GENOCOP: First Experiments and Results

Zbigniew Michalewicz and Swarnalatha Swaminathan

Department of Computer Science,
University of North Carolina, Charlotte, NC, USA
zbyszek@mosaic.uncc.edu

Abstract

In this paper we discuss the first experiments of the genetic-based optimization tool, Genocop. The test-cases presented in the paper may serve as a benchmark for testing other GA-based systems on constrained optimization problems.

1 Introduction

During the previous conference on genetic algorithms (ICGA '91), an idea of a GA-based optimization tool, Genocop, capable for optimizing linearly constrained problems, was discussed [14]. Genocop borrows many ideas from several other randomized methods. The structure of the Genocop is based on genetic algorithm [11]; however, the system uses floating point representation for chromosomes (as evolutionary strategies [2] do). One of the operators (arithmetical crossover) has its origin in scatter search [7], whereas the ideas of simulated annealing [1] (smaller changes with lower temperature) were incorporated in other operator: non-uniform mutation.

The Genocop can be applied to problems from class $\mathcal{P}$, which is defined as follows:

optimize $f(\overline{X})$, $\overline{X} = (x_1, \ldots, x_q) \in R^q$,

subject to the following sets of linear constraints:

1. Domain constraints: $l_i \leq x_i \leq u_i$ for $i = 1, 2, \ldots, q$. We write $\overline{L} \leq \overline{X} \leq \overline{U}$, where $\overline{L} = \langle l_1, \ldots, l_q \rangle$, $\overline{U} = \langle u_1, \ldots, u_q \rangle$.
2. Equalities: $A\overline{X} = \overline{B}$, where $A = (a_{ij})$, $\overline{B} = \langle b_1, \ldots, b_p \rangle$, $1 \leq i \leq p$, and $1 \leq j \leq q$ (p is the number of equations).
3. Inequalities: $N\overline{X} \leq \overline{D}$, where $N = (n_{ij})$, $\overline{D} = \langle d_1, \ldots, d_{m-p} \rangle$, $1 \leq i \leq m - p$, and $1 \leq j \leq q$ ($m - p$ is the number of inequalities).

This formulation is general enough to handle a large class of standard Operations Research optimization problems with linear constraints and any objective function.

Traditionally, genetic algorithms use penalty function approach to handle constraints. The original problem $\mathcal{P}$ is transformed into:

optimize $f(x_1, x_2, \ldots, x_q) + \epsilon \cdot \delta \sum_{i=1}^{m} W_i$

where m is the number of all constraints, δ is a penalty coefficient, ϵ is -1 for maximalization and $+1$ minimalization problems, and W_i is a penalty related to the i-th constraint ($i = 1, \ldots, m$) [9].

The Genocop system provides a way of handling constraints that is both general and problem independent. The Genocop does not use the concept of penalty functions nor it uses a technique of eliminating offspring generated outside the feasible space (e.g., evolutionary strategies). The main idea behind Genocop approach lies in (1) an elimination of the equalities present in the set of constraints, and (2) careful design of special operators, which guarantee to keep all offspring within the constrained solution space. This can be done very efficiently for linear constraints.

The first version of the Genocop was implemented (by the second author) recently. The system was tested on several linearly constrained problems. The paper presents and discusses the first experiments with the Genocop system and provides some directions for future work.

The paper is organized as follows. The next section summarizes the basic ideas of the Genocop system. Section 3 discusses several test cases and presents their experimental results. The last section contains conclusions and some directions for future work.

2 The Genocop System

During the first stage of the procedure Genocop, all equalities are eliminated. This step is described in detail in [15]. We assume there are p independent equality equations (there are easy methods to verify this), i.e., there are p variables $x_{i_1}, x_{i_2}, \ldots, x_{i_p}$ ($\{i_1, \ldots, i_p\} \subseteq \{1, 2, \ldots, q\}$) which can be determined in terms of the other variables. These can, therefore, be eliminated from the problem statement, using traditional linear algebra methods.

Then the Genocop system (version 1.0) tries to locate an initial (feasible) solution by sampling the feasible region. If some predefined number of trials is unsuccessful, the system would prompt the user for a feasible initial point. The initial population consists of identical copies of such initial point (whether generated or provided by the user).

In the further versions of the Genocop system the initialization process will be upgraded to include a few options (like the number of different initial points provided by the user, definition of sampling parts of the feasible space, etc.).

The evaluation process is straightforward. The objective function f is given and each potential solution $\overline{X} = (x_1, \ldots, x_q)$ of the population yields a value of $f(x_1, \ldots, x_q)$.

