A Robust Algorithm for Solving Nonlinear Programming Problems

Yan Li
Computation Center,
Wuhan University, Wuhan, 430072, China
llyyan2000@21cn.com

Li-Shan Kang
State Key Laboratory of Software Engineering,
Wuhan University, Wuhan, 430072, China

Abstract

In this paper, we introduce a new algorithm for solving nonlinear programming (NLP) problems. It is an extension of Guo's algorithm[1] which possesses enhanced capabilities for solving NLP problems. These capabilities include: a) advancing the variable subspace, b) adding a search process over subspaces and normalized constraints, c) using an adaptive penalty function, and d) adding the ability to deal with integer NLP problems, 0-1 NLP problems, and mixed-integer NLP problems which have equality constraints. These four enhancements increase the capabilities of the algorithm to solve nonlinear programming problems in a more robust and universal way. This paper will present results of numerical experiments which show that the new algorithm is not only more robust and universal than its competitors, but also its performance level is higher than any others in the literature.

1 INTRODUCTION TO GUO'S ALGORITHM

It was shown in [1] that Guo's algorithm has many advantages when solving optimization problems. Since the new algorithm presented in this paper is based upon Guo's algorithm, it is advisable to first present briefly its main features.

The general NLP problem can be expressed in the following form:

$$\begin{aligned} & Minimize \quad f(X,Y) \\ s.t. \quad & h_i(X,Y)=0, \quad i=1,2,\cdots,k_1 \\ & g_j(X,Y)\le 0, \; j=1,2,\cdots,k_2 \\ & X^{lower} \le X \le X^{upper} \\ & Y^{lower} \le Y \le Y^{upper} \end{aligned}$$

where $X \in R^p$, $Y \in N^q$, and the objective function $f(X,Y)$, the equality constraints $h_i(X,Y)$ and the inequality constraints $g_j(X,Y)$ are usually nonlinear functions which include both real and integer variables.

Denoting the domain

$$D=\{(X,Y) \mid X^{lower} \le X \le X^{upper}, Y^{lower} \le Y \le Y^{upper}\},$$

we introduce the concept of a subspace V of the domain D. m points $(X_j,Y_j),\ j=1,2,\cdots,m$ in D are used to construct the subspace V, defined as -

$$V=\{(X_v,Y_v)\in D \mid (X_v,Y_v)=\sum_{i=1}^{m} a_i (X_i,Y_i)\}$$

where a_i is subject to $\sum_{i=1}^{m} a_i = 1, -0.5 \le a_i \le 1.5$.

Because Guo's algorithm deals mainly with optimization problems which have real variables and INequality constraints, we assume $k_1 = 0$ and $q = 0$ in the expression (1).

Denoting $w_i(X)=\begin{cases} 0, & g_i(X)\le 0 \\ g_i(X), & \text{otherwise} \end{cases}$

and $W(X)=\sum_{i=1}^{k_2} w_i(X)$

We define a Boolean function *"better"* as :

$$better(X_1,X_2)=\begin{cases} W(X_1)\le W(X_2) & \text{TRUE} \\ W(X_1) > W(X_2) & \text{FALSE} \\ (W(X_1)=W(X_2))\wedge(f(X_1)\le f(X_2)) & \text{TRUE} \\ (W(X_1)=W(X_2))\wedge(f(X_1)> f(X_2)) & \text{FALSE} \end{cases}$$

If *better* (X_1, X_2) is TRUE, this means that the individual X_1 is "better" than the individual X_2.

Guo's algorithm can be described as follows:

Guo's Algorithm

Begin

initialize *popln* $P=\{X_1, X_2, \ldots, X_N\}$; $X_i \in D$

/ since (q = 0 implies no integer variables)*/*

generation count $t := 0$;

$X_{best} = \arg \min_{1\le i\le N} f(X_i)$;

$X_{worst} = \arg \max_{1\le i\le N} f(X_i)$;

while $abs(f(X_{best}) - f(X_{worst})) > e$ **do**

select randomly m points $X_1', X_2', \ldots, X_m'$ from P to form the subspace V;

select randomly one point X' from V;

If $better(X', X_{worst})$ **then** $X_{worst} := X'$;

$t := t + 1$;

$X_{best} = \arg \min_{1\le i\le N} f(X_i)$;

$X_{worst} = \arg \max_{1\le i\le N} f(X_i)$;

end do

output t, P;

end

where N is the size of population P, $(m-1)$ is the dimension of the subspace V (if the m points(vectors) that construct the subspace V are linearly independent), t is the number of generations, e is the accuracy of solution. X_{best} = arg $\underset{1\le i\le N}{Min} f(X_i)$ means that X_{best} is the variable (individual) in X_i (i=1, 2,…, N) that makes the function $f(X)$ have the smallest value.

