Computers and Chemical Engineering 34 (2010) 36–41

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Short communication

A hybrid genetic algorithm for twice continuously differentiable NLP problems

Quan Yuan, Feng Qian*

Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, Shanghai, 200237, PR China

ARTICLE INFO

Article history:
Received 23 September 2008
Received in revised form 28 August 2009
Accepted 19 September 2009
Available online 2 October 2009

Keywords:
Hybrid genetic algorithm
Twice continuously differentiable NLP
Local search
Global optimization

ABSTRACT

This paper introduces a new hybrid genetic algorithm to solve twice continuously differentiable nonlinear programming (NLP) problems. The algorithm combines the genetic algorithm with local solver differently from some hybrid genetic algorithms. Numerical experiments show the new hybrid genetic algorithm can obtain results better than the known ones reported by the literatures.

1. Introduction

In this paper, we focus on a hybrid genetic algorithm (HGA) for twice continuously differentiable non-linear programming (TCDNLP) problems, which can be formulated as:

$$\begin{aligned}\min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) \le 0 \\ & \boldsymbol{h}(\boldsymbol{x}) = 0 \\ & \boldsymbol{x} \in [\boldsymbol{x}^L, \quad \boldsymbol{x}^U]\end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is a vector of n continuous variables, $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{g}(\boldsymbol{x})$ is a vector of inequality constraints, and $\boldsymbol{h}(\boldsymbol{x})$ is a vector of equality constraints. $\boldsymbol{x}^L$ and $\boldsymbol{x}^U$ denote the lower and upper bounds on $\boldsymbol{x}$ respectively. All the functions are twice continuously differentiable. Many practical problems in the fields of engineering, science, finance and economics can be formulated as a TCDNLP problem. Specific problems include phase and chemical equilibrium characterization, distillation sequencing, reactor network design, batch process design, protein folding and portfolio optimization.

Many local optimization techniques are available to solve this type of problem. The most popular are sequential quadratic programming (SQP) (Alkaya, Vasantharajan, & Biegler, 1999; Biegler, Nocedal, Schmid, & Ternet, 2000; Biegler, Schmid, & Ternet (1997); Boggs & Tolle, 1995; Cervantes, Waechter, Tutuncu, & Biegler, 2000; Edgar & Himmelblau, 1988), reduced gradient techniques (Sarma & Reklaitis, 1982; Reklaitis, Ravindran, & Ragsdell, 1983, etc.) and interior algorithms (Byrd, Gilbert, & Nocedal, 2000; Coleman, & Li, 1996; Waltz, Morales, Nocedal, & Orban, 2006). Commercial implementations of algorithms based on these approaches have been available for some years. When the objective function and the inequality constraints are convex and the equality constraints are linear, there is a unique solution which can be identified with a local solver. But in many practical applications, TCDNLP problems have several local solutions because of the non-convexity of the objective and/or feasible region. And using these local method only may be sensitive to the choice of the initial guess and may lead to a convergence failure (Schittkowski & Monma, 1986).

To overcome these drawbacks, the global optimization methods of these problems has attracted growing attention in the last three decades, either through deterministic methods, such as primal-dual methods (Floudas & Visweswaran, 1993), interval methods (Casado, Martinez, Garcia, & Sergeyev, 2003; Shen, 1999), Branch-and-bound methods (Adjiman, Dallwig, Floudas, & Neumaier, 1998; Adjiman, Androulakis, & Floudas, 1998), and filled functions method (Ge & Qin, 1990), or stochastic methods, include taboo search (TS) (Glover & de Werra, 1993), simulated annealing (SA) (Kirkpatrick, Gellat, & Vecci, 1983) and genetic algorithm (GA) (Goldberg, 1989; Michalewicz, 1995).

Hybrid stochastic optimization methods also occupy an important space in global optimization. These methods combine stochastic and deterministic or various stochastic optimization algorithms in order to improve the search efficiency. A large number of papers about such algorithms have been reported, including SQP, SA, TS, GA, evolutionary algorithms (EA), clustering algorithms (CA), and adaptive random search (ARS). For example, the combi-

* Corresponding author at: Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, YIfu Building 908, Meilong Road 130, Shanghai, 200237, PR China. Tel.: +86 21 64252056; fax: +86 21 64252056.
E-mail addresses: bjyuanquan@gmail.com (Q. Yuan), fqian@ecust.edu.cn (F. Qian).

