Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 199 (2008) 186–194
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

A new genetic algorithm for solving nonconvex nonlinear programming problems

M.B. Aryanezhad [a], Mohammad Hemati [b,*]

[a] Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
[b] Department of Industrial Management, Science and Research Campus, Islamic Azad University, Iran

Abstract

In nonlinear programming problems (especially nonconvex problems), attaining the global optimum is crucial. To reach this purpose, the current paper represents a new genetic algorithm for solving nonconvex nonlinear programming problems. The new method is simpler and more intuitive than the existing models and finds the global optimum of the problem in a reasonable time. The proposed technique, to attain the global optimum of problem (especially in large scale problems) instead of increasing the size of population which usually conduces to the curse of dimensionality – that is widespread in usual genetic algorithms – decomposes the main problem into several sub problems, but with lower size of population in each sub problem. This decomposition has been formed in such a way that it could envelop the total solution space. The proposed genetic algorithm, which determines the sufficient sub problems, could converge to global optimum of problem. To be able to measure the proposed genetic algorithm, several problems have been solved in different spectrums. Herein, it has been shown that the proposed technique, both in run time and in solution quality, is preferred to the usual genetic algorithm in this domain (nonlinear continuous programming problems).

Keywords: Genetic algorithm; Nonconvex programming; Decomposition; Sub problems; Global optimum

1. Introduction

Genetic algorithms (GAs) proposed by Holland [1] have attracted considerable attention as global methods for complex function optimization since De Jong considered GAs in a function optimization setting [2]. Since 1962, many nonlinear problems have been solved by using genetic algorithm methods [3]. GAs are being applied to a wide range of programming problems in many domains, e.g., multiobjective routing problems [4], traveling salesman problems [5,6], production and operation management problems such as production control, scheduling facility layout, line balancing, production planning and supply chain management [7].

* Corresponding author.
E-mail address: mo928hem@yahoo.com (M. Hemati).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2007.09.047

Generally, GAs have eight basic components: genetic representation, initial population, evaluation function, reproduction selection scheme, genetic operators, generational selection scheme, stopping criteria and GAs parameter setting [7].

Furthermore, recent results (the No Free Lunch theorems) on search algorithms suggest that GAs (and other methods) may not be a better method than random search [8].

In the nonlinear problems, optimization field is in contrast to convex programming in which each local minimum is a global minimum; in nonconvex programming, if you are successful in finding a local minimum, there is no assurance that it also will be a global minimum [9]. One of the techniques that has been popular in the last four decades for solving nonconvex problems of optimization is genetic algorithm [8,13]. But the main problem that occurred in genetic algorithm is when the dimension of variables increases, the solution of problem conduces to difficulty which is called "curse of dimensionality" and convergence to global optimum is very difficult [10,11].

In this paper, we want to show that by decomposition of the main problem into sub problems using lower and upper bounds of variables, but lower size of population, the global optimum of problem can be reached in a reasonable time. In addition, we show that the proposed technique is an efficient way of converging to global optimum. Furthermore, this technique is preferred to the usual genetic algorithm in both time and solution quality. The new proposed technique has been explained in the following section.

2. Revision of the proposed genetic algorithm

In this article, we want to propose a new technique that works with high efficiency in nonconvex nonlinear programming problems to attain the global optimum. In new algorithm, the lower and upper bounds of variables move with fixed steps together and with each step a sub problem with new lower and upper bounds has been formed. Furthermore in each step, the new genetic algorithm was run. When difference vectors between the lower and upper bounds of variables were zero, the vector of lower and upper bounds of variables moves backward until it reaches the first positions and it is the same when the sub problems are formed. This process can be repeated several times.

2.1. Steps of new technique

Like any algorithm, the proposed algorithm has three steps: initiation step, repeated step and stop step.