2 A NEW OPTIMIZATION ALGORITHM

Since Guo's algorithm deals mainly with continuous NLP problems with INequality constraints, to make it a truly universal and robust algorithm for solving general NLP problems, we extend Guo's algorithm by adding to it the following 8 improvements -

(1) Guo selected randomly only one candidate solution from the current subspace V. This action would tend to ignore better solutions in the subspace, and hence influence negatively the quality of the result and the efficiency of the search. If however, we select randomly several individuals from the subspace, and substitute the best one for the worst one in the current population, the search should be better. So we replace the instruction line in Guo's algorithm:

"select randomly one point X' from V; "

with the two instruction lines:

" select randomly s points $X_1^*, X_2^*, \ldots, X_s^*$ from V;

$$X' = \arg \underset{1\le i\le s}{Min} f(X_i^*);"$$

(2) The dimension m of the subspace in Guo's algorithm is fixed (i.e. m parents reproduce). Thus, when the population is close to the optimal value, the searching range is still large. This would apparently result in unnecessary computation, and effect the efficiency of the search. We can in fact reduce the dimension of the subspaces. We therefore use subspaces with variable dimensions in the new algorithm, by adding the following instruction line to Guo's algorithm:

if $abs(f(X_{best}) - f(X_{worst})) \le h$ *.and.* $m \ge 3$ **then**

$$m := m - 1;$$

where h depends on the computation accuracy e, and $h > e$. For example, if the computation accuracy e = 10^{-14}, then we can set $h = 10^{-2}$ or 10^{-3}.

(3) We know in principle that Guo's algorithm can deal with problems containing EQuality constraints using the device of setting two INequality constraints $0 \le h_i(X,Y)$ and $h_i(X,Y) \le 0$ to replace the equality constraint $h_i(X,Y) = 0$, but the experimental results when employing this device are not ideal. One such method is to use a penalty function. Since the INequality constraints have been included in function *better* in Guo's algorithm, we use the following fitness function which includes only the penalty of the EQuality constraints:

$$F(x) = f(x) + r \sum_{i=1}^{k_1} (h_i(x))^2$$

(4) Using the penalty function described above in numerical experiments, we observe that when a constraint coefficient is very large, it will have a decisive effect on the entire fitness function and influence the accuracy of the solution (referring to Example 1). We therefore introduce a normalization method (see [2]), which finds the largest coefficient in the constraint, then divides the entire constraint by it. For example, the third constraint in Example.1 is:

$$750 * 1728 - px_1^2 x_2 - \frac{4}{3} px_1^3 \le 0$$

The largest coefficient is 750*1728=1,296,000. Divided by 1,296,000, the constraint becomes

$$1 - (px_1^2 x_2 - \frac{4}{3} px_1^3) / 1296000 \le 0$$

The effect of the constraint on the fitness function will now be reduced significantly. The normalized constraint is denoted as $\bar{h}_i(x)$, so $F(x)$ is:

$$F(x) = f(x) + r \sum_{i=1}^{k_1} (\bar{h}_i(x))^2$$

(5) The penalty factor r is usually fixed. We make r a variable namely $r = r(t)$, where t is the iteration count. It can self-adjust according to the reflection information, so we label it a "self-adaptive penalty operator". Since the constraints have been normalized, r is relative only to the range of the objective function, which ensures a balance between the errors of the fitness function and the objective function, in order of magnitude.

(6) Guo's algorithm can deal only with continuous optimization problems. It cannot deal directly with integer or mixed integer NLP problems. In our algorithm, when we are confronted with such problems, we need only replace the integer variables derived from the range of the float of the fitness function with "*integer function*" int(Y), where int(Y) is defined as the integer part of Y. No other changes to the algorithm are needed.

(7) 0-1 NLP problems are a special case of integer NLP problems. The procedure for solving them is similar to (6) except that the 0-1 variables should be defined in the interval

$[0,2)$.

(8)The only genetic operator used in Guo's algorithm was crossover. However, we can add mutation if we know more about the characteristics of the problem. For example, if we know that the minimum is relative to the inner sequence of the individual, we can introduce a mutation operator which sorts the components of each individual in descending order. Some experiments show that the results are better with sorting than without sorting.