0098-1354/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2009.09.006

Nomenclature

Abbreviations

NLP	non-linear programming
TCDNLP	twice continuously differentiable non-linear programming
HGA	hybrid genetic algorithm

Symbols

$\boldsymbol{x}$	a vector in (1)
n	the number of variable in $\boldsymbol{x}$
$f(\boldsymbol{x})$	objective function in (1)
$\boldsymbol{g}(\boldsymbol{x})$	a vector of inequality constraints in (1)
$\boldsymbol{h}(\boldsymbol{x})$	a vector of equality constraints in (1)
$\boldsymbol{x}^L$	lower bounds on $\boldsymbol{x}$
$\boldsymbol{x}^U$	upper bounds on $\boldsymbol{x}$
num	population size
X, X'	population and intermediate population after selection step
x_i, x'_i	ith individual of population and intermediate population after selection step
$x_i^\#$	ith local optimum with respect to x_i
fitness(x_i)	the fitness with respect to x_i
t	the number of generation
$f_{\min}$	the minimum of $f(x_i^\#)$ in one generation
$\Pr(i)$	selection probability defined by (3)
$\text{ac}-\Pr(i)$	accumulative probability defined by (4)
M	a random number in (6)
$\boldsymbol{P}$	a direction vector in (6)
T_{gen}	a given integer defines the maximum generation
m	the number of niches in niche GA

Greek symbols

η	a random number in (0, 1)
α	a random number in (0, 1)
σ	the radius of niche in niche GA

nations of EA and GA (Pham, 1995), GA and SA (Mohanta, Sadhu, & Chakrabarti, 2007; Sadegheih, 2007; Yu, Fang, Yao, & Yuan, 2000), GA/SA/TS (Liu, Ma, & Zhang, 2002), centroid ARS coupled with CA and SA (Maria, 2004), and GA with SQP (Jang, Hahn, & Hal, 2005; Mansoornejad, Mostoufi, & Jalali-Farahani, 2008) have been applied for solving optimization problems.

In this work we present a hybrid genetic algorithm (HGA) combined with genetic algorithm and local solver technique to solve (1). The basic features and details of the HGA are covered in Section 2. Numerical examples are provided in Section 3. Section 4 gives final conclusions.

2. The hybrid genetic algorithm

2.1. General description of genetic algorithms

Genetic algorithms (GAs) are stochastic techniques whose search methods model a natural evolution. At the first generation (i.e., iteration in mathematical meaning), a GA initializes a population of randomly created individuals. Each individual in the population is encoded into a string or chromosome or some float number which represents a possible solution to a given problem. The fitness of an individual is evaluated with respect to a given objective function. Highly fit individuals or solutions are given opportunities to reproduce by exchanging some of their genetic information, in a crossover procedure, with other highly fit individuals. This produces new "offspring" solutions (i.e., children), which share some characteristics taken from both parents. Mutation is often applied after crossover by altering some genes or perturbing float numbers. The offspring can either replace the whole population or replace less fit individuals. This evaluation–selection–reproduction cycle is repeated until a satisfactory solution is found.

The basic steps of the procedure can be shown as:

```
Generate an initial population;
repeat:
Crossover and mutate individuals to produce children;
Evaluate fitness of the children;
Select the population from the children;
until a satisfactory solution has been found.
```

Although the GAs applicability to constrained problems is still actively debated, they have proven capabilities as function optimizers.

2.2. Constraint handling methods of GAs and our idea

When TCDNLPs are solved by GAs, the method to deal with constraints is a crucial point. A huge number of papers have devoted to it. A brief cluster we listed here is:

- Methods based on preserving feasibility of solutions (Goldberg, 1989; Michalewicz, 1992).
- Methods based on decoders (Michalewicz, 1992).
- Methods based on penalty functions (Deb, 2000; Gen & Cheng, 1996; Homaifar, Qi, & Lai, 1994; Joines & Houck, 1994; Michalewicz, 1992; Smith & Tate, 1993).
- Methods based on Multi-objective optimization (Coello Coello, & Christiansen, 2000; Sarma, & Reklaitis, 1982; Summanwar et al., 2002).
- Hybrid methods (GAs are coupled with another technique, normally classical constrained search methods, see Michalewicz & Schoenauer, 1996).

A very comprehensive survey of the popular constraint handling techniques used with GAs is given by Coello Coello (2002).

Besides these methods mentioned above, we describe our idea as follows. The infeasible solutions may be brought out in initial population or be generated by crossover/mutation step. Since some local solver does not require the feasibility of initial guesses, we can do local search to each individual at first. This process can be seen as a repair algorithm. So an infeasible individual may be feasible and a feasible individual may be improved. Meanwhile, the fitness function now is evaluated by the local optimum obtained by the local search instead of being evaluated by the target function originally.