The potential solutions of the population $P(t)$ at each iteration t are classified into three sets:

- R – solutions to reproduce ($\text{card}(R) = k$, a parameter of the method).
- D – solutions to die. Again, $\text{card}(D) = k$, however, the sets R and D need not be disjoint.
- N – the remaining (neutral) solutions (neither in R nor in D).

These selections are performed with respect to the relative fitness of the solutions: a solution with a better than average performance has a higher chance to be selected for reproduction; solutions with a worse than average performance have higher chances to be selected to die.

The number of neutral strings in a generation (at least $pop\_size - 2k$ and at most $pop\_size - k - 1$) depends on the number of selected distinct parents and on the number of overlapping strings in sets R and D. Then a new population $P(t+1)$ is formed (update step of the algorithm), consisting of the $pop\_size - k$ strings (all strings except these selected to die) and k offspring of the k parents. Note also that the best solution is always protected and can not be selected to die (this is called the elitist model in GA community).

There are several operators in the Genocop system which proved to be useful on many test problems. These are:

- **uniform mutation:** for this mutation we select a random gene k (from the set of genes of the given chromosome $\overline{X}$. If $\overline{X} = (x_1, \ldots, x_q)$ is a chromosome and the k-th component is the selected gene, the result is a vector $\overline{X}' = (x_1, \ldots, x'_k, \ldots, x_q)$, where x'_k is a random value (uniform probability distribution) from the range $[left(k), right(k)]$. The values $left(k)$ and $right(k)$ are easily calculated from the set of constraints (inequalities); they depend on the remaining values of the chromosome: $x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_q$.

- **boundary mutation:** is a variation of the uniform mutation with x'_k being either $left(k)$ or $right(k)$, each with equal probability.

- **non-uniform mutation:** is one of the operators responsible for the fine tuning capabilities of the system. It is defined as follows: if $\overline{X} = (x_1, \ldots, x_q)$ is a chromosome and the element x_k was selected for this mutation, the result is a vector $\overline{X}' = \langle x_1, \ldots, x'_k, \ldots, x_q \rangle$, where

$$x'_k = \begin{cases} x_k + \triangle(t, right(k) - x_k) & \text{if a random digit is 0} \\ x_k - \triangle(t, x_k - left(k)) & \text{if a random digit is 1} \end{cases}$$

  The function $\triangle(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\triangle(t, y)$ being close to 0 increases as t increases. This property causes this operator to search the space uniformly initially (when t is small), and very locally at later stages. We have used the following function:

$$\triangle(t, y) = y \cdot \left(1 - r^{(1 - \frac{t}{T})^b}\right),$$

  where r is a random number from $[0..1]$, T is the maximal generation number, and b is a system parameter determining the degree of non–uniformity.

- **simple crossover:** is defined as follows: if $\overline{X}_1 = (x_1, \ldots, x_q)$ and $\overline{X}_2 = (w_1, \ldots, w_q)$ are crossed after the k-th position, the resulting offspring are:
  $\overline{X}'_1 = (x_1, \ldots, x_k, w_{k+1}, \ldots, w_q)$ and $\overline{X}'_2 = (w_1, \ldots, w_k, x_{k+1}, \ldots, x_q)$. Note that the only permissible split points are between individual floating points (using float representation it is impossible to split anywhere else).

  However, such operator may produce nonfeasible offspring. To avoid this problem, we use the property of the convex spaces saying, that there exist $a \in [0, 1]$ such that

$$\overline{X}'_1 = \langle x_1, \ldots, x_k, w_{k+1} \cdot a + x_{k+1} \cdot (1 - a), \ldots, w_q \cdot a + x_q \cdot (1 - a) \rangle$$

  and

$$\overline{X}'_2 = \langle w_1, \ldots, w_k, x_{k+1} \cdot a + w_{k+1} \cdot (1-a), \ldots, x_q \cdot a + w_q \cdot (1-a) \rangle$$

are feasible.

The only question to be answered yet is how to find the largest a to obtain the greatest possible information exchange: due to the real interval, we cannot perform an extensive search. In Genocop we implemented a binary search (to some depth only for efficiency). Then, a takes the largest appropriate value found, or 0 if no value satisfied the constraints. The necessity for such actions is small in general and decreases rapidly over the life of the population.

- **arithmetical crossover:** is defined as a linear combination of two vectors: if $\overline{X}_1$ and $\overline{X}_2$ are to be crossed, the resulting offspring are $\overline{X}'_1 = a \cdot \overline{X}_1 + (1-a) \cdot \overline{X}_2$ and $\overline{X}'_2 = a \cdot \overline{X}_2 + (1-a) \cdot \overline{X}_1$. This operator uses a static system parameter $a \in [0..1]$, as it always guarantees closedness (it will preserve all inequality constraints).

