct, then $x' = x + \chi(y - z)$. With the parameter $\chi = 0.5$.

I-3 Methods to apply operators

With this family OP_0, many ways can be tested, after a first step of trials, the most interesting are :

Mutation + crossover We tried this way with various probabilities as (1/2, 1/2) or (1/3, 2/3) or (1/10, 9/10) with similar results.

Unary operators All operators except crossovers are randomly picked. It is possible to try methods like "only mutations", "only migrations", "only crossovers" ...

Ranking method All operators are simply used in the rank they are defined, for instance if there are three of them and two individuals, the first time the third operator is applied is at the second generation to the first individual.

Random method For each individual, the operator which is used, is randomly picked, into the family OP_0 with uniform repartition.

Sort We provide a really moving set of various genetic operators, each one will have a measure (0 at the begining and cumulated along the evolution). The changes over the course of an evolution, are performed according to their ability to decrease the fitness. For those three methods, each generation t, the sorted operators of OP_t are applied in the same rank to the individuals of the sorted population P_t. In order to minimize, the fitness, when one of the operators, op, is applied to an individual i, the op "score" is increased by f(op(i)) - f(i). The list OP_{t+1} will be defined as sorting OP_t, and thus, the best of them (low score) will be applied to the best of P_t (low fitness) and so on.

Sort and roulette wheel Operators family OP_t is also sorted after each generation, but they are picked for the next generation with a "roulette wheel" to promote the best ones. A simple way to make a simulation of this, is taking the $U^2.|OP|$-th operator, where U is a random float in [0, 1].

Sort and duplication [Gacôgne 1997] If there is at least one progress during a generation, the best operator of OP_t is duplicated at the head and the last operator of OP_t is removed. By this way OP_{t+1} will change all the time during evolution.
Otherwise, when any amelioration is observed, the whole family of operators OP_{t+1} is reinitialized. Thus, we avoid a convergence towards the same operator, which could have a role during a phase of evolution, but would exclude the others for the following evolution.
Competing operators [Tvrdik 2002] Each operator has a changing probability starting from $p_i = 1/n_{op}$, if there are n_{op} operators. The success of the i-th operator, producing an individual, can be measured by its weight $w_i = (\mu - r)/\mu$ in [0, 1] where r is the position ($0 \le i < \mu$) of this individual in the new population. Then, if W_i is the cumul of those ranks, the probability p_i of the i-th operator can be updated by $p_i = Wi / (\sum_{1 \le j \le nop} W_j)$. At generation t, operators of OP_t are used according to their probabilities.

I-4 Replacement policy

The standard genetic algorithm In standard GA, solutions are described in a binary encoding way, parents are picked according to the biased wheel principle and they are always replaced by their children. Note that GA is not elitist in the way that the best individual can be removed.
The Strategy of evolution ES(μ+λ) In evolution strategies (ES) [Schwefel 90], each parent may produce a number of children (λ=7 or around 7 being experimentaly the best choice) [Bäck 95, 97]. ES are a kind of accelerate evolution with the alternative to mix parents and children to keep the μ best in ES($\mu + \lambda$), or to remove old generation in the not elitist - ES(μ, λ) by the new one.
The steady-state algorithm SSGA In the original version, [Whitley, Kauth 88], [De Jong, Sarma 93], two parents are chosen according to the roulette wheel, crossover and mutations are applied and the two children replace the two worst individuals of the population. For all the following work, we shall call SSGA(μ, τ) the algorithm where each one of the μ parents produces a child (with an operator among the predefined genetic operators family OP) in such a way that μ parents have exactly an offspring of μ children, then, with a rate τ (for example 33%) the τ best children remove the τ worst parents.
Evolutionary Strategy with Adaptive Operators, elitist or steady-state-ESAO
Since 1993, we used this algorithm for the tuning of fuzzy controler [Gacôgne 97]. The i-th parent produces in both cases the i-th child thanks to the i-th operator of OP. ESAO is the "sort and duplication" way described above, and in "elitist-ESAO" the worst between father and child, at once, is removed (elitist Lamarck' evolution). In "steady-state" ESAO(μ, τ), the worst part of old generation is updated by the best part of the new one as in SSGA(μ, τ).