Considering the above points, we introduce a new algorithm as follows:

Denoting $Z=(X, Y^*)$, where $Z\in D^*$, and $D^*=\{(X,Y^*)\mid X^{lower}\le X\le X^{upper}, Y^{lower}\le Y^*\le Y^{upper}\}$, where $X\in R^p, Y^*\in R^q$, we define the fitness function as:

$$F(Z)=f(X,\mathrm{int}(Y^*))+r(t)\sum_{i=1}^{k_1}(\bar{h}_i(X,\mathrm{int}(Y^*)))^2$$

Denoting $w_i(Z)=\begin{cases}0, & g_i(X,\mathrm{int}(Y))\le 0\\ g_i(X,\mathrm{int}(Y)), & \text{otherwise}\end{cases}$

and $W(Z)=\sum_{i=1}^{k_2}w_i(Z)$,

We define the Boolean function "*better*" as follows:

$$better(Z_1,Z_2)=\begin{cases}W(Z_1)\le W(Z_2) & \text{TRUE}\\ W(Z_1)>W(Z_2) & \text{FALSE}\\ (W(Z_1)=W(Z_2))\wedge(F(Z_1)\le F(Z_2)) & \text{TRUE}\\ (W(Z_1)=W(Z_2))\wedge(F(Z_1)>F(Z_2)) & \text{FALSE}\end{cases}$$

The new algorithm can now be described -

New Algorithm

Begin

initialize $P=\{Z_1,Z_2,\ldots,Z_N\}$; $Z_i\in D^*$;

$t:=0$;

$Z_{best}=\arg\underset{1\le i\le N}{Min}F(Z_i)$;

$Z_{worst}=\arg\underset{1\le i\le N}{Max}F(Z_i)$;

while $abs(f(Z_{best})-f(Z_{worst}))>e$ **do**

select randomly m points $Z_1', Z_2', \ldots, Z_m'$ from P to form the subspace V;

select s points *randomly* $Z_1^*, Z_2^* \ldots Z_s^*$ from V;

$Z'=\arg\underset{1\le i\le s}{Min}F(Z_i^*)$;

If $better(Z', Z_{worst})$ **then** $Z_{worst}:=Z'$;

$t:=t+1$;

$Z_{best}=\arg\underset{1\le i\le N}{Min}F(Z_i)$;

$Z_{worst}=\arg\underset{1\le i\le N}{Max}F(Z_i)$;

if $abs(f(Z_{best})-f(Z_{worst}))\le h$ *.and.* $m\ge 3$ **then**

$m:=m-1$;

endwhile

output t, P;

end

The new algorithm has the two important features:

a) The ergodicity of the search. During the random search of the subspace, we employ a "non-convex combination" approach, that is, the coefficients a_i of $Z'=\sum_{i=1}^{m}a_iZ_i'$ are random numbers in the interval [-0.5,1.5], which ensures a non-zero probability that any point in the solution space is searched. This ergodicity of the algorithm ensures that the optimum is not ignored.

b) The monotonic fitness decrease of the population (when the minimum is required). Each iteration $(t\to t+1)$ of the algorithm discards only the individual having the worst fitness in the population. This ensures a monotonically decreasing trend of the fitness of the population, which ensures that each individual of the population will reach the optimum.

When we consider the population P(0), P(1), P(2),…, P(t),… as a Markov chain, we can prove the convergence of our new algorithm. See [4].

3 NUMERICAL EXPERIMENT

Example : Pressure Vessel Design Problem.

In this problem, a cylindrical pressure vessel with two hemispherical heads (fig.1) is designed to minimize its fabrication cost. The total cost consists of the cost of the materials plus the cost of forming and welding. Four variables are used. They are the inner radius o f the vessel x_1, the length of the vessel without the heads x_2, the thickness of pressure vessel 0.0625 y_1, and the thickness of the head 0.0625 y_2. Of the four variables, x_1 and x_2 are continuous variables and y_1 and y_2 are discrete variables. Denoting the variable vector (X, Y) =(x_1, x_2, y_1, y_2), the NLP problem can be described as follows:

$$Minimize\quad f(X,Y)=0.6224(0.0625y_1)(0.0625y_2)x_1+1.778(0.0625y_2)x_1^2+3.1661(0.0625y_1)^2x_2+19.84(0.0625y_1)^2x_1$$

$$s.t.\begin{cases}g_1(X,Y)=0.0193\,x_1-0.0625\,y_1\le 0\\ g_2(X,Y)=0.00954\,x_1-0.0625\,y_2\le 0\\ g_3(X,Y)=750*1728-\pi x_1^2x_2-4/3*\pi x_1^3\le 0\\ g_4(X,Y)=x_2-240\le 0\end{cases}$$

The first and second constraints limit the thickness of the

cylinder and the head to lie below a factor of the cylinder diameter. The third constraint ensures that the enclosed volume is greater than a specified value. The fourth constraint limits the length of the cylinder to a prescribed value. The third constraint is the most sensitive.