After the local search and evaluation, two individuals selected from GA is in one of two cases:

(i) They have the same local optimum.
(ii) They have different local optimum.

For case (i), we think the crossover step is useless and let the two individuals mutate synchronously. For case (ii), we let them crossover. This process can keep the diversification of individual and prevent the "premature" phenomenon in GA.

The flow chart of our algorithm is the right one in Fig. 1.

2.3. Characteristic of the algorithm

2.3.1. Difference between our algorithm and general hybrid GAs

Many hybrid genetic algorithms embedded local search algorithm in GA as an auxiliary. We call these hybrid genetic algorithms general hybrid GAs in this paper. The flow charts of the two

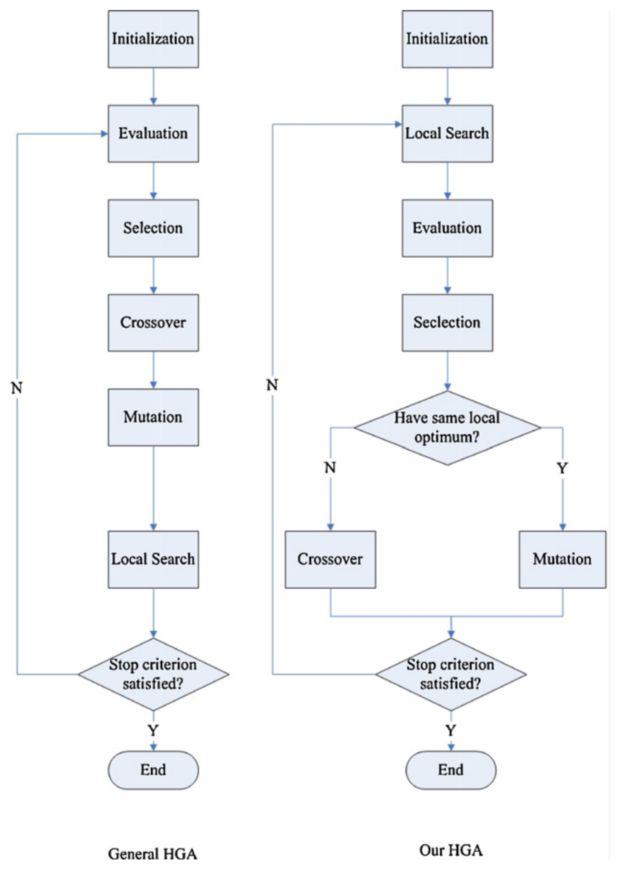

Fig. 1. Flow charts of general HGA and our HGA.

algorithms are shown in Fig. 1. The difference between our algorithm and general hybrid GAs lies in the following aspects:

- General hybrid GAs do local search *after* evaluation, selection, crossover and mutation but our algorithm does it *before* evaluation.
- In general hybrid GAs, both the fitness and the genetic code of an individual are changed after local search. While our algorithm only changes the fitness of an individual without altering the individual itself.
- General hybrid GAs execute crossover and mutation steps respectively. So the probabilities of crossover and mutation need considering. In our algorithm, the execution of crossover or mutation relies on that if two individuals have same local optimum. So the probabilities of crossover and mutation are unnecessary.

2.3.2. *Difference between our algorithm and niche GAs*

Niche GAs are designed for solving multi-modal optimization problems. In niche GAs, the subspaces around some modal values in solution space are compared to niches of biological growth. And the individuals around modal values are compared to species breeding in the niches. Niche GAs use the idea of "fitness sharing". That is, the fitness of all individuals in one niche will be degenerated proportionally to the niche's scale. It means that if some niche concludes more individuals, all fitness values of individuals in this niche will be degenerated in order to stimulate individuals in other niches to multiply.

When niche GAs is implemented, some information of solution must be known prior. The information may include parameter of the radius (σ) or number (m) of modal values. This information is identified with the radius or number of niches in niche GAs. Some papers discuss niche GAs include (but not limited):

- Use σ only. Such as Goldberg and Richardson (1987) and Petrowski (1996).
- Use m only. Such as Yin and Germay (1993).
- Use both σ and m. Such as Miller and Shaw (1996) and Goldberg and Wang (1997).

Our algorithm can be regarded as a kind of niche GA but the niche scale is defined by local search. It needs no information known in advance.

