A Genetic Algorithm Based on a New Real Coding Approach

Guoshan Zhang, Wanliang Liu

School of Electrical Engineering and Automation, Tianjin University, Tianjin, 300072, China
zhanggs@tju.edu.cn

***Abstract*—Genetic algorithm is a kind of common method to solve nonlinear programming problems. To improve the computational efficiency of the algorithm, a genetic algorithm based on a new real code (NRCGA) was proposed, which could solve a class of nonlinear programming problems. The new real coded strategy can be used to repair all of the infeasible chromosomes by simply sorting and keeping search within the feasible region. NRCGA is more accurate than the existing methods on equality constraint handling. Many examples show that the new algorithm has high search efficiency and strong robustness.**

Keywords-genetic algorithm; penalty functions; nonlinear programming; constraint-handling; real-coding

I. INTRODUCTION

Genetic algorithm (GA), which simulates the evolution process of creature, was first introduced by Holland in the 1970s [1]. Because GA can solve optimization problems in which objective function is continuous or discontinuous and has inherent parallelism and strong robustness, it is very useful to deal with nonlinear programming problem.

To avoid the shortcoming of weak local search ability of traditional genetic algorithm, many researchers have improved the genetic algorithm in the last decade. GA based on migration and the artificial selection was proposed by [2], which could overcome the premature phenomena of GA. [3] proposed a generic parent-centric recombination operator (PCX) in order to improve crossover operator. To further expand its application scope, PCX was improved by reference [4], and based on PCX, [5] designed so-called Laplace Crossover. A novel mutation operator based on the wavelet was proposed in [6]. [7] introduced a new mutation operator called power mutation for real coded genetic algorithms (RCGA), which outperformed all other GAs considered in his study. To solve constrained optimization problems, constraint-handling technique is as significant as evolutionary algorithm (including genetic algorithm). Penalty function method and multi-objective method are two main measures of constraint handling.

Because Penalty function method is simple and convenient to implement, it is a common constraint-handling technique with wide application. For instance, [8] proposed a segregated genetic algorithm with two different penalized fitness functions, which try to achieve a balance between too large and too small punishment. In 2007, a cooperative evolutionary algorithm was proposed by reference [9]. In [10], a self-adaptive fitness formulation method is presented. However, a main limitation of the penalty function methods is that most of them require a careful fine-tuning of parameters to obtain competitive results.

In the last decade, many scholars tried to handle constraint with multi-objective method. In [11], Runarsson and Yao (RY) introduced a stochastic ranking method to achieve a balance between objective function and penalty functions stochastically, which was regarded as the most classic constraint-handling technique based on evolutionary algorithm [12]. In 2005, this algorithm was further improved by Runarsson and Yao [13]. In [14], Takahama and Sakai proposed a constrained optimization technique by applying the α-constrained method to the nonlinear simplex optimization (αSimplex). Afterward, Cai and Wang [15] proposed CW algorithm in 2006. To handle constraint, these multi-objective methods perform better property than penalty function methods, but these methods increase additional parameters and require fine-tuning these parameters carefully.

To meet the condition of constraint and avoid adding additional parameters, this paper introduced a genetic algorithm based on a novel real coding approach (NRCGA). This method adopts a new kind of real coding strategies to keep all the offspring feasible after selection, crossover, mutation, simple sorting, encoding and decoding. The new algorithm has significant features. Firstly, the new algorithm has no additional parameters and requires encoding and decoding. Secondly, it can handle a kind of constraints independently and solve complex constrained optimization problems combining with other constraint-handling techniques. Besides, NRCGA is accurate and just has rounding error when it handles equality constraint. Moreover, it searches for solutions within feasible region all the time, converges to the global optimal solution rapidly and has strong robustness.

