A New Kind of Handling Constraint Method for Optimization

Jiechang Wen
Faculty of Applied Mathematics
Guangdong University
of Technology
Guangzhou, CHINA 510009
wjcpig@126.com

Xuefei Yao
Faculty of Applied Mathematics
Guangdong University
of Technology
Guangzhou, CHINA 510009
yaoxuefei01@126.com

***Abstract*—This paper proposes a new kind of constraint handling method for optimization. In the proposed method, a transformation of constraint functions to be another objective function will be discussed in details. With this technique we can transform the constraint problem to be unconstraint multi-objective optimization and use the multi-objective optimization methods to find feasible solutions. The performance of the algorithm is tested on 13 benchmark functions in the literature. The results show that we can find very good solutions by this approach.**

Keywords-constraint handling; evolutionary algorithm; optimiztion methods

I. INTRODUCTION

Many optimization problems in science and engineering have some constraints. Evolutionary algorithms have been successful applied to solve these problems. However, evolutionary algorithms naturally perform unconstraint search. Therefore, they need additional mechanisms to handle constraints in their fitness function and selection. The challenges in constrained optimization problems are the various limits on the decision variables, the constraints involved, the interference among constraints, and the interrelationship between the constraints and the objective functions [1], and so on. Some constrained optimization problems which come from science and engineering are difficult to solve. Due to the extremely difficult to find feasible solutions, finding feasible solutions may be a big challenge.

The general constraint programming problem can be formulated as solving the objective function

$$\begin{cases} \max f_0(x) = f(x_1, x_2, ..., x_n) \\ s.t \quad g_i(x) \geq 0 \quad i = 1, 2, \cdots, q \\ x \in \prod_{i=1}^{n} [a_i, b_i] \end{cases} \tag{1}$$

where x is decision variable, $f_0(x)$ is objective function, $a_i$is the lower boundary of x, b_i is the upper boundary of x, $g_i(x)$ is the i^{th} constraint. There are q constraints which are required to be satisfied by the optimum solution. The presence of constraints will restrict the search space to a feasible region $F \subseteq S$, where a usable solution can be found. The inequality constraints that satisfy $g_j(x) = 0$ at the global optimum solution are called active constraints. All equality constraints are active constraints.

The most popular constraint handling technique is the method based on penalty functions. The penalty functions have been used in many evolutionary algorithms and achieve good results. However, penalty functions have several limits. They require a good tuning to make the most appropriate penalty factors, and behave ill when feasible region is disjoint.

This paper proposes a handling constraint method by using the max-min strategy [1]. It transforms the constraints to be another objective function which can transform the constraint problem to a unconstraint multi-objective optimization.

The remainder of this paper is organized as follows. Section II reviews related works of handling constrained optimization problems. In Section III, the proposed algorithm is presented in details. Section IV discusses the results obtained for the benchmark functions. Finally, Section V presents some concluding remarks and relevant observations.

II. LITERATURE SURVEY

Before presenting the proposed algorithm, we will first review some of the related works in this field of study.

The penalty functions, which were proposed in 1940s and later expanded by many researchers, are the simplest and the most commonly used methods. Still it has some limits. In order to overcome these limitations, many methods based on penalty function have been proposed, such as death penalty function method, static penalty function method and dynamic penalty function method etc.

In [2] and [3], methods based on an adaptive penalty function and a distance measure are employed. The two proposed algorithms basically modify the objective function of an individual using its distance measure and penalty value. Two types of penalties are added to each infeasible individual to identify the best infeasible individuals in the current population. The amount of each of the two penalties added is controlled by the number of feasible individuals currently present in the population.

In [11], three selection criteria based on feasibility is proposed:

978-0-7695-4297-3/10 $26.00 © 2010 IEEE
DOI 10.1109/CIS.2010.30

1) Between 2 feasible solutions, the one with the highest fitness value wins.
2) If one solution is feasible and the other one is infeasible, the feasible solution wins.
3) If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

In [4], a boundary approach is included as a constraint-handling technique in an algorithm inspired by the ant colony metaphor to solve single constraint optimization problems. It provides an alternative approach to reach the boundary between the feasible and infeasible space, which could be useful when facing problems with active constraints. However, it does not work so well when facing inactives.

In [5], Runarsson and Yao introduced a stochastic ranking method to achieve a balance between objective and penalty functions stochastically. A probability factor Pf is used to determine whether the objective function value or the constraint violation value determines the rank of each individual.