2.1.1. Initiation step

Formulate the nonlinear problem as below:

$$\begin{aligned} & p^{h,i} \quad h = 1,\ldots,10 \quad i = 1,2,\ldots,n \\ \text{Minimize:} \quad & f(X), \\ \text{subject to:} \quad & g_j(x) \leqslant 0, \quad j = 1,2,\ldots,m(1), \\ & h_k(x) = 0, \quad k = 1,2,\ldots,L \\ & l_i \leqslant x \leqslant u_i, \quad i = 1,2,\ldots,n, \\ & x \in S. \end{aligned} \tag{1}$$

If the local optimum from each sub problem was shown by $\tilde{f}^{h,i}$, then for $\bar{f}$ (the global optimum) we have

$$\bar{f} = \min\{\tilde{f}^{h,i}\}. \tag{2}$$

The descriptions of model structures and indexes are

1. h[1]: h is number of runs. In this article $h = 1, 2, \ldots, 10$.

[1] The number of 10 runs is optional, but for assurance of convergences of solution, at least 10 runs are required

2. i[2]: i is the number of sub problems. In this article $i = 1, 2, \ldots, 30$.
3. $f(x)$, g_j, h_k are functions that are defined in Euclidean space (E^n).
4. The vector X is defined in $n(n \geqslant 1)$ dimensions and is continuous.
5. The variable X has lower bound (l_i) and upper bound (u_i). The sub problems are formed by these bounds.
6. The subclass of S is a feasible solution of problem. It is a compact set and subset of E^n.
7. All functions are differentiable.
8. The structure of problem in this paper is minimization, therefore if the problem is maximization, multiply the objective function at "-1".
9. Each of the constraints which are of the form, "$\geqslant$", multiply that constraint at "-1" and convert it at form "$\leqslant$".

2.1.2. The repeated step

1. Design 30 subproblems and 10 runs as below:

$i = 30, \quad h = 10,$

$P^{1,1}$

$$\begin{aligned} &\text{Minimize:} && f(X) \\ &\text{subject to:} && g_j(x) \leqslant 0, \quad j = 1, 2, \ldots, m, \\ &&& h_k(x) = 0, \quad k = 1, 2, \ldots, L, \\ &&& l_1 \leqslant x \leqslant u_1, \\ &&& x \in S. \\ &&& \ldots \\ &&& \ldots \\ &&& \ldots \end{aligned} \tag{3}$$

$P^{1,30}$

$$\begin{aligned} &\text{Minimize:} && f(X) \\ &\text{subject to:} && g_i(x) \leqslant 0, \quad j = 1, 2, \ldots, m, \\ &&& h_j(x) = 0, \quad k = 1, 2, \ldots, L, \\ &&& l_{30} \leqslant x \leqslant u_{30}, \\ &&& x \in S. \end{aligned}$$

Therefore, for 10 runs and 30 sub problems in each run, design as follows:

$$p^{1,1}, \ldots, p^{1,30}, p^{2,1}, \ldots p^{2,30}, \ldots, p^{10,1}, \ldots, p^{10,30}. \tag{4}$$

2. Design the below vectors as the following:

L_i Lower bound vector of the variables.
U_i Upper bound vector of the variables.

$$d = U_i - L_i, \tag{5}$$

$$d_s = \frac{d}{2}, \tag{6}$$

$$d_m = \frac{i}{2}, \tag{7}$$

i is the number of sub problems.
The reason of d, d_s and d_m is for preventing the infeasible solution.

[2] Similar to h, the number of 30 sub problems is optional, but for assurance that sub problems envelop the total solution space, it is bare bones.

3. Design the length of step for moving the variables. If d_l was the length of step for lower bound and d_u was the length of step for upper bound of variables, then we have

$$d_l = \frac{d_s}{d_m}, \tag{8}$$

$$d_u = \frac{d_s}{d_m}. \tag{9}$$

4. The lower bound vector of variables move to the upper bound of variables as below (with each step one sub problem is formed):

$$L_i = L_i + d_l. \tag{10}$$

5. The upper bound vector of variables move to the lower bound of variables as below (with each step one sub problem is formed):

$$U_i = U_i - d_u. \tag{11}$$

Heretofore, half of the sub problems i is formed and difference between the lower bound vector and upper bound vector of variables reaches to zero. For the rest of the sub problems (half of i), the process is as given below.