II. PROBLEM MODEL

The first nonlinear programming model is described as:

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & \sum_{j=1}^{k_i} g_{ij}(x_{ij}) \le a_i, \\ & x_{ij} \in R^+, \end{aligned} \tag{1}$$

where,

$$a_i \ge 0,\ i = 1,2,..., m,$$

$$\sum_{i=1}^{m} k_i = n,\ k_i \ge 2, n \ge 2,$$

$$x=(x_{11},x_{12},\ldots,x_{1k_1},x_{21},x_{22},\ldots,$$
$$x_{2k_2},\ldots,x_{m1},x_{m2},\ldots,x_{mk_m})^T,$$

j, m and n are all positive integers. R^+ indicates nonnegative real number set. g_{ij} is a nonnegative continuous function of R^+ and the range of its value is $S_i([0,a_i]\subseteq S_i\subseteq R^+)$. f is a function of n variables. At least one function of f and g_{ij} is nonlinear function.

The second nonlinear programming model is described as:

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & \sum_{j=1}^{k_i} g_{ij}(x_{ij}) \le a_i, \\ & 0\le x_{ij}\le b_{ij}, \end{aligned} \tag{2}$$

where,

$$a_i\ge 0,\ i=1,2,\ldots,m,$$
$$\sum_{i=1}^{m} k_i=n,\ k_i\ge 2,\ n\ge 2,$$
$$0\le b_{ij}\le +\infty,\ 1\le j\le k_i,$$
$$x=(x_{11},x_{12},\ldots,x_{1k_1},x_{21},x_{22},\ldots,$$
$$x_{2k_2},\ldots,x_{m1},x_{m2},\ldots,x_{mk_m})^T,$$

j, m and n are all positive integers. g_{ij} is a nonnegative continuous monotone increasing function of R^+ and the range of its value is $S_i([0,a_i]\subseteq S_i\subseteq R^+)$. f is a function of n variables. At least one function of f and g_{ij} is nonlinear function. At least one of b_{ij} is a finite nonnegative real number.

The distinction of model (1) and model (2) is that there are upperbound-constraint of variables and the requirements of g_{ij}'s monotonicity in model (2), and there are no these requirements in model (1).

III. A GENETIC ALGORITHM BASED ON A NOVEL CODING APPROACH (NRCGA)

A. New Encoding and Decoding

Encoding: Suppose any feasible solution of model (1) or (2) is

$$x=(x_{11},x_{12},\ldots,x_{1k_1},\ldots,x_{m1},x_{m2},\ldots,x_{mk_m}),$$

let $x_i=(x_{i1},x_{i2},\ldots,x_{ik_i})$, in that way $x=(x_1,x_2,\ldots,x_m)$. For any i, let $y_i=(y_{i1},y_{i2},\ldots,y_{ik_i})$ and $y_{it}=\sum_{j=1}^{t} g_{it}(x_{it})$, $t=1,2,..,k_i$, in that way $y=(y_1,y_2,\ldots,y_m)$ is model (1) or (2)'s encoding.

Remark 1: Because all of g_{ij} are nonnegative functions, we can see $0\le y_{i1}\le y_{i2}\le\ldots\le y_{ik_i}\le a_i$. Therefore, we need to generate k_i random-uniform numbers ranging from 0 to a_i and sorting these numbers from small to large order.

Decoding: Suppose model (1) or (2)'s chromosome after encoding is $y=(y_1,y_2,\ldots,y_m)$. For any $1\le i\le m$, $y_i=(y_{i1},y_{i2},\ldots,y_{ik_i})$. Let

$$\begin{aligned} z_i &= (z_{i1},z_{i2},\ldots,z_{ik_i}) \\ &= (y_{i1},y_{i2}-y_{i1},\ldots,y_{ik_i}-y_{ik_i-1}) \end{aligned}.$$

