Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints

Zbigniew Michalewicz and Girish Nazhiyath
Department of Computer Science
University of North Carolina
Charlotte, NC 28223, USA
e-mail: {*zbyszek,gnazhiya*}*@uncc.edu*

Abstract

During the last two years several methods have been proposed for handling nonlinear constraints by genetic algorithms for numerical optimization problems; most of them were based on penalty functions. However, the performance of these methods is highly problem-dependent; moreover, many methods require additional tunning of several parameters. In this paper we present a new optimization system (Genocop III), which is based on concepts of co-evolution and repair algorithms. We present the results of the system on a few selected test problems and discuss some directions for further research.

I. Introduction

The general nonlinear programming problem (for continuous variables) is to find $\overline{X}$ so as to

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

where $\overline{X} \in \mathcal{F} \subseteq \mathcal{S}$. The set $\mathcal{S} \subseteq R^n$ defines the search space and the set $\mathcal{F} \subseteq \mathcal{S}$ defines a *feasible* search space. Usually, the search space $\mathcal{S}$ is defined as a n-dimensional rectangle in R^n (domains of variables defined by their lower and upper bounds):

$$l(i) \leq x_i \leq u(i), \quad 1 \leq i \leq n,$$

whereas the feasible set $\mathcal{F} \subseteq \mathcal{S}$ is defined by a set of additional $m \geq 0$ constraints:

$$g_j(\overline{X}) \leq 0, \text{ for } j = 1, \ldots, q, \text{ and } h_j(\overline{X}) = 0, \text{ for } j = q+1, \ldots, m.$$

It is also convenient to divide all constraints into four subsets: linear equations LE, linear inequalities LI, nonlinear equations NE, and nonlinear inequalities NI. Of course, $g_j \in LI \cup NI$ and $h_j \in LE \cup NE$. In fact, we need not consider linear equations LE, since we can remove them by expressing values of some variables as linear functions of remaining variables and making appropriate substitutions [12].

Most research on applications of evolutionary computation techniques to nonlinear programming problems has been concerned with complex objective functions with $\mathcal{F} = \mathcal{S}$. Several test functions used by various researchers during the last 20 years considered only domains of n variables; this was the case with five test functions F1–F5 proposed by De Jong [2], as well as with many other test cases proposed since then [3, 4, 19]. Only recently several approaches were reported to handle general nonlinear programming problems.

This paper surveys briefly these methods (next two Sections) and provides a description of a new system, Genocop III, which is based on the concepts of co-evolution and repair algorithms (Section IV). Section V presents the first experimental results of Genocop III and concludes the paper.

II. The original Genocop system

The Genocop (for GEnetic algorithm for Numerical Optimization of COnstrained Problems) system [12] assumes linear constraints only and a feasible starting point (or feasible initial population). A closed set of operators maintains feasibility of solutions. For example, when a particular component x_i of a solution vector $\overline{X}$ is mutated, the system determines its current domain $dom(x_i)$ (which is a function of linear constraints and remaining values of the solution vector $\overline{X}$) and the new value of x_i is taken from this domain (either with flat probability distribution for uniform mutation, or other probability distributions for non-uniform and boundary mutations). In any case the offspring solution vector is always feasible. Similarly, arithmetic crossover, $a\overline{X} + (1-a)\overline{Y}$, of two feasible solution vectors $\overline{X}$ and $\overline{Y}$ yields always

a feasible solution (for $0 \leq a \leq 1$) in convex search spaces (the system assumes linear constraints only which imply convexity of the feasible search space $\mathcal{F}$).

The Genocop (its third version, together with all previous versions, is available from anonymous, ftp.uncc.edu, directory coe/evol, file genocop3.0.tar.Z) gave surprisingly good performance on many test functions [8]; for example the following test case (this is a test function $G1$ with 13 variables from [10]):

$$\text{minimize } G1(\overline{X}, \overline{Y}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{9} y_i,$$

subject to:

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

requires less than 1000 generations to get the global solution $(\overline{X}^*, \overline{Y}^*) = (1,1,1,1,1,1,1,1,1,3,3,3,1)$, where $G1(\overline{X}^*, \overline{Y}^*) = -15$.

The method can be generalized to handle nonlinear constraints provided that the resulting feasible search space $\mathcal{F}$ is convex. But the weakness of the method lies in its inability to deal with nonconvex search spaces (i.e, to deal with nonlinear constraints in general). After surveying other approaches (Section III) for the general nonlinear programming problem, Section IV describes the most recent extension of the Genocop method (Genocop III) to overcome the above restriction.