I-5 Heterogeneous policy

Ideas about partitions in sub-populations or nitching are well-known. Here, we use a very simple fuzzy extension of equality adapted to the decimal representation in view of making a drastic elimination of similar individuals (similar genotypes, not similar phenotypes). This elimination (clearing) procedure is added in removing the worst of each pair of individuals scrolling the population from the worst to the best after each generation. This elimination of neighboring chromosomes is performed only if some similarity degree is under a threshold, and then, a new random individual can be introduced by migration to remove too similar chromosomes. Of course when this procedure is realized, it must be again followed by a population sorting.

The rate of similarity between two vectors in $[0, 1]^n$ will be the minimum of the rates of common digits seen on the shortest length for each pair of components. For instance, with a function called *prox*, we could have *prox* ([1;2;3], [1;2;3;4;5;6]) = 100% and *prox* ([1;2;3;4;5;6;7;8;9;0], [1;2;3;4;8;5;6;7;8;9;0]) = 40%. We always, use the 33% threshold.

II RESULTS

II-1 Previous results

The criteria of comparison being the averaging number of evaluations of the fitness to reach a solution, for example a threshold 10^{-4}, figure 1 shows different algorithms experiments that have been carried out [Gacogne 2000].

GA is performed with population sizes from 50 to 100 and probabilities $p_c = 0.5$ and $p_m = 0.1$. We must add that other parameters give similar results.

ES 7 is the strategy ES($\mu + \lambda$) where $\mu = 10$ and each parent produces 7 children. This is very improved when a family of operators is randomly used instead of only mutation and crossover ("op" versus "mut+cross").

ES 7 ¬elitist means ES(μ=10, λ=7)

SSGA is the one with $\mu = 9$ and $\tau = 33\%$.

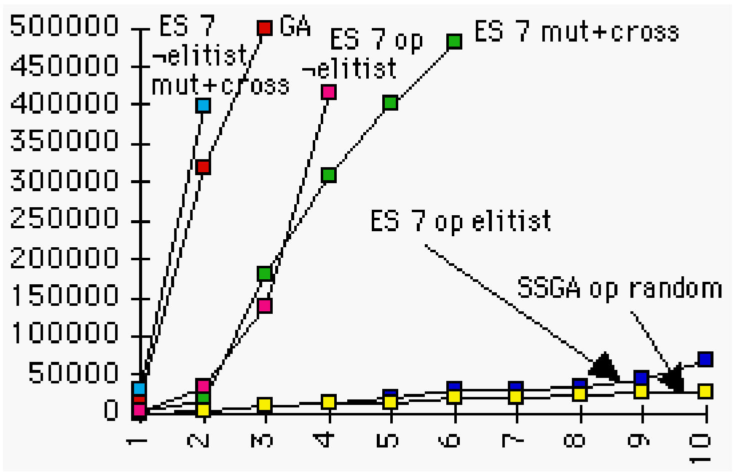

Fig 1 Averaging on 20 runs of the number of evaluations to reach 10^{-4} in the domain $[-500, 500]^{dim}$ for the Griewank function where dimension satisfy $1 \leq dim \leq 10$.

$F_G(x) = (1/4000)\sum_{1 \leq i \leq dim} x_i^2 - \prod_{1 \leq i \leq dim} \cos(x_i/\sqrt{i}) + 1$.

We think it is possible to explain those performances on two points : strategies where the best fitness is not decreasing along the generations, such as standard GA or non elitist ES, loose very often their best individual. Strategies as ES($\mu + \lambda$) or elitist-ESAO where the average fitness is decreasing are too much elitist, they show a too homogeneous population. The aim of evolutionary algorithms in view of any optimization problem is not getting a good population, but a good solution, and the best current solution is helped by the more or less good other solutions.

Strategies as SSGA where the averaging fitness is not decreasing, may give a sign of heterogeneousness.

II-2 Few individuals and many operators

This could be the conclusion of previous works, where we showed an improvement with the steady-state replacement policy on very small populations [Gacogne 2002].

Very few results found in the litterature, are performed with a small population [Coello, Pulido 2000].

Figure 2 shows a piece of the representation for the averaging (on 20 runs) number of Griewank function evaluations (*1000) according to small μ from 2 to 15 with a replacement part τ between 1 and μ-1. For greater population sizes μ, the "valley" for the minimum is increasing.