Then, we can solve n single-variable equations $g_{ij}(x_{ij})=z_{ij}$. It can conclude that $0\le z_{ij}=y_{ij}-y_{i\,j-1}\le a_i$ from Remark 1 . Besides, g_{ij} is a nonnegative continuous function of R^+ and the range of g_{ij}'s value is $S_i([0,a_i]\subseteq S_i\subseteq R^+)$. Therefore, equation $g_{ij}(x_{ij})=z_{ij}$ has at least one nonnegative root x_{ij}^*. Let $x_{ij}^{(1)}$ indicate one nonnegative root of equation $g_{ij}(x_{ij})=z_{ij}$, then $x_{(1)}=(x_{11}^{(1)},x_{12}^{(1)},\ldots,x_{1k_1}^{(1)},\ldots,x_{m1}^{(1)},x_{m2}^{(1)},\ldots,x_{mk_m}^{(1)})$, is the chromosome of model (1) after decoding. The chromosome of model (2) after decoding is $x_{(2)}=(x_{11}^{(2)},x_{12}^{(2)},\ldots,x_{1k_1}^{(2)},\ldots,x_{m1}^{(2)},x_{m2}^{(2)},\ldots,x_{mk_m}^{(2)})$,where $x_{ij}^{(2)}=\min\{x_{ij}^{(1)},b_{ij}\}$.

B. Other Operaters of GA

Population initialization: suppose the size of initial population is P. Firstly, for every i, we generate k_i random-uniform numbers ranging from 0 to a_i, sort these numbers from small to large order and then get an array $y_i=(y_{i1},y_{i2},\ldots,y_{ik_i})$. Secondly, let array $z=[y_1,y_2,\ldots,y_m]$, which is a chromosome of model (1) or (2) after encoding. Finally, repeat this process P times and you can get P chromosomes which compose the initial population.

Fitness function: we just choose objective function as fitness function in this paper.

Selection, crossover and mutation: offspring is selected according to stochastic universal sampling. Hybrid crossover operator is adopted, which consists of linear recombination with mutation features and two-point crossover. We choose mutbga which is a real value variation function from reference [16] for mutation function.

Repair: First, $P \times n$ matrix namely P chromosomes are partitioned into m parts and each part is a $P \times k_i$ ($1 \le i \le m$, $\sum_{i=1}^{m} k_i = n$) submatrix. Then we sort the elements of every rows of all of the submatrixes from small to large order and then get a new $P \times n$ matrix namely P chromosomes of offspring.

Handling equality constraint: y_{ik_i} is replaced by a_i in all of the genetic operations once the i-th inequality constraint of model (1) or (2) becomes equality constraint.

Remark 2: The algorithms proposed in this paper can handle model (1) or model (2) independently. If we add even more complex equality constraints or inequality constraints to model (1) or model (2), the additional constraints could be handle with penalty function method or multi-objective method.

Procedure NRCGA:

```
population initialization;
Setting maximum number of generations is $\max gen$;
$t = 0$;
while $t < \max gen$ do
    selection;
    crossover;
    mutation;
    repair;
    $t = t + 1$;
end.
```

IV. NUMERICAL EXPERIMENTS

NRCGA will be compared with traditional real-coded genetic algorithm (RCGA) and standard genetic algorithm (SGA) firstly. NRCGA and RCGA adopt the same selection, crossover and mutation operators, of which the key distinction is population initialization, encoding, decoding and fitness function. RCGA generates initial population according to traditional method and doesn't require decoding operator. RCGA and SGA add penalty function $p(x)$ to fitness function,

$$p(x) = \sum_{i=1}^{m} c_i p_i(x),$$

where, $p_i(x) = \max\{0, \sum_{j=1}^{k_i} g_{ij}(x_{ij}) - a_i\}$,

$c = (c_1, c_2, ..., c_m)$ and c is m dimension coefficient vector.

Test problems 1-5 can be solved by the new algorithm independently. Table 1 records simulation results of problems 1-4 and Table 2 records simulation results of problem 5. Problems 1-4 are typical nonlinear programming problems from reference [17-19]. Problem 5 is the third problem (g03) of 13 well-known benchmark problems (g01-g13). The remaining 12 benchmark problems can be solved by the algorithm combining NRCGA with other constraint-handling techniques.