We performed many numerical experiments using the new algorithm. When we set the parameters to: N =30, e =10^{-14}, h =10^{-3}, m =10, s = 8, r (t) = 100000+int (t/1000), we obtained the best result ever obtained in the literature, as listed in Table 1.

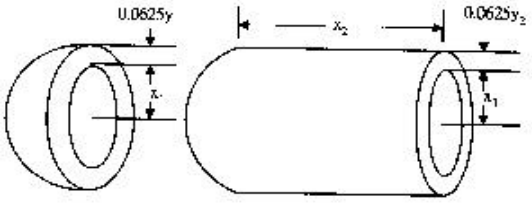

Fig.1 The Pressure Vessel Model

Many people have tested this problem. To illustrate the performance of the new algorithm, we compare our result with other published results, as listed in Table 1.

4 CONCLUSION

Judging by the results obtained from the above numerical experiments, we conclude that our new algorithm is both universal and robust. It can be used to solve function optimization problems with complex constraints, such as NLP problems with inequality and (or) equality constraints, or without constraints. It can solve 0-1 NLP problems, integer NLP problems and mixed integer NLP problems. When confronted with different types of problems, we don't need to change our algorithm. All that is needed is to input the fitness function, the constraint expressions, and the upper and lower limits of the variables of the problem. Our algorithm usually finds the global optimal value.

References

[1] Guo Tao, Kang Lishan. A new evolutionary algorithm for function optimization. Wuhan University Journal of Nature Science. Vol. 4 No.4 1999,409~414

[2] Kalyanmoy Deb. GeneAS: A robust optimal design technique for mechanical component design. Evolutionary algorithm in engineering application: Springer-Verlag,1997, 497~514

[3] Carlos A. Coello: Self-adaptive penalties for GA-based optimization, in Proceedings of the Congress on Evolutionary Computation, Washington, D.C USA, IEEE Press, 1999,537~580

[4] Jun He, Lishan Kang. On the convergence rates of genetic algorithms. Theoretical Computer Science, 229 (1999) 23~29

[5] Sandgren, E: Nonliner integer and discrete programming in mechanical design, ASME J. Mechanical Design, 1990,112: 223~229

[6] J.F.Fu,R.G.Fenton and W.L.Cleghorn: A mixed integer-discrete-continuous programming method and its application to engineering design optimization. Engineering Optimization, 1991,17(3):263~280

[7] B.K.Kannan and S.N.Kramer: An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. Journal of Mechanical Design. Transactions of the ASME, 1994,116:318~320

[8] Y.J.Cao and Q.H.Wu: Mechanical design optimization by mixed-variable evolutionary programming, in proceedings of the 1997 International Conference on Evolutionary Computation, Indianapolis,Indiana, 1997,443~446

[9] Yung-Chien Lin :A hybrid method of evolutionary algorithms for mixed-integer nonlinear optimization problem, in Proceedings of Congress on Evolutionary Computation, 1999 (3): 2159~2166

Design variables	Sandgren [5]	Fu [6]	Kannan [7]	Cao [8]	Deb [2]	Lin [9]	Carlos [3]	this paper
x_1	47.700	48.381	58.291	51.1958	48.3290	48.5765	40.3239	38.8601
x_2	117.701	111.74	43.690	90.7821	112.679	110.056	200.000	221.365
y_1	20	18	20	16	15	15	13	12
y_2	10	10	10	10	8	8	7	6
g_1(X,Y)	-0.204	-0.1913	-0.000	-0.0119	-0.0048	0.0000	-0.0343	-0.0000
g_2(X,Y)	-0.170	-0.1634	-0.069	-0.1366	-0.0389	-0.0336	-0.0528	-0.0043
g_3(X,Y)	-6.3×10^{6}	-75.875	-21.216	-13584	-3652.9	0.0000	-27.106	-0.0000
g_4(X,Y)	-122.299	-128.26	-196.23	-149.22	-127.32	-129.94	-40.000	-18.635
f (X,Y)	8129.80	8084.62	7198.20	7108.62	6410.38	6370.70	6288.74	5850.38

Table 1 Comparative Results with the Pressure Vessel Problem