3 Test Cases

In order to evaluate the Genocop method, a set of test problems have been carefully selected to illustrate the performance of the algorithm and to indicate that it has been successful in practice. The eight test cases include quadratic, nonlinear, and discontinuous functions with several linear constraints.

All runs of the system were performed on SUN SPARC station 2. We used the following parameters for all experiments:

> *pop_size* $= 70$, $k = 28$ (number of parents in each generation), $b = 2$ (coefficient for non-uniform mutation), $a = 0.25$ (parameter of arithmetical crossover).

For each test case we run the Genocop ten times. For most problems, the number of generations T was either 500 or 1000 (harder problems required larger number of iterations). All test cases and the results of the Genocop system are reported in the following subsections.

3.1 Test Case #1

The problem (taken from [5]) is

$$\text{minimize } f(\overline{X}, y) = -10.5x_1 - 7.5x_2 - 3.5x_3 - 2.5x_4 - 1.5x_5 - 10y - 0.5\sum_{i=1}^{5} x_i^2,$$

subject to:

$$\begin{array}{ll} 6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 \leq 6.5, & 10x_1 + 10x_3 + y \leq 20, \\ 0 \leq x_i \leq 1, & 0 \leq y. \end{array}$$

The global solution is $(\overline{X}^*, y^*) = (0, 1, 0, 1, 1, 20)$, and $f(\overline{X}^*, y^*) = -213$.

The Genocop system found the optimum in all ten runs; a typical optimum point found was:

(0.000055, 0.999998, 0.000041, 0.999989, 1.000000, 19.999033),

for which the value of the objective function is equal to -212.990997. A single run of 1000 iterations took 29 sec of CPU time.

3.2 Test Case #2

The problem (taken from [10]) is

$$\text{minimize } f(\overline{X}) = \textstyle\sum_{j=1}^{10} x_j (c_j + \ln \frac{x_j}{x_1 + \ldots + x_{10}}),$$

subject to:

$$\begin{array}{ll} x_1 + 2x_2 + 2x_3 + x_6 + x_{10} = 2, & x_4 + 2x_5 + x_6 + x_7 = 1, \\ x_3 + x_7 + x_8 + 2x_9 + x_{10} = 1, & x_i \geq 0.000001, \ (i = 1, \ldots, 10), \end{array}$$

where

$$\begin{array}{l} c_1 = -6.089;\ c_2 = -17.164;\ c_3 = -34.054;\ c_4 = -5.914;\ c_5 = -24.721; \\ c_6 = -14.986;\ c_7 = -24.100;\ c_8 = -10.708;\ c_9 = -26.662;\ c_{10} = -22.179; \end{array}$$

The best known solution reported in [10] is

$$\begin{array}{l} (\overline{X}^*) = (.01773548, .08200180, .8825646, .0007233256, .4907851, \\ \qquad .0004335469, .01727298, .007765639, .01984929, .05269826), \end{array}$$

and $f(\overline{X}^*) = -47.707579$.

The Genocop system (in all ten runs) found points with better value than the one above:

$$\begin{array}{l} (\overline{X}^*) = (.04034785, .15386976, .77497089, .00167479, .48468539, \\ \qquad .00068965, .02826479, .01849179, .03849563, .10128126), \end{array}$$

for which the value of the objective function is equal to -47.760765. A single run of 500 iterations took 11 sec of CPU time.

3.3 Test Case #3

The problem (taken from [5]) is

$$\text{minimize } f(\overline{X}, \overline{Y}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{9} y_i,$$

subject to:

$$\begin{array}{ll}
2x_1 + 2x_2 + y_6 + y_7 \leq 10, & 2x_1 + 2x_3 + y_6 + y_8 \leq 10, \\
2x_2 + 2x_3 + y_7 + y_8 \leq 10, & -8x_1 + y_6 \leq 0, \\
-8x_2 + y_7 \leq 0, & -8x_3 + y_8 \leq 0, \\
-2x_4 - y_1 + y_6 \leq 0, & -2y_2 - y_3 + y_7 \leq 0, \\
-2y_4 - y_5 + y_8 \leq 0, & 0 \leq x_i \leq 1,\ i = 1,2,3,4, \\
0 \leq y_i \leq 1,\ i = 1,2,3,4,5,9, & 0 \leq y_i,\ i = 6,7,8.
\end{array}$$

The global solution is $(\overline{X}^*, \overline{Y}^*) = (1,1,1,1,1,1,1,1,1,3,3,3,1)$, and $f(\overline{X}^*, \overline{Y}^*) = -15$.