All computational experiments were conducted on a notebook HP CQ40-406 with matlab2009.

In Table 1, we set three algorithms' initial population size P =100, maximum number of generations G =50, genetic generation gap $GGAP$ =0.9, the times of independent runs of each algorithm $Gtime$ =10. For NRCGA and RCGA, we set probability of linear recombination with mutation feature $p1 = 0.8$, two-point crossover probability $p2 = 0.3$, mutation probability $p3 = 0.05$. For SGA, we set crossover probability $p4 = 0.8$, mutation probability $p5 = 0.01$. Penalty coefficients c of RCGA and SGA for problems 1-4 are set by a careful fine-tuning.

Because the constraints of problems 1-4 are very simple, extra computation time of decoding and repair operator of NRCGA is relatively small when it is compared with the entire runtime of the algorithm. Similarly, computation cost of penalty term of RCGA and SGA is also very small. Consequently, the computational complexity of these three algorithms is very nearly the same. The number of evaluations of the objective function for these three algorithms is all 5000.

Table 1 shows velocity of convergence of NRCGA is much faster than that of RCGA and SGA. Compared with RCGA and SGA, NRCGA remarkably outperforms them in terms of the best, mean, and worst objective function values and the standard deviations. In addition, NRCGA can find all of the global optimal solutions of problems 1-4, but RCGA and SGA can't.

Problem 1 [17]

$$\begin{aligned} \min \quad & f(x) = 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 \\ & \qquad + 2x_1x_3 - 8x_1 - 6x_2 - 4x_3 + 9 \\ s.t. \quad & x_1 + x_2 + 2x_3 \le 3 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

The global minimum is $x^* =$ (4/3, 7/9, 4/9), where $f(x^*)$ =1/9 and $c = 2$.

Problem 2 [18]

$$\begin{aligned} \max \quad & f(x) = \sqrt{x_1} + \sqrt{x_2} + \sqrt{x_3} \\ s.t. \quad & x_1^2 + 2x_2^2 + 3x_3^2 \le 1 \\ & x_1, x_2, x_3 \ge 0 \end{aligned}$$

The global maximum is $x^* =$ (0.63409848, 0.39945701, 0.30484275), where $f(x^*) =$ 1.98045479 and $c = -10$.

Problem 3 [19]

$$\max \quad f(x) = 5x_1 - x_1^2 + 8x_2 - 2x_2^2$$

$$s.t.\quad 3x_1 + 2x_2 \le 6$$
$$0 \le x_1 \le 2$$
$$0 \le x_2 \le 3.$$

The global maximum is $x^* = (1, 1.5)$, where $f(x^*) = 11.5$ and $c = -20$.

Problem 4 [19]

$$\max \quad f(x) = \sum_{i=1}^{9} x_i x_{i+1} + \sum_{i=1}^{8} x_i x_{i+2} + x_1 x_9 + x_2 x_{10} + x_1 x_5 + x_4 x_7$$

$$s.t.\quad \sum_{i=1}^{10} x_i \le 1,$$
$$0 \le x_i \le 0.5,\ i = 1,2,...,10$$

The global maximum is $x^* = (0,\ 0,\ 0,\ 0.25,\ 0.25,\ 0.25,\ 0.25,\ 0,\ 0,\ 0)$, where $f(x^*) =$ 0.375 and $c = -10$.

Problem 5 [11]

$$\min \quad f(x) = \left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$$

$$s.t.\quad h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$
$$0 \le x_i \le 1,\ i = 1,2,...,\ n$$

where $n = 10$. The global minimum is at $x_i^* = 1/\sqrt{n}\left(i = 1,2,...,n\right)$, where $f(x^*) = 1$.