6. The backward moving of lower bound of variables is as given below:

$$L_i = L_i - d_l. \tag{12}$$

7. The backward moving of upper bound of variables is as given below:

$$U_i = U_i + d_u. \tag{13}$$

8. Now the number of sub problems (for example, 30 sub problems) have formed; therefore, the instructions for running the genetic algorithm, saving the results and displaying the most minimum from all sub problems are necessary. For example, in the matlab software:

```
options = gaoptimset (CrossoverFraction, .2, PopulationSize, 20, CrossoverFcn,
@crossoverheuristic, HybridFcn, @fmincon, MutationFcn, {@mutationgaussian, .5,
.75})
```

[[x, fval, reason,output] = ga(@fitnessfcn, nvars, A, b, Aeq, beq, Lb, Ub,@nonlcon, options)]Fitnessfcn is the objective function that is programmed in a m.file; "nvars" is the number of variables; "A" and "Aeq" are matrices; "$c(x)$" and "ceq(x)" are functions that return vectors; "$f(x)$", "$c(x)$", and "ceq(x)" can be nonlinear functions; "nonlcon" are nonlinear constraints for the problem that is programmed in a m.file.Furthermore in this article, for usual genetic algorithm and proposed genetic algorithm, the basic parameters are the following (with the exception that in the usual genetic algorithm the Population size is varied from 20 to 2000):Crossover rate: .2. Population size: 20. Crossover function:Heuristic.
Hybrid function: fmincon (used algorithm: SQP, Quasi-Newton, line-search).
Mutation:function = Gaussian, Scale = .5, Shrink = .75.
Summary, proposed and optional parameters in matlab are:

```
options = PopulationType:doubleVector
      PopInitRange: [2 × 1 double]
    PopulationSize: 20.00
       EliteCount: 2.00
    CrossoverFraction: 0.20
  MigrationDirection: forward
    MigrationInterval: 20.00
    MigrationFraction: 0.20
       Generations: 100.00
         TimeLimit: Inf
         FitnessLimit: -Inf
```

StallGenLimit: 50.00
StallTimeLimit: 20.00
TolFun: 0.00
TolCon: 0.00
InitialPopulation: [ ]
InitialScores: [ ]
InitialPenalty: 10.00
PenaltyFactor: 100.00
PlotInterval: 1.00
CreationFcn: @gacreationuniform
FitnessScalingFcn: @fitscalingrank
SelectionFcn: @selectionstochunif
CrossoverFcn: @crossoverheuristic
MutationFcn: {[1 × 1 function_handle] [0.50] [0.75]}
HybridFcn: @fmincon
Display: final
PlotFcns: [ ]
OutputFcns: [ ]
Vectorized: off

Videlicet, the difference between the usual genetic algorithm and the proposed genetic algorithm is in the population size, which in the usual genetic algorithm (in numerical examples) varies from 20 to 2000 but in proposed genetic algorithm it is constant in each problem with 20, but the number of sub problems can be varied.

9. Select the most minimum for objective function and a decision variable vector $(x_1^*, x_2^*, \ldots, x_n^*)$ for it.

$$\bar{f} = \min\{\tilde{f}^{h,i}\}. \tag{14}$$

10. Repeat the process that has been explained in the repeated step (from 1 to 10) 10 times.

2.1.3. The stop step

In the proposed genetic algorithm, the stop condition is

$$\bar{f}^{h=1} = \bar{f}^{h=2} = \cdots = \bar{f}^{h=10}. \tag{15}$$

If the stop condition that is mentioned above is not fulfilled, return to the repeated step and increase the number of sub problems until the stop condition is fulfilled.