It is surprising that for other problems like, for instance, the Gauss queens problem, the shape of this surface is always the same but with minima around 7 or 9 and a relative $33\% < \tau < 66\%$.

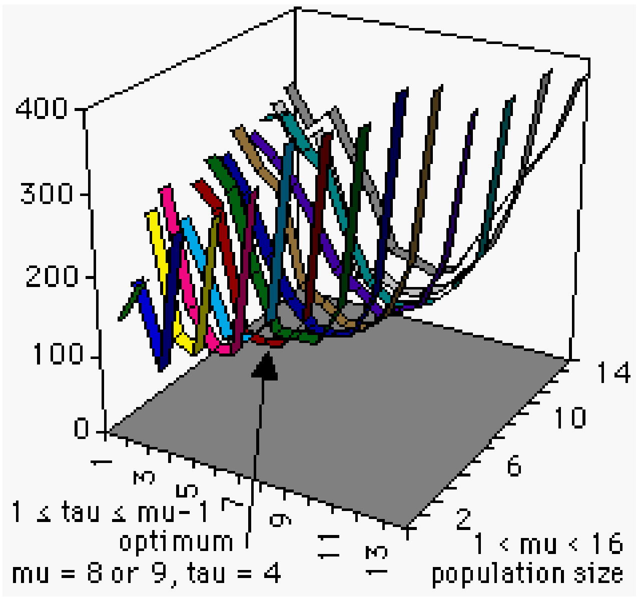

Fig 2 Optimization of the Griewank function in dimension 30 with SSGA(μ, t) for $2 \leq \mu \leq 15$

II-3 Methods to apply the operators

In order to minimize functions having 0 as minimum value, we show the comparison between methods of application of the operators.