III. PROPOSED ALGORITHM

A. Transformation of constraint function

We first give the transformation of constraint functions to another objective function. Let $c_j(x^i)$ be the j^{th} constraint of x^i. Let $c_j(x^i)$ be the j^{th} constraint of x^i. Set $TC_j(x^i)$ to be the $min\{c_j(x^i)$. Compute the min of $TC_j(x^i)$ of each constraint of all individuals and make it be $MinTC_j$. The feasibility of the j^{th} constraint of x^i is defined as $(1-\frac{TC_j(x^i)}{MinTC_j+\alpha})$, where α is the control parameter. The feasibility of x^i is $t(x^i)=\sum_{j=1}^{q}(1-\frac{TC_j(x^i)}{MinTC_j+\alpha})$. Therefore, $t(x^i)\in[0,q]$. When $t(x^i)=q$, it means x^i is a feasible individual. The bigger of $t(x^i)$ is, the constraint is less violative.

After the transformation is done, the transformation become the second object, which means we can translate the constraint optimization with single object to a unconstraint multi-objective optimization

$$\begin{cases} \max f_1(x)=t(x_1,x_2,...,x_n) \\ \max f_0(x)=f(x_1,x_2,...,x_n) \\ x\in\prod_{i=1}^{n}[a_i,b_i] \end{cases} \quad (2)$$

B. The equality of two optimizations

Definition 1: For constraint optimization of (1), if there is an individual $x^*\in F$ that compared to any individual $x\in F$, $f_0(x^*)\geq f_0(x)$, then the x^* is the global optimal solution of (1). Here F means feasible space.

Theorem 1: The necessary and sufficient conditions of x^* are that the global optimal solution of (1) is $x^*\in F\cap E_w(f,X)$ and $f_0(x^*)=max_{x\in F}f_0(x)$. Here $E_w(f,X)$ is the weakly efficient solution.

The sufficient condition is obvious. From the condition that x^* is the global optimal solution of (1), we can know that $x^*\in F$ and $f_0(x^*)=max_{x\in F}f_0(x)$ .Suppose $x^*\notin E_w(f,X)$, then there is a solution $x_1\in X$, which satisfy $f_i(x^*) < f_i(x_1)$ $(i=1,2,..,m)$. It also means $t(x^*)<t(x_1)$. Obviously this is against $x^*\in F$. The necessary condition therefore is proved.

From theorem 1 we can know that if we can find out the weakly efficient solution of (2), then the solution with max $f_0(x)$ is the global optimal solution of (1).

C. Frame algorithm

The frame algorithm is as follows.

Step 1 Initialization: Set the population size N, the number of evolution generation gen and the maximum number of evolution generation maxgen, give the probability P_m in mutation. Design the weight vectors $w^1,w^2,\dots,w^N$. Compute the object function value and the constraint functions values of each solution in P'. Transform the constraint functions to object function $t(x)$.

Step 2 Reproduction: Based on the information of old population P, use crossover and mutation operator to generate N new solutions and set them to be Q.

Step 3 Selection: Select N solutions from $P\cup Q$ by using max-min strategy [8], and replace all solutions in P by them.

Step 4 Stopping Criterion: If the stopping criterion is satisfied, stop, otherwise, go to **Step 2**.

In the following, we give and discuss the details of reproduction and selection.

1) Reproduction

The crossover and mutation in [1] will be adopted in our study. We provide a short introduction, anyone who wants to know more can check [1] for details.

In [1], every individual has its corresponding external set and all individuals in P are classified into T class. The crossover is operated between an individual in P and another individual randomly chosen from its corresponding external set. Suppose x^i in P belongs to the t^{th} class $(t\epsilon(1,2,\dots,T))$, choosing the x^j randomly in the t^{th} class external set. After crossover, the new individual is generated as follows:

$$x^c(x)=x^i+rc.(x^i-x^j)$$

where $rc=rand.(1-rand^{-\left(1-\frac{gen}{maxgen}\right)^{0.7}})$, $rand$ is a random number in [-1,1], gen is the current generation and $maxgen$ is the max generation.

After crossover, every component in x^c is mutated with the probability of p_m and x^c is mutated once at least. If h^{th} component of x^c is mutated, then:

$$x_h^c=x_h^c+rm.(x_h^{max}-x_h^{min})$$

where $rm=0.15.rand.(1-rand^{-\left(1-\frac{gen}{maxgen}\right)^{0.7}})$, other parameters are the same as crossover.