III. The other approaches

During the last two years several methods were proposed for handling nonlinear constraints by genetic algorithms for numerical optimization problems. Most of them are based on the concept of penalty functions, which penalize unfeasible solutions, i.e.,[1]

$$eval(\overline{X}) = \begin{cases} f(\overline{X}), & if\ \overline{X} \in \mathcal{F} \\ f(\overline{X}) + penalty(\overline{X}), & otherwise, \end{cases}$$

where $penalty(\overline{X})$ is zero, if no violation occurs, and is positive, otherwise. In most methods a set of functions f_j $(1 \leq j \leq m)$ is used to construct the penalty; the function f_j measures the violation of the j-th constraint in the following way:

$$f_j(\overline{X}) = \begin{cases} \max\{0, g_j(\overline{X})\}, & if\ 1 \leq j \leq q \\ |h_j(\overline{X})|, & if\ q+1 \leq j \leq m. \end{cases}$$

However, these methods differ in many important details, how the penalty function is designed and applied to unfeasible solutions; we discuss them briefly in turn; for a full discussion, see [10].

One of the methods was proposed by Homaifar, Lai, and Qi [6]. The method assumes that for every constraint we establish a family of intervals which determine appropriate penalty coefficient. For each constraint it creates several levels (ℓ) of violation, and for each level of violation and for each constraint, it creates a penalty coefficient R_{ij} $(i = 1,2,\ldots,\ell,\ j = 1,2,\ldots,m)$; higher levels of violation require larger values of this coefficient. It starts with a random population of individuals (feasible or unfeasible), and evolves the population using the following formula

$$eval(\overline{X}) = f(\overline{X}) + \sum_{j=1}^{m} R_{ij} f_j^2(\overline{X}).$$

Another method was proposed by Joines and Houck [7]. As opposed to the previous method, the authors assumed dynamic penalties. Individuals are evaluated (at the iteration t) by the following formula:

$$eval(\overline{X}) = f(\overline{X}) + (C \times t)^{\alpha} \sum_{j=1}^{m} f_j^{\beta}(\overline{X}),$$

where C, α and β are constants.

One of the methods was proposed by Schoenauer and Xanthakis [17]. It starts with a random population of individuals (feasible or unfeasible), and tries to minimize violation of the first constraint. Once a given percentage of the population (so-called flip threshold ϕ) is feasible for this constraint,[2] the current population is used as the starting point for the next phase of the evolution, where minimization of the

[1] In the rest of the paper we assume minimization problems.

[2] The method suggests the use of a sharing scheme (to maintain diversity of the population).

next constraint is attempted. During this phase, points that do not satisfy one of the previously considered constraints are eliminated from the population. In the final iteration, the system tries to optimize the objective function rejecting unfeasible individuals.

The next method was described by Michalewicz and Attia [11]; it is incorporates the Genocop system (described in the previous section) and its modified version (Genocop II, see [8] for details) starts by selecting a random single point as a starting point (the initial population consists of copies of this single individual), which satisfies linear constraints (LE and LI). The initial temperature τ is set and individuals are evaluated by the following formula

$$eval(\overline{X}, \tau) = f(\overline{X}) + \frac{1}{2\tau} \sum_{j=1}^{m} f_j^2(\overline{X}),$$

After some number of generations the temperature τ is decreased and the best solution found so far serves as a starting point of the next iteration. The process continues until the temperature reaches the freezing point.

The next method was developed by Powell and Skolnick [15]. The method is a classical penalty method with one notable exception. Each individual is evaluated by the formula:

$$eval(\overline{X}) = f(\overline{X}) + r \sum_{j=1}^{m} f_j(\overline{X}) + \lambda(t, \overline{X}),$$

where r is a constant; however, there is also a component $\lambda(t, \overline{X})$. This is an additional iteration dependent function which influences the evaluations of unfeasible solutions. The point is that the method distinguishes between feasible and unfeasible individuals by adopting an additional heuristic rule (suggested earlier in [16]): for any feasible individual $\overline{X}$ and any unfeasible individual $\overline{Y}$: $eval(\overline{X}) < eval(\overline{Y})$, i.e., any feasible solution is better than any unfeasible one. This can be achieved by adding additional penalty component $\lambda(t, \overline{X})$ to all unfeasible individuals $\overline{X}$; the value of this component is determined by the values of the best unfeasible and the worst feasible individuals in a given iteration t.

The final method rejects unfeasible individuals (death penalty); the method has been used, for example, by evolution strategies [1] and simulated annealing.