Many efforts have been made to obtain global point in nonlinear optimization by convexity rule in GAs. Some of these efforts were made using Markov chains [12]. In general, these studies have been relying on a large population size and low mutation rate, that have been done statistically [13]. For instance, it has been shown that convergence can be achieved above a critical population size. Upper bounds have been derived for this critical population size [14]. Other studies have found a bound for the number of iterations necessary to find the global optimum with a pre-specified level of confidence [15,16].

In this paper, for obtaining global point in most difficult situations at the field of nonlinear optimization, namely nonconvex nonlinear optimization, a new technique has been proposed, that relies on the number of sub problems, but with lower population.

3. Numerical examples

Three benchmark problems have been taken from the literature to test the performance of the proposed genetic algorithm. The results were compared with the usual genetic algorithm according to the solution of quality and time, because *the solution quality* and *time of computations* are the main indexes of any algorithm [4]. In addition to this, computational results for three examples of these indexes are listed in Tables 1–3 and 5.

To make a fair comparison, we performed 10 runs on each problem. The problems and results are as follows:

Test 1

$$\text{Min} \quad f(x) = \frac{1}{2}\sum_{i=1}^{n}(X_i^4 - 16xi^2 + 5xi) - 100 \leqslant xi \leqslant 100, \quad i = 1, 2, \ldots, n.$$

This problem has one global minimum [17] at

$$x_i^* = -2.90354 \quad (i = 1, 2, \ldots, n), \quad f(x^*) = n \cdot (-39.1662).$$

Test 2

$$\text{Min} \quad f(x) = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_{2+}j] + (x_1 + 1.42513)^2 + (x_2 + .80032)^2 - 10 \leqslant x_1, \quad x_2 \leqslant 10.$$

This problem has one global minimum [17] at

$$f^* = -186.73909 \text{ at } x^* = (-1.42513, -.80032).$$

Test 3. (Considering the following nonconvex programming problem with eight constraints:)

$$\text{Minimize:} \quad (x) = x_1^3 + (x_2 - 5)^2 + 3(x_3 - 9)^2 - 12x_3 + 2x_4^3 + 4x_5^2 + (x_6 - 5)^2 - 6x_7^2 + 3(x_7 - 2)x_8^2 - x_9x_{10} + 4x_9^3 + 5x_1x_3 - 3x_1x_7 + 2x_8x_7$$

Table 1
Comparison of genetic algorithm and proposed genetic algorithm (test 1 with 10 variables ($n = 10$))

Genetic algorithm				Proposed genetic algorithm				
Run	Size of population	Time (s)	f^*	Run	Size of population	Number of sub problems	Time (s)	f^*
1	20	3	−363.39	1	20	30	10	−391.6617
2	50	4	−349.25	2	20	30	10	−391.6617
3	100	6	−264.43	3	20	30	10	−391.6617
4	150	6	−377.52	4	20	30	10	−391.6617
5	200	12	−377.52	5	20	30	10	−391.6617
6	400	18	−292.70	6	20	30	10	−391.6617
7	450	20	−335.11	7	20	30	10	−391.6617
8	500	21	−377.52	8	20	30	10	−391.6617
9	600	25	−335.11	9	20	30	10	−391.6617
10	800	30	−363.39	10	20	30	10	−391.6617

Table 2
Comparison of genetic algorithm and proposed genetic algorithm (test 1 with 50 variables ($n = 50$))

Genetic algorithm				Proposed genetic algorithm				
Run	Size of population	Time (s)	f^*	Run	Size of population	Number of sub problems	Time (s)	f^*
1	100	10	−1816.94	1	20	40	120	−1958.31
2	200	15	−1831.08	2	20	40	120	−1958.31
3	400	25	−1887.62	3	20	40	120	−1958.31
4	800	90	−1944.17	4	20	40	120	−1958.31
5	2000	120	−1915.90	5	20	40	120	−1958.31
6	2000	120	−1619.03	6	20	40	120	−1958.31
7	2000	120	−1732.12	7	20	40	120	−1958.31
8	2000	120	−1873.49	8	20	40	120	−1958.31
9	2000	120	−1760.39	9	20	40	120	−1958.31
10	2000	120	−1915.90	10	20	40	120	−1958.31