2) Selection

a) Construction of fitness

The fitness function utilizes the weighted min-max strategy which is similar to the one in literature [1]. The weight vectors are designed as follows.

$$\begin{cases} w_1(x) = \dfrac{1}{\cos\theta_1} \\ w_2(x) = \dfrac{1}{\sin\theta_1 \cos\theta_2} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ w_{m-1}(x) = \dfrac{1}{\sin\theta_1 \sin\theta_2 \dots \sin\theta_{m-2} \cos\theta_{m-1}} \\ w_{m-1}(x) = \dfrac{1}{\sin\theta_1 \sin\theta_2 \dots \sin\theta_{m-2} \sin\theta_{m-1}} \end{cases}$$

Set $h_j(x^i) = f_j(x^i) - f_j^*, j = 0,1$

where $f_1^* = 0.7 * dim_c - dim_c * 0.7 * selta * 2$, and f_0^* is the minimum of the objective function $f_0(x)$ of the feasible solutions in $P \cup Q'$. And dim_c means the number of the constraint function, $selta = 1 - 0.001^{(1-\frac{gen}{maxgen})^5}$, gen is the current generation, and *maxgen* is the max generation.

The i^{th} fitness function is designed as:

$$F_i(x) = \min_{0 \le j \le 1} \{w_j^i h_j(x)\}$$

b) Selection of individuals

Step 4.1: Transform the objective function $f_j(x^i)$ to $h_j(x^i)$ $(x^i \epsilon P \cup Q')$.

Step 4.2: Compute the projection of $h_j(x^i)$ to unit hypersphere as

$$u_j(x^i) = \frac{h_j(x^i)}{(\sum_{k=1}^{m} h_k(x^i))^{\frac{1}{2}}}$$

where $j = 0,1$, and $x^i \epsilon Q'$

Step 4.3: for each weight vector w^i, compute the fitness of each individual and select the individual with max fitness.

IV. Experimental Results

A. Experimental Setup

The problems studied include a set of well-known test cases traditionally adopted in the specialized literature [6]. The function can also find in [5]. We are going to test 13 functions with comparison to [3] (denoted as SAPCO). From Table I [3] we can observe that the test functions involve various types of problems. Some are maximization problems while others are minimization problems. We transform all the problems to minimization problems. The functions also vary from linear, nonlinear, quadratic, cubic, to polynomial. The number and the types of the constraints (LI means linear inequality, NE means nonlinear equality, and NI means nonlinear inequality) are also different. The feasibility ratio r is an estimate of the ratio ρ of the feasible space to that of the entire search space. The number of active constraints is represented by a. n denotes the number of decision variables involved.

For each test function we applied the proposed algorithm 50 independent runs, and the generation size was set to 4994 (the total function evaluation is therefore 500 000).

B. Parameter Setting

The parameters of the propose algorithms are set as follows: The population size *N* is 100 and $p_m = \frac{1}{n}$.

TABLE I. Summary of main characteristics of the benchmark problems

Test function		n	Type of function	ρ	LI	NE	NI	a
g01	Min	13	Quadratic	0.011%	9	0	0	6
g02	Max	20	Nolinear	99.990%	1	0	1	1
g03	Max	10	Nolinear	0.002%	0	1	0	1
g04	Min	5	Quadratic	52.123%	0	0	6	2
g05	Min	4	Nolinear	0.000%	2	3	0	3
g06	Min	2	Nolinear	0.006%	0	0	2	2
g07	Min	10	Quadratic	0.000%	3	0	5	6
g08	Max	2	Nolinear	0.856%	0	0	2	0
g09	Min	7	Nolinear	0.521%	0	0	4	2
g10	Min	8	linear	0.001%	3	0	3	3
g11	Min	2	Quadratic	0.000%	0	1	0	1
g12	Max	3	Quadratic	4.779%	0	0	93	0
g13	Min	5	Nolinear	0.000%	0	3	0	3

C. Comparative Study

The results the comparison are summarized in Table II and Table III. Table II summaries the comparison between the proposed algorithm and SAPCO. The best, the worst, the mean and the median results of each test function are reported. The numbers of infeasible runs from the 50 trials for each test function are also found in the same table. In addition we have highlighted the best values we found that are identical to the already known optimum values.