The Genocop system found the optimum in all ten runs; a typical optimum point found was:

(1.000000, 1.000000, 1.000000, 1.000000, 0.999995, 1.000000, 0.999999,
1.000000, 1.000000, 2.999984, 2.999995, 2.999995, 0.999999),

for which the value of the objective function is equal to -14.999965. A single run of 1000 iterations took 55 sec of CPU time.

3.4 Test Case #4

The problem (taken from [6]) is

$$\text{maximize } f(\overline{X}) = \frac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3} + \frac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3},$$

subject to:

$$\begin{array}{ll}
x_1 + x_2 - x_3 \leq 1, & -x_1 + x_2 - x_3 \leq -1, \\
12x_1 + 5x_2 + 12x_3 \leq 34.8, & 12x_1 + 12x_2 + 7x_3 \leq 29.1, \\
-6x_1 + x_2 + x_3 \leq -4.1, & 0 \leq x_i,\ i = 1,2,3.
\end{array}$$

The global solution is $\overline{X}^* = (1,0,0)$, and $f(\overline{X}^*) = 2.471428$.

The Genocop system found the optimum in all ten runs; a single run of 500 iterations took 9 sec of CPU time.

3.5 Test Case #5

The problem (taken from [5]) is

$$\text{minimize } f(\overline{X}) = x_1^{0.6} + x_2^{0.6} - 6x_1 - 4x_3 + 3x_4,$$

subject to:

$$\begin{array}{ll} -3x_1 + x_2 - 3x_3 = 0, & x_1 + 2x_3 \leq 4, \\ x_2 + 2x_4 \leq 4, & x_1 \leq 3, \\ x_4 \leq 1, & 0 \leq x_i,\ i = 1, 2, 3, 4. \end{array}$$

The best known global solution is $\overline{X}^* = (\frac{4}{3}, 4, 0, 0)$, and $f(\overline{X}^*) = -4.5142$.

The Genocop system found this point in all ten runs; a single run of 500 iterations took 9 sec of CPU time.

3.6 Test Case #6

The problem (taken from [4]) is

$$\begin{aligned} \text{minimize } f(\overline{X}) = {} & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + \\ & +(1 - x_3)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1), \end{aligned}$$

subject to:

$$-10.0 \leq x_i \leq 10.0,\ i = 1, 2, 3, 4.$$

The global solution is $\overline{X}^* = (1, 1, 1, 1)$, and $f(\overline{X}^*) = 0$.

The Genocop system approached the optimum quite closely in all ten runs; a typical optimum point found were:

(0.953835, 0.909714, 1.043783, 1.089695),

for which the value of the objective function is equal to 0.007296, and

(1.018352, 1.037137, 0.980476, 0.961101),

for which the value of the objective function is equal to 0.001333. However, the number of runs was set at 500000, which took 103 min of CPU time.

3.7 Test Case #7

The problem (taken from [5]) is

$$\text{minimize } f(x, \overline{Y}) = 6.5x - 0.5x^2 - y_1 - 2y_2 - 3y_3 - 2y_4 - y_5,$$

subject to:

$$\begin{array}{l} x + 2y_1 + 8y_2 + y_3 + 3y_4 + 5y_5 \leq 16, \\ -8x - 4y_1 - 2y_2 + 2y_3 + 4y_4 - y_5 \leq -1, \\ 2x + 0.5y_1 + 0.2y_2 - 3y_3 - y_4 - 4y_5 \leq 24, \\ 0.2x + 2y_1 + 0.1y_2 - 4y_3 + 2y_4 + 2y_5 \leq 12, \\ -0.1x - 0.5y_1 + 2y_2 + 5y_3 - 5y_4 + 3y_5 \leq 3, \\ y_3 \leq 1,\ y_4 \leq 1, \text{ and } y_5 \leq 2, \\ x \geq 0,\ y_i \geq 0, \text{ for } 1 \leq i \leq 5. \end{array}$$

The global solution is $(x, \overline{Y}^*) = (0, 6, 0, 1, 1, 0)$, and $f(x, \overline{Y}^*) = -11.005$.

The Genocop system approached the optimum quite closely in all ten runs; a typical optimum point found was:

$$(0.000000\ 5.976089,\ 0.005978,\ 0.999999,\ 1.000000,\ 0.000000),$$

for which the value of the objective function is equal to -10.988042. A single run of 1000 iterations took 29 sec of CPU time.