Table 3
Comparison of genetic algorithm and proposed genetic algorithm (test 2)

Genetic algorithm				Proposed genetic algorithm				
Run	Size of population	Time (s)	f^*	Run	Size of population	Number of sub problems	Time (s)	f^*
1	500	12	−186.7309	1	20	30	12	−186.7309
2	500	12	−109.7082	2	20	30	12	−186.7309
3	500	12	26.5725	3	20	30	12	−186.7309
4	500	12	−109.7082	4	20	30	12	−186.7309
5	500	12	−172.2223	5	20	30	12	−186.7309
6	500	12	−186.7309	6	20	30	12	−186.7309
7	500	12	−186.7309	7	20	30	12	−186.7309
8	500	12	−172.2223	8	20	30	12	−186.7309
9	500	12	−186.7309	9	20	30	12	−186.7309
10	500	12	−109.7082	10	20	30	12	−186.7309

$$
\begin{aligned}
\text{subject to:}\quad & -3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4-2x_5x_6x_8+12\geqslant 0,\\
& -5x_1^2-8x_2-(x_3-6)^2+2x_4+40\geqslant 0,\\
& -x_1^2-2(x_2-2)^2+2x_1x_2-14x_5-6x_5x_6\geqslant 0,\\
& .5(x_1-8)^2-2(x_2-4)^2-3x_5^2+x_5x_8+30\geqslant 0,\\
& 3x_1-6x_2-12(x_9-8)2+7x_{10}\geqslant 0,\\
& 4x_1+5x_2-3x_7+9x_8\leqslant 105,\\
& 10x_1-8x_2-17x_7+2x_8\leqslant 0,\\
& -8x_1+2x_2+5x_9-2x_{10}\leqslant 12,\\
& -5\leqslant x_i\leqslant 10,\quad i=1,\ldots,10.
\end{aligned}
$$

This problem was solved by GENOCOPIII (Genetic algorithm for Numerical Optimization of Constrained Problems) and revised by GENOCOPIII [3].The results, which were obtained by these methods, are summarized in Table 4.

The results of solving this problem with genetic algorithm and the proposed genetic algorithm are listed in Table 5.

4. Numerical results

It can be observed from Tables 1–3 and 5 that the proposed genetic algorithm has performed well with high success and precision. *In test 1* with 10 variables, in genetic algorithm not 1 of 10 runs even with 800 for population size and 30 s timing[3], reached the optimization solution ($f(x) = -391.6617$) but in the proposed genetic algorithm all 10 runs reached this optimization solution with 10 s timing (Table 1).

Again in Test 1 with 50 variables, in genetic algorithm not 1 of 10 runs even with 2000 for size of population and 120 s timing reached the optimization solution ($f(x) = -1958.31$), but in the proposed genetic algorithm all 10 runs reached this optimization solution with 120 s timing (Table 2).

In Test 2 with 12 s timing, in genetic algorithm four runs reached the optimization solution ($f(x) = -186.7309$) and in the proposed genetic algorithm again all 10 runs reached the optimization solution Table 3.

In Test 3 (with 60 s timing) not one of the runs in the genetic algorithm reached the optimization solution. However, in the proposed genetic algorithm all runs reached the optimization solution Table 5. Furthermore, the proposed algorithm in comparison with GENOCOPIII and the revised GENOCOPIII (Table 4) has high efficiency in quality solution and timing (compare Table 4 with the proposed genetic algorithm in Table 5).

[3] Computer properties: Intel(R), Pentium(R)M, Processor, 2.13 GHZ and 1 GB of RAM.

Table 4
Comparison of optimal solution between GENOCOPIII and Revised GENOCOPIII

Items	GENOCOPIII	Revised GENOCOPIII
Best solution	−160.163	−216.602
Worst solution	7.513	−13.531
Average	−48.975	−124.312
Mean computational time (s)	763.918	275.193

Reference: [3, p. 121].