From Table II we can notice that the proposed algorithm was able to find feasible solutions for almost all the 50 runs of the 13 test functions except for g10 and g13. g10 has 4 infeasible runs and g13 has 2 infeasible runs. That is because in the proposed algorithm all equality constraints were relaxed by a threshold value of 0.0001, which is very small. Once the threshold value is set to 0.0002, all 50 runs of the 13 test functions can find feasible solutions.

Our algorithm was also able to produce very good results for all of the benchmark functions. As can be seen in Table II, the proposed algorithm has found the known optimum values in 5 of the 13 functions. In g01, g03, g08 and g11 values exactly equal to the already known optimum values are found. In test function g06 values are less than the known optimum values are found. However, that is because all equality constraints have been relaxed by a threshold value of 0.0001. In the rest of the test functions the algorithm also found very good results which are very close to the known optimum values. The optimum values could not be found in g02 and g05 but the best results found are very close. In g07 and g09 the results our algorithm produced are again very close to the known optimum values. Nonlinear equality and nonlinear inequality constraints imposed difficulty in finding the optimum value in g10 and g13. As a result, there are some runs that couldn't find the feasible solutions in g10 andg13. Compared to SAPCO, we

can find that the proposed algorithm has a better performance in g01, g02, g03, g06, g07, g08, g09 and g11 by mean of 50 runs.

Table III compares the best results of the proposed algorithm with 5 other related algorithms. The stochastic ranking [5] had produced the best results in these algorithms. However this algorithm only produced feasible solutions in only 6 out of the 30 runs performed in test function g10. That is much bigger than the proposed algorithm. In general we can say that our algorithm can perform as good as or in some cases better than other algorithms in the related field.

TABLE II. RESULTS COMPARISON BETWEEN THE PROPOSED ALGORITHM AND SAPCO

Test function	Optimum value	algorithm	best	worst	mean	median	Standard deviation	Infeasible runs
g01	-15.000	Proposed	**-15.000**	-14.999	-14.999	-15.000	0.000333	0
		SAPCO	**-15.000**	-13.097	-14.552	-14.966	0.700	0
g02	0.803619	Proposed	0.803607	0.803577	0.803593	0.803594	0.000007	0
		SAPCO	0.803202	0.745712	0.755798	0.789398	0.133210	0
g03	1.000	Proposed	**1.000**	1.000	1.000	1.000	0.00009	0
		SAPCO	**1.000**	0.887	0.964	0.971	0.301	0
G04	-30665.539	Proposed	-30658.556	-30069.318	-30506.829	-30558.507	149.321880	0
		SAPCO	-30665.401	-30656.471	-30659.221	-30663.921	2.043	0
g05	5126.498	Proposed	5126.585	5508.640	5260.830	5156.140	345.359669	0
		SAPCO	5126.907	5564.642	5214.232	5208.897	247.476	0
g06	-6961.814	Proposed	-6961.932	-6961.745	-6961.9	-6961.901	0.029503	0
		SAPCO	-6961.046	-6943.304	-6953.061	-6953.823	5.876	0
g07	24.306	Proposed	24.353	25.459	24.607	24.563	0.191898	0
		SAPCO	24.838	33.095	27.328	25.415	2.172	0
g08	0.095825	Proposed	**0.095825**	0.095825	0.095825	0.095825	3.94e-012	0
		SAPCO	**0.095825**	0.092697	0.095635	0.095825	0.001055	0
g09	680.630	Proposed	680.694	681.373	680.903	680.877	0.149569	0
		SAPCO	680.773	682.081	681.246	681.235	0.322	0
g10	7049.331	Proposed	7331.455	18163.166	11535.154	11829.958	2335.46007	4
		SAPCO	7069.981	7489.406	7238.964	7201.017	137.773	0
g11	0.750	Proposed	**0.750**	0.750	0.750	0.750	0.000015	0
		SAPCO	0.749	0.757	0.751	0.750	0.002	0
g12	1.000000	Proposed	0.736304	0.736296	0.736302	0.736302	0.000002	0
		SAPCO	**1.000000**	0.999548	0.999940	1.000000	0.000141	0
g13	0.053950	Proposed	0.096277	17.390815	1.557133	0.932593	2.900733	3
		SAPCO	0.053941	0.885276	0.286270	0.054713	0.275463	0