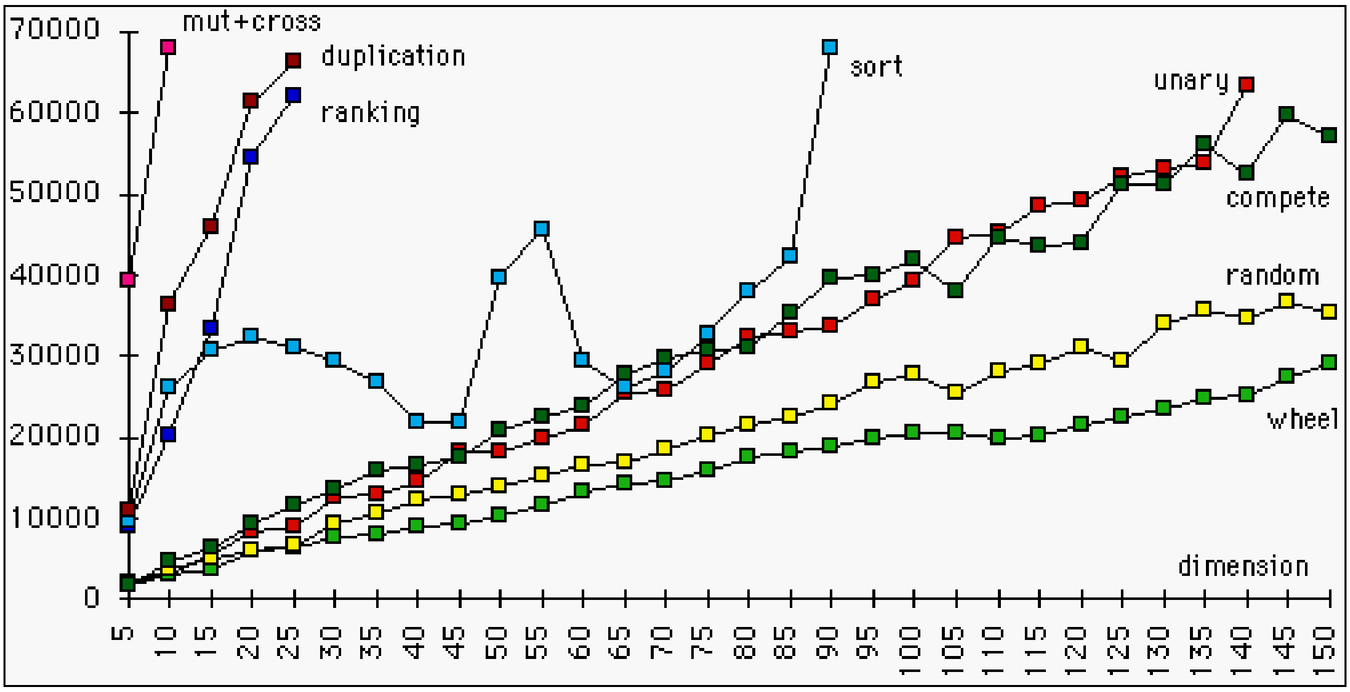

Fig 3 Optimization of the Rastrigin function

$F_R(x) = 0.01[\sum_{1 \le i \le \dim} x_i^2 + 10 - 10\cos(2\pi x_i)]$ for $|x_i| < 500$ with SSGA(9, 3)

The functions like the Rastrigin or Griewank ones, provide similar comportments for the evolutionary methods used.

For a convex function like the DeJong one $F_J(x) = \sum |\mathrm{round}(x_i)|$ for $x = (x_1, x_2, ..., x_{dim})$ $[-500, 500]^{dim}$, results are roughly similar, whereas the sorting or duplication methods for the operators are quite better.

Testing the same 7 methods and also the "mutation + crossover", on different problems, the results are not always exactly the same, but lightly different. The well known Gauss queens problem on a chessboard from 5*5 to 30*30 is quite chaotic, but "random" and "unary" methods are the best methods.