Table 5
Comparison of genetic algorithm and proposed genetic algorithm (test 3)

Genetic algorithm				Proposed genetic algorithm				
Run	Size of population	Time (s)	f^*	Run	Size of population	Number of sub problems	Time (s)	f^*
1	1200	60	4.1288	1	20	30	60	−216.6565
2	1200	60	197.8580	2	20	30	60	−216.6565
3	1200	60	118.9106	3	20	30	60	−216.6565
4	1200	60	4.1288	4	20	30	60	−216.6565
5	1200	60	−105.3536	5	20	30	60	−216.6565
6	1200	60	4.1288	6	20	30	60	−216.6565
7	1200	60	4.1288	7	20	30	60	−216.6565
8	1200	60	4.1288	8	20	30	60	−216.6565
9	1200	60	4.1288	9	20	30	60	−216.6565
10	1200	60	4.1288	10	20	30	60	−216.6565

Consequently, the proposed genetic algorithm has high efficiency in solving problems for convergences to the global optimization.

From the aspect of run time, the proposed algorithm runs with low population size (i.e. 20 populations for the proposed genetic algorithm in comparison with 1200 populations for genetic algorithm in Test 3). Furthermore, in the proposed genetic algorithm each sub problem for run is dependent on the other. Therefore, the curse of dimensionality that is widespread in the usual genetic algorithms does not exist in the proposed genetic algorithm.

References

[1] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, 1992.
[2] D.E. Goldberg, Genetic Algorithms in Search Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[3] M. Sakawa, K. Yauchi, An interactive fuzzy satisfying method for multiobjective nonconvex programming problems through floating point genetic algorithms, European Journal of Operational Research 117 (1999) 113–124.
[4] M.S. Osman, M.A. Abo-sina, A.A. Mousa, An effective genetic algorithm approach to multiobjective routing problems (MORPs), Applied Mathematics and Computation 163 (2005) 769–781.
[5] K. Katayama, H. Sakamoto, H. Narihisa, The efficiency of hybrid mutation genetic algorithm for the traveling salesman problem, Mathematical and Computer Modeling 31 (2000) 197–203.
[6] C. Sangit, C. Cecilia, A.L. Lucy, Genetic algorithms and traveling salesman problems, European Journal of Operational Research 93 (1996) 490–510.
[7] H. Aytug, M. khouja, F.E. Vergara, Use of genetic algorithms to solve production and operations management problems: a review, International Journal of Production Research 41 (17) (2003) 3955–4009.
[8] J.G. Koeh, New directions in genetic algorithm theory, Annuals of Operation Research 75 (1997) 49–68.
[9] F.S. Hillier, G.J. Lieberman, Introduction to Operations Research, seventh ed., Mc-Graw Hill, 2001, pp. 668.
[10] Z. Hua, F. Huang, An effective genetic algorithm approach to large scale mixed integer programming problems, Applied Mathematics and computation 174 (2006) 897–909.
[11] D. Ashlock, Evolutionary Computation for Modeling and Optimization, 2006, Springer Science, Business Media Inc., 2005, pp. 257.
[12] A.E. Nix, M.D. Vose, Modeling genetic algorithms with Markov chains, Annals of Mathematics and Artificial Intelligence 5 (1992) 79–88.
[13] R.L. Haupt, S.E. Haupt, Practical Genetic Algorithms, second ed., John Wiley & Sons INC., 2004.
[14] R. Cerf, Asymptotic convergence of genetic algorithms, Advances in Applied Probability 30 (1998) 521–550.

[15] H. Aytug, G.J. Koehler, stopping criteria for finite length genetic algorithms, ORSA Journal of Computing 8 (1996) 183–191.
[16] D. Greenhalgh, Convergence criteria for genetic algorithms, SIAM Journal of Computation 30 (2000) 269–282.
[17] Y.J. Wang, J.S. Zhang, An efficient algorithm for large scale global optimization of continuous functions, Journal of Computational and Applied Mathematics 206 (2007) 1015–1026.