TABLE III. COMPARISON OF BEST RESULTS

Test function	Optimum value	Koziel and Michalewiez[7]	Runarsson and Yao	Deb[8]	Farmani and Wright 2003[9]	Venkatraman and Yen[10]	SAPCO	Proposed algorithm
g01	15.0000	14.7860	15.0000	15.0000	15.0000	14.9999	15.0000	**-15.000**
g02	0.803619	0.799530	0.803515	NA	0.802970	0.803190	0.803202	**0.803607**
g03	1.0000	0.9997	1.0000	NA	1.0000	1.0000	1.0000	**1.000**
g04	30665.539	-30664.900	-30665.539	-30665.537	-30665.500	-30665.531	-30665.401	-30664.03
g05	5126.498	NA	5126.497	NA	5126.989	5126.630	5126.907	5126.585
g06	-6961.814	-6952.100	-6961.814	NA	-6961.800	-6961.179	-6961.046	-6962.071
g07	24.306	24.620	24.307	24.373	24.480	24.411	24.838	24.353
g08	0.095825	0.095825	0.095825	NA	0.095825	0.095825	0.095825	**0.095825**
g09	680.630	680.910	680.630	680.630	680.640	680.762	680.773	680.694
g10	7049.331	7147.900	7054.316	7060.220	7061.340	7060.553	7069.981	7331.455
g11	0.750	0.750	0.750	NA	0.75	0.749	0.749	**0.750**
g12	1.000000	NA	1.000000	NA	NA	NA	1.000000	0.736304
g13	0.053950	NA	0.053957	NA	NA	NA	0.053941	0.096277

V. CONCLUSION

In this paper we propose a new kind of handling constraint method to constraint optimization problems. The transformation of single objective optimization problems to multi-objective optimization problems is a very useful constraint handling technique. The challenges in constrained optimization problems can be solved by the velocity of convergent to m of the first object. In this way we don't have to tune the ratio of feasible and infeasible. All we have to do is to set a proper parameter that can control the velocity. The performance of our algorithm was tested on 13 benchmark functions, and the algorithm was able to find competitive results with all well-regarded algorithms in literature.

ACKNOWLEDGEMENT

This work was jointly supported by the Natural Science Foundation of China (60974077), the Natural Science Foundation of Guangdong Province (07001797, 8151009001000044).

REFERENCES

[1] Hai-lin Liu, Xueqiang Li, "The multiobjective evolutionary algorithm based on determined weight and sub-regional search", cec2009.

[2] Yonas Gebre Woldesenbet, Gary G. Yen, "Constraint Handling in Multiobjective Evolutionary Optimization", IEEE Trans. on Evolutionary Computation, vol. 13, pp.514-525,no. 3, June 2009.

[3] Biruk Tessema, Gary G. Yen, "A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization", 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006.

[4] Guillermo Leguizamón, Carlos A. Coello Coello," Boundary Search for Constrained Numerical Optimization Problems With an Algorithm Inspired by the Ant Colony Metaphor", IEEE transactions on evolutionary computation, vol. 13, no. 2,pp.350-368, April 2009.

[5] T.P. Runarsson and X. Yao, "Stochastic ranking for constraint evolutionary optimization," IEEE Transaction on Evolutionary Computation, vol. 4, pp. 344-354, 2000.

[6] J. J. Liang, Thomas Philip Runarsson, Efren Mezura-Montes Maurice Clerc, P. N. Suganthan1, Carlos A. Coello Coello, K. Deb, Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization, September 18,2006.

[7] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homorphous mappings, and constrained parameter optimization," Evolutionary Computation, vol. 7, pp. 19-44, 1999.

[8] K. Deb, "An efficient constraint handling methods for genetic algorithms," Computer Methods in Applied Mechanics and Engineering, vol. 186, pp. 311-338, 2000.

[9] R. Farmani and J. Wright, "Self-adaptive fitness formulation for constrained optimization," IEEE Transaction on Evolutionary Computation, vol. 7, no. 5, pp. 445-455, 2003.

[10] S. Venkatraman and G.G. Yen, "A generic framework for constrained optimization using genetic algorithms," IEEE Transaction on Evolutionary Computation, vol. 9, no. 4, pp. 424-435, 2005.

[11] Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. Computer Methods in Applied Mechanics and Engineering, 186(2/4):311–338, 2000.