2.3.3. *Advantages and disadvantages*

Our HGA has several advantages. First, putting local search before evaluation can eliminate the necessity of a penalization of infeasible solutions and/or special crossover and mutation operators. Second, it follows from Fig. 1 that the crossover and mutation steps are executed not respectively but relying on that if two individuals have the same optimum in our HGA. So the parameters of crossover and mutation's probability are controlled by algorithm automatically. Last, we need not to find an exact global optimum in the computation. The HGA will find the global optimum successfully only if the points obtained from GA 'drop' into the neighborhood of the global optimum.

The disadvantages of our HGA lie in two ways. First, due to the application of the local search on every individual in every generation, the cost of computation is cumbersome. Second, the population generated by our HGA is often divergent. So some stop criterion based on convergence cannot be used in our algorithm.

To compensate these drawbacks, we take small population size and an integer is taken as the maximum number of generations. *Elitist strategy*, i.e., optimum in every generation is recorded, is used. From our experiments, small population size (such as 10) and moderate maximum number of generations (such 50 or 100) can guarantee a good performance of our algorithm.

2.4. *Details of the algorithm*

2.4.1. *Representation, local search and fitness evaluation*

The algorithm starts with initialization of the first generation. It creates a population $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{num}\}$ with *num* individuals which encoded into *real float number*, randomly from box constraints $[\boldsymbol{x}^L, \quad \boldsymbol{x}^U]$ in (1). Then to every individual, a local search is applied in order to find a local optimum. In this paper we use a local solver implemented by KNITRO $^{\text{TM}}$ (see Waltz & Plantenga, 2008). If the local solver cannot find a local optimum to some individual, we discard it, regenerate a new individual and repeat local search. After local search, we obtain local optimum solution $\boldsymbol{x}_1^{\#}, \ldots, \boldsymbol{x}_{num}^{\#}$ and local optimum value $f(\boldsymbol{x}_1^{\#}), \ldots, f(\boldsymbol{x}_{num}^{\#})$.

Following Handbook of Evolutionary Computation (1997), the fitness function of ith individual is defined as

$$\text{fitness}(\boldsymbol{x}_i(t)) = \frac{1}{1 + f(\boldsymbol{x}_i^{\#}(t)) - f_{\min}(t)}, \quad \boldsymbol{x}_i(t) \in [\boldsymbol{x}^L, \boldsymbol{x}^U], \tag{2}$$

where $f_{\min} = \min\{f(\boldsymbol{x}_1^{\#}(t)), \ldots, f(\boldsymbol{x}_{num}^{\#}(t))\}$ in generation t.

2.4.2. *Selection*

Roulette-wheel selection is used as a proportionate selection scheme in our algorithm. In this scheme, the ith individual is given a selection probability:

$$\Pr(i) = \frac{\text{fitness}(\boldsymbol{x}_i)}{\sum_{i=1}^{num} \text{fitness}(\boldsymbol{x}_i)} \tag{3}$$

where *num* is the size of the population. So we construct a roulette-wheel with *num* slots. The each slot is as wide as the probability

Table 1
Comparison of several methods for Problem 1 (N/A: not available) (PART I).

Results	GRG	MGA	Static penalty	Co-evolutionary penalty	Gen
Best	−30373.949	−31005.7966	−30790.27159	−31020.859	−30183.576
Mean	N/A	−30862.8735	−30446.4618	−30984.2407	N/A
S.D.	N/A	73.240	226.3428	73.6335	N/A

for selection. Then we compute accumulative probability to the *i*th individual:

$$\mathrm{ac}-\Pr(i)=\sum_{j=1}^{i}\Pr(j). \tag{4}$$

Then we rotate the roulette-wheel *num* times to select *num* individuals for the next step. In each rotation a random number $\eta\in(0,1)$ is generated. The *i* th ($i>1$) individual is selected if $\mathrm{ac}-\Pr(i-1)\le\eta<\mathrm{ac}-\Pr(i)$ (if $0<\eta<\mathrm{ac}-\Pr(1)$ the first individual is selected). So after selection we obtain an intermediate population $X'(t)=\{\boldsymbol{x}'_1(t),\ldots,\boldsymbol{x}'_{num}(t)\}$ in generation *t*.

2.4.3. Crossover/mutation

The crossover/mutation step is implemented as follows:

If $\boldsymbol{x}'_i(t)$ and $\boldsymbol{x}'_j(t)$ in X' have different local optimum solutions, we cross the two individuals as

$$\begin{aligned}\boldsymbol{x}_i(t+1)&=\alpha\boldsymbol{x}'_i(t)+(1-\alpha)\boldsymbol{x}'_j(t)\\ \boldsymbol{x}_j(t+1)&=\alpha\boldsymbol{x}'_j(t)+(1-\alpha)\boldsymbol{x}'_i(t).\end{aligned} \tag{5}$$

where $\alpha\in(0,1)$ is a random number.