3.8 Test Case #8

The problem was constructed from three separate problems taken from [10] in the following way:

$$\text{minimize } f(\overline{X}) = \begin{cases} f_1 = x_2 + 10^{-5}(x_2 - x_1)^2 - 1.0 & if\ \ 0 \leq x_1 < 2 \\ f_2 = \frac{1}{27\sqrt{3}}((x_1 - 3)^2 - 9)x_2^3 & if\ \ 2 \leq x_1 < 4 \\ f_3 = \frac{1}{3}(x_1 - 2)^3 + x_2 - \frac{11}{3} & if\ \ 4 \leq x_1 \leq 6 \end{cases}$$

subject to:

$$\begin{array}{l} x_1/\sqrt{3} - x_2 \geq 0, \\ -x_1 - \sqrt{3}x_2 + 6 \geq 0, \\ 0 \leq x_1 \leq 6, \text{ and } x_2 \geq 0. \end{array}$$

The function f has three global solutions:

$$\overline{X}_1^* = (0, 0),\ \overline{X}_2^* = (3, \sqrt{3}), \text{ and } \overline{X}_3^* = (4, 0),$$

in all cases $f(\overline{X}_i^*) = -1$ $(i = 1, 2, 3)$.

We made three separate experiments. In experiment k $(k = 1, 2, 3)$ all functions f_i except f_k were increased by 0.5. As a result, the global solution for the first experiment was $\overline{X}_1^* = (0, 0)$, the global solution for the second experiment was $\overline{X}_2^* = (3, \sqrt{3})$, and the global solution for the third experiment was $\overline{X}_3^* = (4, 0)$.

The Genocop found global optimas in all runs in all three cases; a single run of 500 iterations took 9 sec of CPU time.

4 Conclusions

The results of the experiments described in the previous section indicate a few notable features of the Genocop method. First of all, the method converged to the global optimum on a set of different functions (quadratic, nonlinear, discontinuous) and in one case (test #2) the optimum found was better than the best known. This GA-based method deals with linear constraints in a direct way; there is no need to tune penalty parameters for each separate problem. The method has a huge potential: currently the system is extended to handle integer and Boolean variables, are fixed frequencies for all reproduction operators are being replaced by the adaptive ones. Additionally Genocop would constitue the main part of Genocop II, which should handle *any* set of nonlinear constraints (equations and inequalities). The first tests of Genocop II gave excellent results: the system found the optimum in all six test cases (the Genocop II and the experimental results are described in a separate paper [16]).

References

[1] Aarts, E.H.L. and Korst, J., *Simulated Annealing and Boltzmann Machines*, Wiley, Chichester, UK, 1989.

[2] Bäck, T., Hoffmeister, F., and Schwefel, H.-P., *A Survey of Evolution Strategies*, in [3], pp.2–9.

[3] Belew, R. and Booker, L. (Editors), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1991.

[4] Colville, A.R., *A Comparative Study on Nonlinear Programming Codes*, IBM Scientific Center Report 320-2949, New York, 1968.

[5] Floudas, C.A. and Pardalos, P.M., *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer-Verlag, Lecture Notes in Computer Science, Vol.455, 1987.

[6] Floudas, C.A. and Pardalos, P.M., *Recent Advances in Global Optimization*, Princeton Series in Computer Science, Princeton University Press, Princeton, NJ, 1992.

[7] Glover, F., *Heuristics for Integer Programming Using Surrogate Constraints*, Decision Sciences, Vol.8, No.1, pp.156–166, 1977.

[8] Glover, F., *Tabu Search for Nonlinear and Parametric Optimization*, Technical Report, Graduate School of Business, University of Colorado at Boulder; preliminary version presented at the EPFL Seminar on OR and AI Search Methods for Optimization Problems, Lausanne, Switzerland, November 1990.

[9] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, MA, 1989.

[10] Hock, W. and Schittkowski K., *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol.187, Springer-Verlag, 1981.

[11] Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[12] McCormick, G.P., *Computability of Global Solutions to Factorable Nonconvex Programs*, Part I: Convex Underestimating Problems, Mathematical Programming, Vol.10, No.2 (1976), pp.147–175.

[13] Michalewicz, Z. and Janikow, C., *Genetic Algorithms for Numerical Optimization*, Statistics and Computing, Vol.1, No.2, pp.75–91, 1991.

[14] Michalewicz, Z. and Janikow, C., *Handling Constraints in Genetic Algorithms*, in [3], pp.151–157.

[15] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, AI Series, New York, 1992.

[16] Michalewicz, Z. and Attia, N.F., *Genetic Algorithm + Simulated Annealing = GENOCOP II: A Tool for Nonlinear Programming*, submitted for publication, 1993.

[17] Schaffer, J., (Editor), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1989.

[18] Schwefel, H.-P., *Numerical Optimization for Computer Models*, Wiley, Chichester, UK, 1981.