IV. Genocop III

This method incorporates the original Genocop system (described in Section II), but also extends it by maintaining two separate populations, where a development in one population influences evaluations of individuals in the other population. The first population consists of so-called search points from $\mathcal{S}$ which satisfy linear constraints of the problem (as in the original Genocop system). The feasibility (in the sense of linear constraints) of these points is maintained, as before, by specialized operators. The second population consists of so-called reference points from $\mathcal{F}$; these points are fully feasible, i.e., they satisfy *all* constraints.[3] Reference points $\overline{R}$, being feasible, are evaluated directly by the objective function (i.e., $eval(\overline{R}) = f(\overline{R})$). On the other hand, unfeasible search points are "repaired" for evaluation and the repair process works as follows. Assume, there is a search point $\overline{S} \notin \mathcal{F}$. In such a case the system selects[4] one of the reference points, say $\overline{R}$, and creates random points $\overline{Z}$ from a segment between $\overline{S}$ and $\overline{R}$ by generating random numbers a from the range $\langle 0, 1 \rangle$: $\overline{Z} = a\overline{S} + (1-a)\overline{R}$. Once a feasible $\overline{Z}$ is found, $eval(\overline{S}) = eval(\overline{Z}) = f(\overline{Z})$.[5] Additionally, if $f(\overline{Z})$ is better than $f(\overline{R})$, then the point $\overline{Z}$ replaces $\overline{R}$ as a new reference point. Also, $\overline{Z}$ replaces $\overline{S}$ with some probability of replacement p_r.

The Genocop III avoids many disadvantages of other systems (see Section III). It introduces few additional parameters (the population size of reference points, probability of replacement) only. It always returns a feasible solution. A feasible search space $\mathcal{F}$ is searched by making references from the search points. The neighborhoods of better reference points are explored more often. Some reference points are moved into the population of search-points, where they undergo transformation by specialized operators (which preserve linear constraints).

V. Experiments, results, and conclusions

Recently a set of five test problems ($G1$–$G5$) was proposed [10] for constrained numerical optimization. All test cases are summarized in Table 1; for each problem Gi we list number n of variables, type of the function f, the ratio $\rho = |\mathcal{F}|/|\mathcal{S}|$, the number of constraints of each category (linear inequalities LI,

[3] If Genocop III has difficulties in locating such a reference point for the purpose of initialization, the user is prompted for it. In cases, where the ratio $|\mathcal{F}|/|\mathcal{S}|$ is very small, it may happen that the initial set of reference points consists of a multiple copies of a single feasible point.

[4] Better reference points have better chances to be selected; a selection method based on nonlinear ranking was used.

[5] Clearly, in different generations the same search point $\mathcal{S}$ can evaluate to different values due to the random nature of the repair process.

nonlinear equations NE and inequalities NI), the number a of active constraints at the optimum, and the optimum value of the objective function. For the full description of these test cases the reader is referred to [10].

Problem	n	Type of f	ρ	LI	NE	NI	a	Optimum
$G1$	13	quadratic	0.0111%	9	0	0	6	-15.000
$G2$	8	linear	0.0010%	3	0	3	6	7049.331
$G3$	7	polynomial	0.5121%	0	0	4	2	680.630
$G4$	5	nonlinear	0.0000%	0	3	0	3	0.054
$G5$	10	quadratic	0.0003%	3	0	5	6	24.306

Table 1: Summary of five test cases. The ratio $\rho = |\mathcal{F} \cap \mathcal{S}|/|\mathcal{S}|$ was determined experimentally by generating 1,000,000 random points from $\mathcal{S}$ and checking whether they belong to $\mathcal{F}$. LI, NE, and NI represent the number of linear inequalities, and nonlinear equations and inequalities, respectively.

The first problem $G1$ was presented in Section II. The results of Genocop III on $G1$ are identical to these of the original Genocop: since there are no nonlinear constraints, there is no need for a population of reference points. Out of remaining four test cases, we experimented with three ($G2$, $G3$, and $G5$); the problem $G4$ contains nonlinear equations NE, and the current version of Genocop III does not handle them yet.

The Genocop III was run for 5,000 iterations (as other systems discussed in [10]). The probabilities of all operators were set at 0.08, and the both population sizes were 70.

The results were very good. For example, for the problem $G2$ the best result was 7286.650: much better than the best result of the best system from these discussed in Section III (for this problem, it was Genocop II with 7377.976). Similar performance was observed on two other problems, $G3$ (with 680.640) and $G5$ (with 25.883). Additional interesting observation was connected with stability of the system. Genocop III had a very low standard deviation of results. For example, for problem $G3$, all results were between 680.640 and 680.889; on the other hand, other system produced variety of results (between 680.642 and 689.660, see [10]).

Of course, all resulting points $\overline{X}$ were feasible, which was not the case with other systems (e.g., Genocop II produced a value of 18.917 for the problem $G5$, the systems based on the methods of Homaifar, Lai, and Qi [6] and Powell and Skolnick [15] gave results of 2282.723 and 2101.367, respectively, for the problem $G2$).