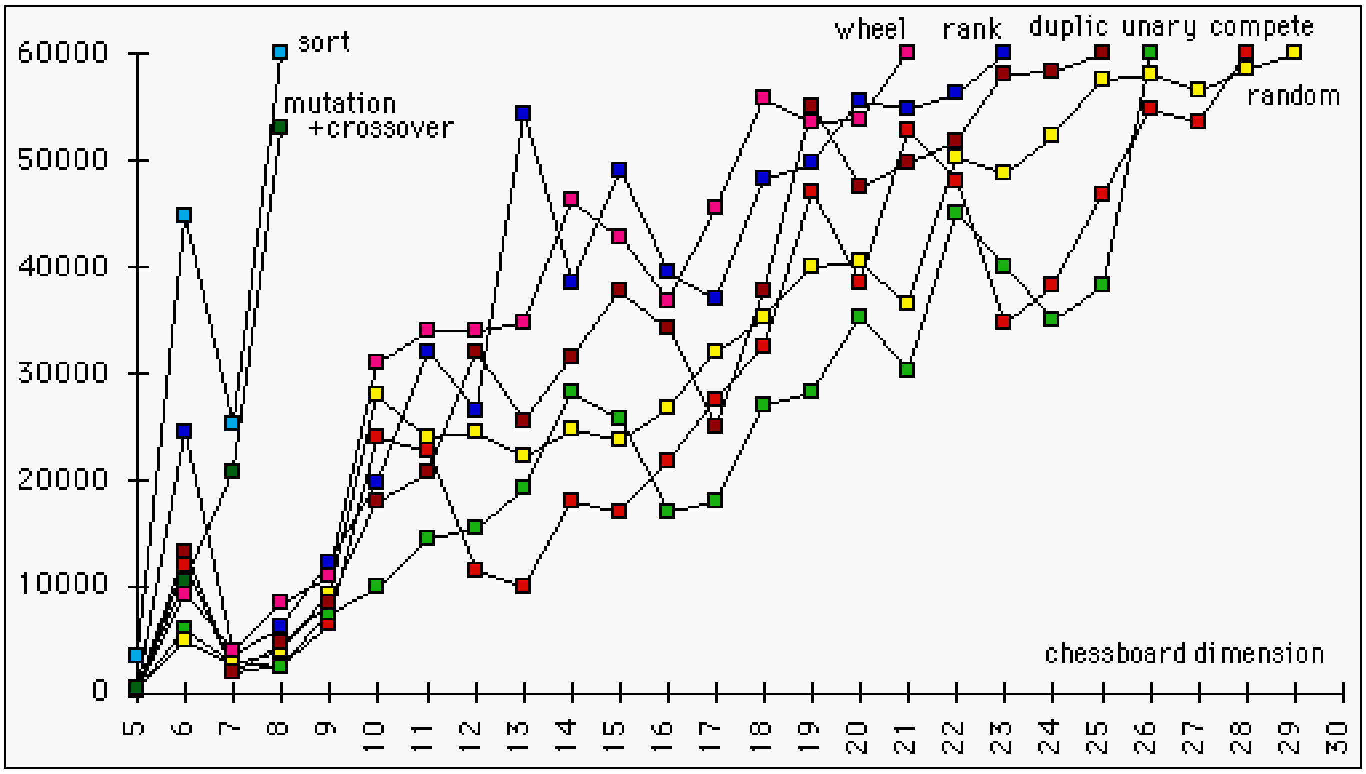

Fig 4 Gauss queens problem with SSGA(8, 4), averaging on 50 runs

We made experiments on the royal road problem [Mitchell, Forest 92] according to a string length from 8 to 128 and the travelling salesman problem with 5 to 50 cities.

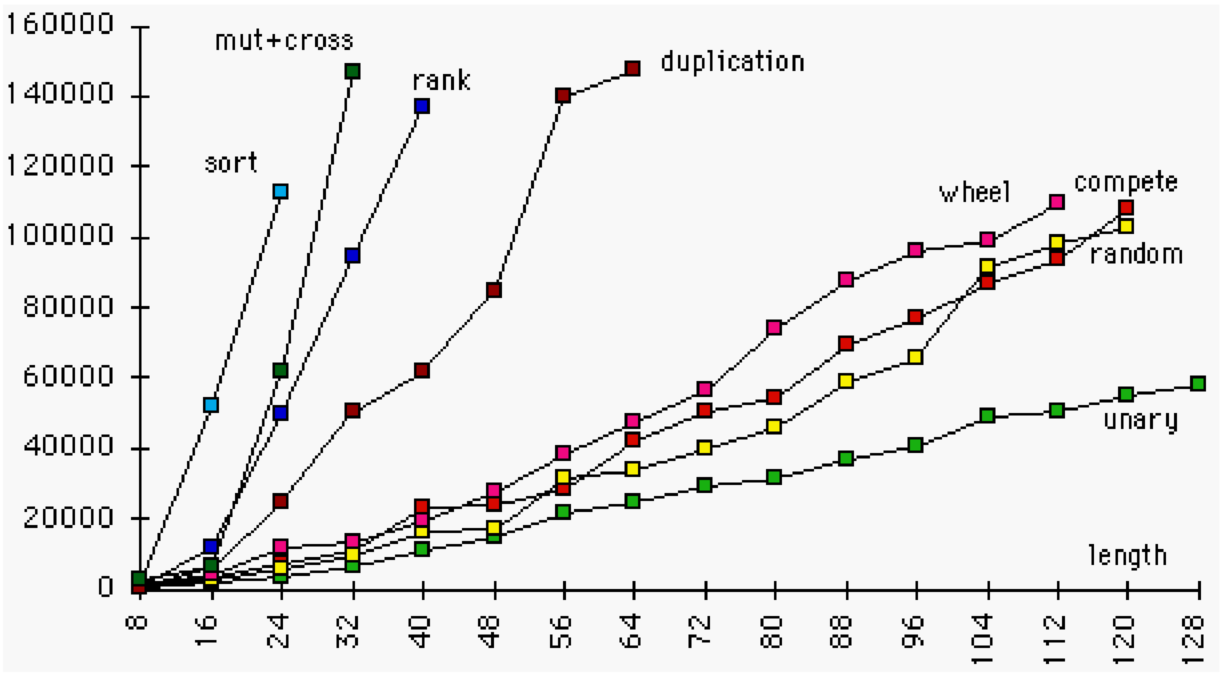

Fig 5 Royal road problem with SSGA(8, 4), averaging on 50 runs

Remark The interest of our representation is a versatility and a possibility to include constraints. For instance, for an x in $[0,1]^n$, the queens problem is described by the fact that the row in the i-th column is Floor($n.x_i$) if n is the chessboard dimension. For the royal road problem with n a 8-multiple, x $[0,1]^n$, could be in {0,1} with Round.