Table.1 Computational results of problem 1-4

		Problem 1[17]	Problem 2[18]	Problem 3[19]	Problem 4[19]
optimal		0.111111	1.980455	11.5000	0.3750
SGA	best	0.111151	1.979552	11.4996	0.2902
	mean	0.113536	1.977024	11.4585	0.2587
	worst	0.116891	1.971903	11.2411	0.2254
	St.dev	2.4e-3	2.3e-3	7.7e-2	2.0e-2
RCGA	best	0.111135	1.979214	11.4989	0.3663
	mean	0.111698	1.975347	11.4802	0.3314
	worst	0.113514	1.962437	11.4344	0.2962
	St.dev	6.9e-4	5.0e-3	2.2e-2	2.2e-2
NRCGA	best	0.111111	1.980455	11.5000	0.3750
	mean	0.111111	1.980450	11.5000	0.3747
	worst	0.111111	1.980439	11.5000	0.3737
	St.dev	7.2e-9	5.8e-6	2.4e-7	3.7e-4

Table.2 Computational results of problem 5

	SAFF[10]	RY[11]	IRY[13]	αSimplex[14]	CW[15]	NRCGA	NRCGA2
best	-1.000	-1.000	-1.001	-1.001	-1.000	-1.000	-1.000
mean	-1.000	-1.000	-1.001	-1.001	-1.000	-1.000	-1.000
worst	-1.000	-1.000	-1.001	-1.001	-1.000	-1.000	-1.000
St.dev	7.5e-5	1.9e-4	6.0e-9	8.5e-14	2.8e-16	5.3e-11	2.9e-16
δ	Y	Y	Y	Y	N	N	N
Max_FES	1400000	350000	350000	330000	350000	350000	50000

NRCGA2 has evolved from NRCGA and is almost the same as NRCGA. The key distinction between NRCGA2 and NRCGA is their crossover operator and repair operator. The initial population of NRCGA2 is the decoding of the initial population of NRCGA. With regard to crossover operator, NRCGA2 only adopt intermediate recombination. Repair operator of NRCGA2 is divided into two steps. The first step is to take absolute value of corresponding real number of each chromosome genes. The second step is to get unit vector of the corresponding real vector of each chromosome.

In table 2, for NRCGA, we set initial population size P =100, maximum number of generations G =3500, genetic generation gap $GGAP$ =0.9, times of independent runs of the algorithm $Gtime$ =30, probability of linear recombination with mutation feature $p1 = 0.8$, two-point crossover probability $p2 = 0.3$, mutation probability $p3 = 0.05$. For NRCGA2, we set initial population size P =100, maximum number of generations G =500, genetic generation gap $GGAP$ =0.9, times of independent runs of the algorithm $Gtime$ =30, probability of intermediate recombination $p1 = 0.8$, mutation probability $p3 = 0.05$. From the sixth row in Table 2, 'Y' means that it needs δ and 'N' means that it doesn't need δ. If parameter δ is added, equality constraints have been converted into inequality constraints, such as $|h_1(x)| \le \delta$, using the degree of violation $\delta = 0.0001$. Missing parameter δ means that the algorithm just has rounding error when it handles equality constraint. 'Max_FES' signifies the maximum number of function evaluations for problem 5.

The sixth row in Table 2 shows that CW, NRCGA and NRCGA2 needn't use parameter δ and almost have no constraint violation, while the remaining four algorithms' constraint error are all 0.0001. The second, third and fourth rows in Table 2 suggest that IRY and α Simplex can find the optimal solution (not global optimal solution) with the degree of constraint violation $\delta = 0.0001$, but the remaining five algorithms can reach the global optimal solution in theory accurately. From line 7 in Table 2, we can see that NRCGA2 has minimum computational cost, but SAFF has maximum computational cost. From line 5 in Table 2, it is obvious that NRCGA2 and CW have strongest robustness because of their minimum standard deviation. NRCGA has stronger robustness than SAFF, RY and IRY. From references [10-15], RY, IRY, αSimplex and CW are all algorithms requiring several additional parameters, while SAFF, NRCGA and NRCGA2 don't need additional parameters. In a word, for problem 5, NRCGA2 is the best algorithm, NRCGA and CW are second.