Clearly, Genocop III is a promising tool for constrained nonlinear optimization problems. However, there are many issues which require further attention and experiments. These include investigation of the significance of the ratio of $\rho = |\mathcal{F}|/|\mathcal{S}|$. Note that it is possible to represent some linear constraints as nonlinear constraints; this change in the input file would change the space of reference points making it smaller and the space of linearly feasible search points would be larger. However, it is unclear, how these changes would affect the performance of the system.

Another group of experiments is connected with a single parameter: probability of replacement p_r. Recently [13] a so-called 5%-rule was reported: this heuristic rule states that in many combinatorial optimization problems, an evolutionary computation technique with a repair algorithm provides the best results when 5% of repaired individuals replace their unfeasible originals. It would be interesting to check this rule in numerical domains. Current version of Genocop III used $p_r = 0.20$.

Also, we plan (in a very near future) to extend Genocop III to handle nonlinear equations. This would require an additional parameter (ϵ) to define the precision of the system. All nonlinear equations $h_j(\overline{X}) = 0$ (for $j = q+1, \ldots, m$) would be replaced by a pair of inequalities:

$$-\epsilon \leq h_j(\overline{X}) \leq \epsilon.$$

This new version of Genocop III should handle directly the problem $G4$.

Acknowledgments:

This material is based upon work supported by the National Science Foundation under Grant IRI-9322400. The first author is also at the Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland; e-mail: *zbyszek@ipipan.waw.pl.*

References

[1] Bäck, T., Hoffmeister, F., and Schwefel, H.-P., *A Survey of Evolution Strategies*, Proceedings of the Fourth ICGA, Morgan Kaufmann Publishers, Los Altos, CA, 1991, pp.2–9.

[2] De Jong, K.A., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, (Doctoral dissertation, University of Michigan), *Dissertation Abstract International*, 36(10), 5140B. (University Microfilms No 76-9381).

[3] Eshelman, L.J. and Schaffer, J.D, *Real-Coded Genetic Algorithms and Interval Schemata*, Foundations of Genetic Algorithms – 2, Morgan Kaufmann, Los Altos, CA, 1993, pp. 187–202.

[4] Fogel, D.B. and Stayton, L.C., *On the Effectiveness of Crossover in Simulated Evolutionary Optimization*, BioSystems, Vol.32, 1994, pp.171–182.

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

[6] Homaifar, A., Lai, S. H.-Y., Qi, X., *Constrained Optimization via Genetic Algorithms*, Simulation, Vol.62, No.4, 1994, pp.242–254.

[7] Joines, J.A. and Houck, C.R., *On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems With GAs*, Proceedings of the IEEE ICEC 1994, pp.579–584.

[8] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, 2nd edition, 1994.

[9] Michalewicz, Z., *A Survey of Constraint Handling Techniques in Evolutionary Computation Methods*, Proceedings of the 4th Annual Conference on EP, MIT Press, Cambridge, MA, 1995.

[10] Michalewicz, Z., *Genetic Algorithms, Numerical Optimization, and Constraints*, Proceedings of the Sixth ICGA, Morgan Kaufmann, 1995, pp.151–158.

[11] Michalewicz, Z., and Attia, N., *Evolutionary Optimization of Constrained Problems*, Proceedings of the 3rd Annual Conference on EP, World Scientific, 1994, pp.98–108.

[12] Michalewicz, Z. and Janikow, C., *Handling Constraints in Genetic Algorithms*, Proceedings of the Fourth ICGA, Morgan Kaufmann, 1991, pp.151–157.

[13] Orvosh, D. and Davis, L., *Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints*, Proceedings of the Fifth ICGA, Morgan Kaufmann, 1993, p.650.

[14] Paredis, J., *Co-evolutionary Constraint Satisfaction*, Proceedings of the 3rd PPSN Conference, Springer-Verlag, 1994, pp.46–55.

[15] Powell, D. and Skolnick, M.M., *Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints*, Proceedings of the Fifth ICGA, Morgan Kaufmann, 1993, pp.424–430.

[16] Richardson, J.T., Palmer, M.R., Liepins, G., and Hilliard, M., *Some Guidelines for Genetic Algorithms with Penalty Functions*, in Proceedings of the Third ICGA, Morgan Kaufmann, 1989, pp.191–197.

[17] Schoenauer, M., and Xanthakis, S., *Constrained GA Optimization*, Proceedings of the Fifth ICGA, Morgan Kaufmann, 1993, pp.573–580.

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

[19] Wright, A.H., *Genetic Algorithms for Real Parameter Optimization*, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, Morgan Kaufmann, 1991, pp. 205–218.