For the travelling salesman problem, also with x [0,1[^{n}, the first city is Floor(n.x$_1$), the i-th city is the Floor((n-i+1)x$_i$)-th element of [1..n] - {v$_1$ ... v$_{i-1}$}. With an optimal known road, it is the search of a particularn permutation among n! permutations of n cities.

We can say, for those last problems, that "rank" or "sort" methods are bad but "duplication" is better, "wheel", "compete" or "random" are roughly similar and the "unary" method is quite better. For those trials we can explain easily that crossovers are not benefic as they destroy good schemas, that is why the "unary" operators method is often a little better.

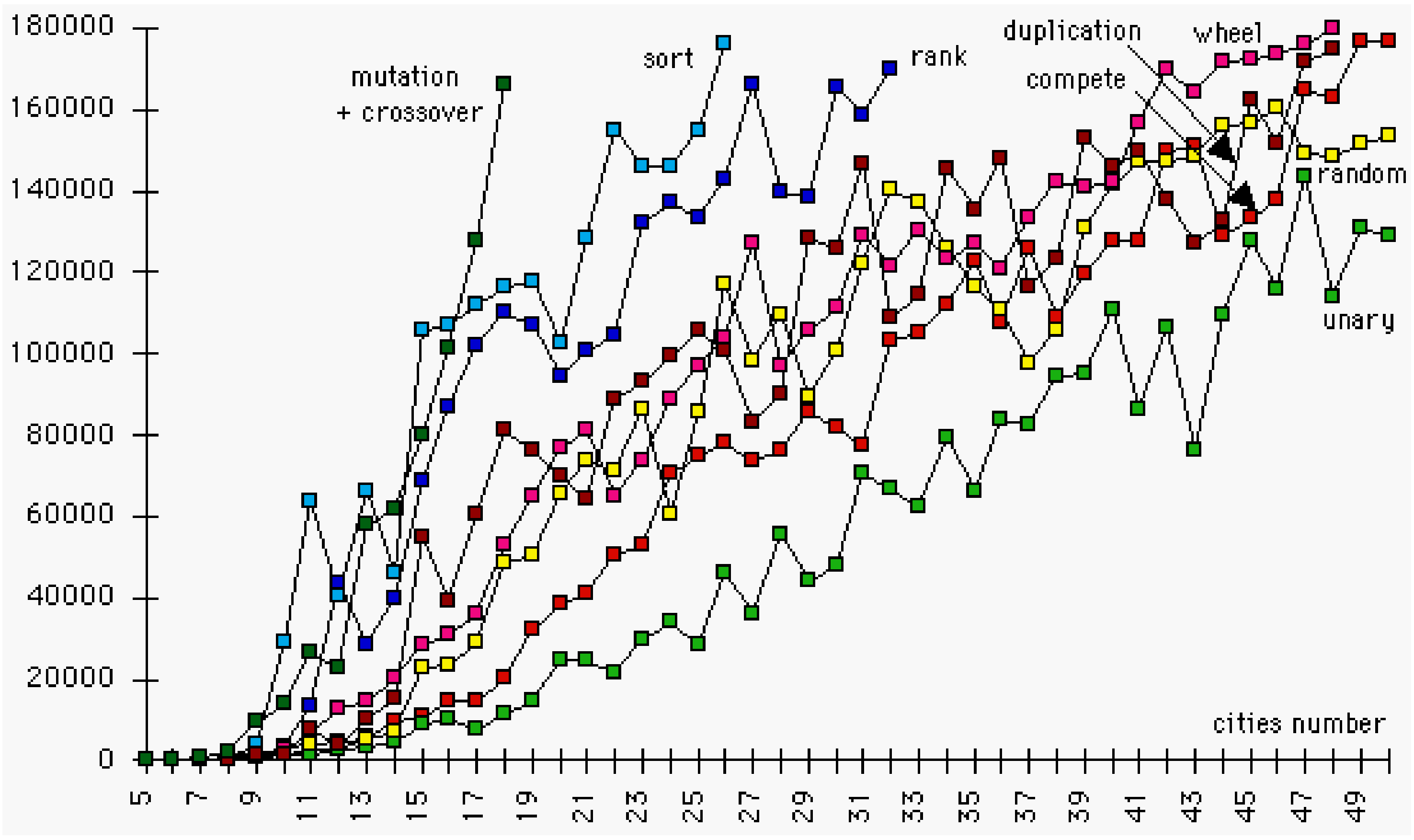

Fig 6 Travelling salesman problem for 5 to 50 cities SSGA(12, 4)