V. Conclusions

A genetic algorithm based on a new real coding approach (NRCGA) was proposed in this paper. As solving model (1) or model (2), it has high search efficiency and strong robustness, and keeps searching within the feasible region and doesn't need additional parameters. If we add even more complex equality constraints or inequality constraints to model (1) or model (2), the additional constraints can be handled with penalty function method or multi-objective method. So the new algorithm can be considered as an efficient auxiliary algorithm of penalty function method or multi-objective method for constrained optimization problems.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant No 61074088.

References

[1] J.H. Holland. Adaptation in Nature and Artificial Systems[M]. Boston: MIT Press, 1975.

[2] J.C. PothS, T.D. Giddens and S.B. Yadaw. The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection[J]. IEEE Trans. on Systems, Man and Cybernetics, 1997, 24(1): 73-56.

[3] Deb K., Anand A., Joshi D.. A computationally efficient evolutionary algorithm for real-parameter evolution[J]. Evolutionary Computation Journa1, 2002, 10(4): 371-395.

[4] Sinha A., Tiwari S., Deb K.. A population-based, steady state procedure for real-parameter optimization[C]. The 2005 IEEE Congress on Evolutionary Computation, Location: Edinburgh, Scotland, Vol. 1 (2005), pp. 514-521.

[5] Deep K., Thakur M.. A new crossover operator for real-coded genetic algorithms. Applied Mathematics and Computation 2007, 188(1): 895-911.

[6] Ling S. H., Leung F. H. F. An improved genetic algorithm with average-bound crossover and wavelet mutation operations[J]. Soft Computing, 2007, 11: 7-31.

[7] Deep K., Thakur M.. A New Mutation Operator for Real-coded Genetic Algorithms[J]. Applied Mathematics and Computation, 2007, 193(1): 211-230.

[8] Le RRG, Knopf-Lenoir C, Haftka RT. A segregated genetic algorithm for constrained structural optimization[C]. In: Proc. of the 6th Int. Conf. on Genetic Algorithms. San Francisco: Morgan Kaufinan Publishers, 1995, 558-565.

[9] Huang F, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization[J]. Applied Mathematics and computation, 2007, 186(1): 340-356.

[10] Farmani R, Wright JA. Self-Adaptive fitness formulation for constrained optimization. IEEE Trans. on Evolutionary Computation. 2003, 7(5): 445-455.

[11] Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Trans. on Evolutionary Computation, 2000, 4(3): 284-294.

[12] Wang Y, Cai ZX, Zhou YR, Xiao CX. Research and development of constrained optimization evolutionary algorithms[J]. Journal of Software, 2009, 20(1): 11-29.

[13] Runarsson TP, Yao X. Search biases in constrained evolutionary optimization[J]. IEEE Trans. On Systems Man, Cybernetics, 2005, 35(2): 233-243.

[14] Takahama T, Sakai S. Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations[J]. IEEE Trans. on Evolutionary Computation, 2005, 9(5): 437-451.

[15] Cai ZX, Wang Y. A multiobjective optimization based evolutionary algorithm for constrained optimization[J]. IEEE Trans. on Evolutionary Computation, 2006, 10(6): 658-675.

[16] Lei Yinjie, Zhang Shanwen, Li Xuwu, Zhou Chuangming. Matlab genetic algorithm toolbox and its application[M]. Xi' an: Xi' an Electronic Science and Technology University Press, 2005.

[17] Shi Hongyan, Su Xiaoming. Applicable Intelligent Optimization methods[M]. Dalian: Dalian University of Technology Press, 2009.

[18] Li Donghui, Tong Xiaojiao, Wan Zhong. Numerical Optimization Algorithms and Theory[M]. Beijing: Science Press, 2010.

[19] Andrea Reese. Random number generations in genetics algorithms for unconstrained and constrained optimization[J]. Nonlinear Analysis, 2009, 71: 679-692.