If the two individuals have same local optimum solution, we mutate them as

$$\begin{aligned}\boldsymbol{x}_i(t+1)&=\boldsymbol{x}'_i(t)+M\cdot\boldsymbol{P}\\ \boldsymbol{x}_j(t+1)&=\boldsymbol{x}'_j(t)-M\cdot\boldsymbol{P}.\end{aligned} \tag{6}$$

where M is a random real number, $\boldsymbol{P}=\boldsymbol{x}'_i(t)-\boldsymbol{x}'_j(t)$ is a direction vector.

In actual computation, $\boldsymbol{x}_i(t+1)$ and $\boldsymbol{x}_j(t+1)$ may be out of the box constraints $[\boldsymbol{x}^L,\boldsymbol{x}^U]$. We can reduce the value of M if this happens. If the norm $\|\boldsymbol{P}\|$ is too small, we can regenerate a random direction vector $\boldsymbol{P}$.

2.4.4. Stop criterion and other details

The maximum number of generations is used as the stop criterion, i.e., the algorithm will stop when it reaches the number of maximum generation. Elitist strategy (see the explanation in Section 2.3.3) is used and the best one is accepted as the final solution.

3. Numerical experiments

We tested our HGA in three problems. The population size we take is 10 and the maximum generation number is 50 in all problems. The programs are coded by MATLAB.

3.1. Problem 1

This problem was first proposed and solved by Himmelblau (1972). It has five continuous variables, six non-linear inequality constraints and ten boundary conditions. The problem can be stated as follows:

$$\begin{array}{ll}\min & 5.3578547x_3^2+0.835691x_1x_5+37.293239x_1-40792.141\\ \text{s.t.} & 0\le 85.334407+0.0056858x_2x_5+0.000026x_1x_4-0.0022053x_3x_5\le 92\\ & 90\le 80.51249+0.0071317x_2x_5+0.0029955x_1x_2+0.0021813x_3^2\le 110\\ & 20\le 9.300961+0.0047026x_3x_5+0.0012547x_1x_3+0.0019085x_3x_4\le 25\\ & 78\le x_1\le 102\\ & 33\le x_2\le 45\\ & 27\le x_i\le 45,\quad i=3,\ldots,5\end{array}$$

Table 2
Comparison of several methods for Problem 1 (PART II).

Results	Dynamic	Annealing	Adaptive	This paper
Best	−30903.877	−30829.201	−30903.877	−31025.5608
Mean	−30539.9156	−30442.126	−30448.007	−31025.5605
S.D.	200.035	244.619	249.485	1e−4

This problem has been used as a benchmark for GAs. Coello Coello (2002) used different kinds of GA in literatures to solve it. We list the result in his paper and ours in Tables 1 and 2. The methods in Table 1 from the left to right are generalized reduced gradient method (GRG) ((Edgar & Himmelblau, 1988), GA based on multi-objective optimization (MGA) (Coello, 2000a), GA based on static penalty (static penalty) (Homaifar et al., 1994), GA based on co-evolutionary penalty (co-evolutionary penalty) (Coello, 2000b)) and GA based on both local and global reference (Gen) (Gen & Cheng, 1997). The methods in Table 2 from the left to right are GA based on dynamic penalty (dynamic) (Joines & Houck, 1994), GA based on annealing penalty (annealing) (Michalewic & Attia, 1994), GA based on adaptive penalty (adaptive) (Hadj-Alouane & Bean, 1997) and our HGA. The solutions reported in Tables 1 and 2 were produced after performing 30 runs. It can be seen easily from the two tables that our algorithm produces better results than all others do.

3.2. Problem 2

This problem is taken from Deb (2000). It is a welded beam design problem, the practical background is omitted here. It has four design variables and five inequality constraints:

$$\begin{array}{ll}\min & 1.10471x_1^2x_2+0.04811x_3x_4(14.0+x_2)\\ \text{s.t.} & \tau-\tau_{\max}\le 0\\ & \sigma-\sigma_{\max}\le 0\\ & x_1-x_4\le 0\\ & 0.10471x_1^2x_2+0.04811x_3x_4(14.0+x_2)-5.0\le 0\\ & 0.125-x_1\le 0\\ & \delta-\delta_{\max}\le 0\\ \text{where} & \tau=\sqrt{(\tau')^2+2\tau'\tau''\dfrac{x_2}{2R}+(\tau'')^2}\\ & \tau'=\dfrac{P}{\sqrt{2}x_1x_2},\quad \tau''=\dfrac{MR}{J},\quad M=P\left(L+\dfrac{x_2}{2}\right)\\ & R=\sqrt{\dfrac{x_2^2}{4}+\left(\dfrac{x_1+x_3}{2}\right)^2}\\ & J=2\left\{\sqrt{2}x_1x_2\left[\dfrac{x_2^2}{12}+\left(\dfrac{x_1+x_3}{2}\right)^2\right]\right\}\\ & \sigma=\dfrac{6PL}{x_4x_3^2},\quad \delta=\dfrac{4PL^3}{Ex_3^3x_4}\\ & P_c=\dfrac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}\left(1-\dfrac{x_3}{2L}\sqrt{\dfrac{E}{4G}}\right)\\ & P=6000,\quad L=14,\quad E=30\times10^6\\ & G=12\times10^6,\ \tau_{\max}=13,600,\quad \sigma_{\max}=30000,\\ & \delta_{\max}=0.25\\ & 0.1\le x_1\le 2.0,\quad 0.1\le x_2\le 10.0,\quad 0.1\le x_3\le 10.0,\\ & 0.1\le x_4\le 2.0\end{array}$$

Table 3
Comparison of the results of Problem 2.

Design variables	Best solution found		
	This paper	Deb's method	Siddall's method
x_1	0.2057	0.2489	0.2444
x_2	3.4705	6.1730	6.2189
x_3	9.0366	8.1789	8.2915
x_4	0.2057	0.2533	0.2444
Solution	1.7249	2.4331	2.3815

Deb (2000) solved this problem using a simple GA with binary representation and penalty function suggested by Goldberg (1989). The problem has also been solved by Ragsdell and Phillips (1976) using geometric programming. We compare our result with theirs in Table 3. The solution shown for the technique proposed here is the best produced after 30 runs, and the mean from the 30 runs performed is 1.7240 with a standard deviation of 6e−3.

3.3. Problem 3

This problem presented by Michalewicz (1995) can be stated as:

$$\begin{aligned}
\text{min}\quad & x_1+x_2+x_3\\
\text{s.t.}\quad & 1-0.0025(x_4+x_6)\geq 0\\
& 1-0.01(x_8-x_5)\geq 0\\
& x_1x_6-833.33252x_4-100x_1+83333.333\geq 0\\
& x_2x_7-1250x_5-x_2x_4+1250x_4\geq 0\\
& x_3x_8-x_3x_5+2500x_5-1,250,000\geq 0\\
& 100\leq x_1\leq 10,000\\
& 1000\leq x_i\leq 10,000,\quad i=2,3\\
& 10\leq x_i\leq 1000,\quad i=4,\ldots,8
\end{aligned}$$

The problem has eight variables and five inequality constraints (two linear and three non-linear). The reported known solution is 7049.330923. Although the objective function is linear, Michalewicz (1995) experienced that it is difficult to solve. He has obtained 7377.976. Report from Deb (2000), the best solution obtained with niching and mutation at maximum generations 4000 and population size 80 has a function value of 7060.221. BASIC GA of Shopova and Vaklieva-Bancheva (2006) reaches the solution of 7095.485139959999, with the population size 80 and 10,000 generations. Our algorithm obtained a solution 7049.2480, which is a little better than the best known solution. The optimum is obtained in all 30 runs.

4. Conclusions

A new hybrid genetic algorithm is proposed in this paper. The local solver in this algorithm only changes the fitness of an individual without altering the individual itself. Putting local search before evaluation can eliminate the necessity of a penalization of infeasible solutions or special crossover and mutation operators. And the execution of crossover and mutation step does not depend on some determinate probability. It can be applied to a class of global optimization problems for twice continuously differentiable NLP problems. Numerical simulations show that the new hybrid genetic algorithm can obtain results better than the ones reported in literatures. The generalization of the algorithm to other optimization problems, such as mixed integer programming problems, will be studied in further work.

Acknowledgments

The authors are grateful to anonymous reviewers of their valuable comments and suggestions.

F. Qian's work is supported in part by National Science Fund for Distinguished Young Scholars (60625302), National Natural Science Foundation of China (General Program, 60704028), Major State Basic Research Development Program of Shanghai (07JC14016), and 973 Program (2009CB320603).

References

Adjiman, C. S., Androulakis, I. P., & Floudas, C. A. (1998). A global optimization method, αBB, for general twice-differentiable NLPs-II. Implementation and computational results. *Computational Chemical Engineering*, *22*, 1159–1179.

Adjiman, C. S., Dallwig, S., Floudas, C. A., & Neumaier, A. (1998). A global optimization method, αBB, for general twice-differentiable NLPs-I. Theoretical advances. *Computational Chemical Engineering*, *22*, 1137–1158.