CONCLUSION

As in other fields, models are first taken from nature to try to give a faithful but simple simulation, then nature is kept away in a second time. Different heuristics imagined around the paradigm of artificial evolution are more or less adapted to different problems. We assume that we have no means to choose the best representation, but when we have one, genetic operators can be defined in an intuitive way according to this representation.
The first conclusion of our study, for our representation and its operators, is an improvement of ES, even with the best empirical number of children 7, when a lot of various operators are randomly applied instead of only mutation and crossover.
A second conclusion is that a small size under 10 individuals provides better results, the best updating rate seems always being around 33% as also the best clearing rate.
Our third conclusion is it appears that neither of the methods is preferable for all situations and it will be difficult to find a better way than applying randomly operators.

References

Bäck T. *Evolutionary Algorithms in theory and practice,* Oxford University Press, 1995

Bäck T. Fogel D.B. Schwefel H.P. *Handbook of evolutionary computation,* Oxford University Press, 1997

Coello C., Pulido G. A micro genetic algortihm for multiobjective optimization, Intrenal report Laboratorio National de Informatica Avanzada 2000

Davis L. *Adaptating operator probabilities in genetic algorithm,* Proc. 5th Int. Conf. on GA p61-69, Morgan K. 1993

De Jong K. Sarma J., *Generation Gaps Revisited*, in Foundations of G.A. Morgan Kaufmann, p5-17, 1993

Gacôgne L. *Research of Pareto set by genetic algorithm, application to multicriteria optimization of fuzzy controler*, EUFIT p.837-845, Aachen, 1997

Gacôgne L. *Benefit of a steady state genetic algorithm with adaptive operators,* Mendel Brno p236-242, 2000

Gacôgne L. *Steady-state evolutionary algorithm with an operators family,* EISCI Kosice p173-182, IOS Press 2002

Goldberg D.E. *Genetic algorithms in search, optimization and machine learning,* Addison Wesley, 1989

Holland J.H *Adaptation in natural and artificial system.* Ann Arbor University of Michigan Press, 1975

Lampinen J. *Differential evolution - New naturally parallel approach for engineering design optimization, in Developments in computational mechanics with high performance computing.* Civil Computing Press Edimburgh p217-228, 1999

Michalewicz Z. *Genetic algorithms + data structures= evolution programs*, Springer Verlag 1992

Mitchell M. Forrest S. Holland J.H. *The royal road for genetic algorithm : fitness landscapes and GA performances,* Varela F.J. Bourgine P. Ed. 1992

Schierkamp-Vosen D., Mühlenbein H. *Predictive models for breeder genetic algorithm,* Evol. Comp. 1 p25-49, 1993

Schwefel H.P. *Systems analysis, systems design and evolutionary strategies,* System analysis, Modeling and Simulation vol.7 p.853-864, 1990

Schwefel H.P. *Evolution and optimum seeking*. Sixth generation computer technology series. Wiley, 1995

Storn P. Price K. *Differential evolution a simple and efficient adaptative scheme for global optimization over continuous spaces,* Global Optimization vol 11 p341-359, 1997 (www.icsi.berkeley.edu/~storn)

Storn R. *On the usage of differential evolution for function optimization,* NAFIPS p519-523, 1996

Tuson A. *Adaptating operator rate probabilities in genetic algorithms* MSC Department of Artificial Intelligence, University of Edinburgh 1995

Tuson A. Ross P. *Cost based operator rate adaptation, an investigation* LCNS 1141 p461-469, PPSN 1996

Tvrdik J. Misik L. Krivy I. *Competing heuristics in evolutionary algorithms,* Intelligent Technology, Theory and Applications IOSpress vol 76 p159-165, 2002

Whitley D. Kauth J. *Genitor : a different genetic algorithm,* Proc. of Rocky Mountain Conf. on A.I. p118, 1988