Alkaya, D., Vasantharajan, S., & Biegler, L. T. (1999). Successive quadratic programming: Applications in the process industry. In C. Floudas (Ed.), *Encyclopedia of optimization*. Kluwer Academic Publishers.

Biegler, L. T., Nocedal, J., Schmid, C., & Ternet, D. J. (2000). Numerical experience with a reduced Hessian method for large scale constrained optimization. *Computational Optimization and Applications*, *15*, 45.

Biegler, L. T., Schmid, C., & Ternet, D. (1997). A multiplier-free, reduced hessian method for process optimization. In L. T. Biegler, T. F. Coleman, A. R. Conn, & F. N. Santosa (Eds.), *Large-scale optimization with applications. Part II: Optimal design and control* (p. 101). Springer.

Boggs, P. T., & Tolle, J. W. (1995). Sequential quadratic programming. *Acta Numerica*, *4*, 1–52.

Byrd, R. H., Gilbert, J. C., & Nocedal, J. (2000). A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, *89*(1), 149–185.

Casado, L. G., Martinez, J. A., Garcia, I., & Sergeyev, Ya. D. (2003). New interval analysis support functions using gradient information in a global minimization algorithm. *Journal of Global Optimization*, *25*, 345–362.

Cervantes, A. M., Waechter, A., Tutuncu, R., & Biegler, L. T. (2000). A reduced space interior point strategy for optimization of differential algebraic systems. *Computational Chemical Engineering*, *24*, 39–51.

Coello, C. A. C. (2000a). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, *17*, C319–C346.

Coello, C. A. C. (2000b). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, *41*(2), 113–127.

Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computational Methods of Applied Methodology and Engineering*, *191*, 1245–1287.

Coello Coello, C. A., & Christiansen, A. D. (2000). Multiobjective optimization of trusses using genetic algorithms. *Computers & Structures*, *75*(6), 647–660.

Coleman, T. F., & Li, Y. (1996). An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal of Optimization*, *6*, 418–445.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computational Methods of Applied Methodology and Engineering*, *18*, 311–318.

Edgar, T. F., & Himmelblau, D. M. (1988). *Optimization of chemical processes*. New York: McGraw-Hill.

Floudas, C. A., & Visweswaran, V. (1993). A primal-relaxed dual global optimization approach. *Journal of Optimizational Theory and Applications*, *78*, 187.

Ge, R., & Qin, Y. (1990). The global convexized filled fnctions for global optimization. *Applied Mathematical Computations*, *35*, 131–158.

Gen, M., & Cheng, R. (1996). Interval programming using genetic algorithms. In *Proceedings of the 1st international symposium on soft computing for industry*.

Gen, M., & Cheng, R. (1997). *Genetic algorithms & engineering design*. New York: Wiley.

Glover, F., & de Werra, D. (Eds.). (1993). *Tabu Search, vol. 41 of annals of operations research*. J.C. Baltzer AG.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefettstette, J. J. (Ed.), *Proceedings of the 2nd international conference of genetic algorithms and their applications*, Hillsdale, NJ: Lawrence Erlbaurn (pp. 41–49).

Goldberg, D. E. & Wang, L. (1997). *Adaptive niching via coevolutionary sharing*, IlliGAL Report 97007.

Handbook of Evolutionary Computation. (1997). IOP Publishing Ltd. and Oxford University Press, release 97/1.

Hadj-Alouane, A. B., & Bean, J. C. (1997). A genetic algorithm for the multiple-choice integer program. *Operational Research*, *45*, 92–101.

Himmelblau, D. M. (1972). *Applied non-linear programming*. The University of Texas, Austin, TX: McGraw-Hill Book Company.

Homaifar, A., Qi, C., & Lai, S. (1994). Constrained optimization via genetic algorithms. *Simulation*, *62*(4), 242–254.

Jang, W., Hahn, J., & Hal, K. R. (2005). Genetic/quadratic search algorithm for plant economic optimizations using a process simulator. *Computational Chemical Engineering*, *30*(2), 285–294.

Joines, J., & Houck, C. (1994). On the use of non-stationary penalty funtions to solve nonlinear constrained optimization problems with GAs. In *Fogel: Proceedings of* *the 1st IEEE conference on evolutionary computation*. Orlando, FL: IEEE Press (pp. 579–584).

Kirkpatrick, S., Gellat, D., & Vecci, M. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Liu, Y., Ma, L., & Zhang, J. (2002). Reactive power optimization by GA/SA/TS combined algorithms. *International Journal of Electrical Power*, *24*(9), 765–769.

Mansoornejad, B., Mostoufi, N., & Jalali-Farahani, F. (2008). A hybrid GA-SQP optimization technique for determination of kinetic parameters of hydrogenation reactions. *Computational Chemical Engineering*, *32*(7), 1447–1455.

Maria, G. (2004). A review of algorithms and trends inkinetic model identification for chemical and biochemical systems. *Chemical and Biochemical Engineering Quarterly*, *18*, 195.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs* (2nd ed.). Berlin, Germany: Springer.

Michalewicz, Z. (1995). Genetic algorithms, numerical optimization, and constraints. In L. Eshelman (Ed.), *Proceedings of the 6th international conference on genetic algorithms* (pp. 151–158). San Mateo, CA: Morgan Kaufmann.

Michalewic, Z., & Attia, N. F. (1994). Evolutionary optimization of constrained problems. In *Proceedings of the 3rd annual conference on evolutionary programming. (pp. 98–108)*. Singapore: World Scientific.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolution of Computers*, *4*(1), 1–32.

Miller, B. L., & Shaw, M. J. (1996). Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *Proceedings of the 3rd IEEE conference evolutionary computation. (pp. 786–791)*. Piscataway, NJ: IEEE Press.

Mohanta, D. K., Sadhu, P. K., & Chakrabarti, R. (2007). Deterministic and stochastic approach for safety and reliability optimization of captive power plant maintenance scheduling using GA/SA-based hybrid techniques: A comparison of results. *Reliability Engineering & System Safety*, *92*(2), 187–199.

Pham, Q. T. (1995). Competitive evolution: A natural approach to operator selection. In X. Yao (Ed.), *Progress in evolutionary computation* (pp. 49–60). Berlin: Springer.

Petrowski, A. (1996). A clearing procedure as niching method for genetic algorithms. In *Proceedings of the 3rd IEEE conference evolutionary computation. (pp. 798–803)*. Piscataway, NJ: IEEE Press.

Ragsdell, K. M., & Phillips, D. T. (1976). Optimal design of a class of welded structures using geometric programming. *ASME Journal of Engineering for Industry: Series B*, *98*(3), 1021–1025.

Reklaitis, G. V., Ravindran, A., & Ragsdell, K. M. (1983). *Engineering optimization methods and applications*. New York: Wiley.

Sarma, P. V. L. M., & Reklaitis, G. V. (1982). Optimization of a complex chemical process using an equation oriented model. *Mathematical Programming Study*, *20*, 113–160.

Schittkowski, K., & Monma, C.L. (Ed.), (1986). NLPQL: A FORTRAN subroutine solving constrained non-linear programming problems, *Annals of Operations Research*, *5*, 485–500.

Shen, P. (1999). An interval algorithm for finding all global minimizers of nonsmooth functions. *Chinese Journal of Numerical Mathematical Applications*, *21*(3), 15–20.

Shopova, E. G., & Vaklieva-Bancheva, N. G. (2006). BASIC—A genetic algorithm for engineering problems solution. *Computational Chemical Engineering*, *30*, 1293–1309.

Smith, A., & Tate, D. (1993). Genetic optimization using a penalty function. In *Forrest: Proceedings of the 5th international conference on genetic algorithms. (pp. 499–505)*. San Mateo, CA: Morgan Kaufmann Publishers.

Summanwar, V. S., Jayaraman, V. K., Kulkarni, B. D., Kusumakar, H. S., Gupta, K., & Rajesh, J. (2002). Solution of constrained optimization problems by multi-objective genetic algorithm. *Computational Chemical Engineering*, *26*, 1481–1492.

Sadegheih, A. (2007). Sequence optimization and design of allocation using GA and SA. *Applied Mathematical Computations*, *186*(2), 1723–1730.

Waltz, R. A., Morales, J. L., Nocedal, J., & Orban, D. (2006). An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming*, *107*(3), 391–408.

Waltz, R. A., & Plantenga, T. D. (2008). KNITRO ^{TM} 5.2 user's manual, www.ziena.com/knitro.htm.

Yin, X., & Germay, N., (1993). A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In Albrecht, R. F., Reeves, C. R., Steele, N. C. (Eds.), *Proceedings of the international conference artificial neural nets and genetic algorithms. (pp. 450–457)*, New York: Springer-Verlag.

Yu, H., Fang, H., Yao, P., & Yuan, Y. (2000). A combined genetic algorithm/simulated annealing algorithm for large scale system energy integration. *Computational Chemical Engineering*, *24*, 2023.