Pergamon

Computers Math. Applic. Vol. 36, No. 5, pp. 11–21, 1998

0898-1221/98 $19.00 + 0.00

PII: S0898-1221(98)00146-1

A Hybrid Genetic Algorithm for a Type of Nonlinear Programming Problem

JIAFU TANG AND DINGWEI WANG
P.O. 135, School of Information Science & Engineering
Northeastern University
Shenyang, Liaoning 110006, P.R. China

A. IP
Department of Manufacturing Engineering
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong, S.A.R.

R. Y. K. FUNG
Department of Manufacturing Engineering & Engineering Management
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, S.A.R.

(Received November 1997; revised and accepted May 1998)

Abstract—Based on the introduction of some new concepts of semifeasible direction, Feasible Degree (FD_1) of semifeasible direction, feasible degree (FD_2) of illegal points 'belonging to' feasible domain, etc., this paper proposed a new fuzzy method for formulating and evaluating illegal points and three new kinds of evaluation functions and developed a special Hybrid Genetic Algorithm (HGA) with penalty function and gradient direction search for nonlinear programming problems. It uses mutation along the weighted gradient direction as its main operator and uses arithmetic combinatorial crossover only in the later generation process. Simulation of some examples show that this method is effective.

Keywords—Nonlinear programming, Hybrid genetic algorithm, Weighted gradient direction, Feasible degree, Semifeasible direction.

1. INTRODUCTION

Nonlinear Programming (NLP) is an important branch of Operations Research and has wide applications in the areas of military, economics, engineering optimization, and science management. There are several types of traditional methods for nonlinear programming [1], however, since there are many local optimizations for NLP, most of the solution methods may solve it only on an approximate basis. Recently, based on strict optimization theory and algorithm, many researchers have proposed some new stochastic optimization methods, such as the genetic algorithm [2–4], simulated annealing [5], Tabu search [6], and various hybrid methods [5,6].

The authors are greatly indebted to the referees for their suggestions and comments on the revision of the paper. This paper is supported by the National Natural Science Foundation NNSF (Project No. 69684005) of the P.R. of China.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

The Genetic Algorithm (GA), which was first proposed by Holland [7], is one of the most important stochastic optimization methods. As an intelligent optimization method, GA has made great achievements in the solution to traveling salesman problems, transport problems, 0-1 programming problems, and multiobjective optimization problems. However, it has made little contribution to nonlinear programming problems [2,4].

In fact, in the construction and application of GA to NLP, the coding and decoding processes are important and difficult. In addition, the handling with the system constraints, especially the measurement and evaluation of illegal chromosomes (points) are key techniques with GA. Currently, several methods have been developed to deal with system constraints and have been reviewed by Michalewicz in 1995 [8]. Among of which, a large penalty in the construction of the fitness function was often used to evaluate the infeasible solutions, but this is essentially narrow of the search space, by eliminating all illegal points from the evolutionary process and may lessen the ability to find better candidates for the global optimization. Li and Gen [2] developed a method for Nonlinear Mixed Integer Programming (NMIP) problems by means of GA and a type of penalty function.

This paper focuses on the application of GA to a type of differentiable nonlinear programming problem. By means of introducing some new concepts of semifeasible direction, feasible degree (FD_1) of semifeasible direction, feasible degree (FD_2) of illegal points, a new fuzzy method for formulating and evaluating illegal points, and three new kinds of evaluation function are proposed in this paper. Based on the fuzzy method and new kinds of evaluation functions, we have developed a special Hybrid Genetic Algorithm (HGA) with penalty function and gradient direction search to solve nonlinear programming problems.

The rest of this paper is organized as follows. Section 2 explains the basic idea and overall procedure of HGA. Finally, simulation results and analysis of some examples and the conclusion are given in Sections 3 and 4 .

2. HYBRID GENETIC ALGORITHM WITH PENALTY FUNCTION AND GRADIENT DIRECTION SEARCH

2.1. Canonical Form of NLP

NLP problems with n variables and m constraints may be written as the following Canonical form NLP:

$$\begin{aligned} \max \quad & f(x) = f(x_1, x_2, \ldots, x_n), \\ \text{s.t.} \quad & x \in Q = \{x \in E_n \mid g_i(x) \le 0,\ i = 1, 2, \ldots, m\}. \end{aligned} \tag{1}$$

In this paper, we assume that $f(x)$ and $g_i(x)$ are differentiable in E_n.

2.2. Basic Idea of the Hybrid Genetic Algorithm

In the procedure of solution to NLP by means of a penalty function method, we first transmit NLP into a unconstrained optimization problem by means of a penalty function, then solve a series of unconstrained optimization problems with a certain penalty multiplier to obtain the optimal solution or near optimal solution to the original problem. As the penalty multiplier tends to zero or infinite, the iteration point also tends towards optimal. However, at the same time, the objective function of the unconstrained optimization problem might gradually become worse. This lead to computer difficulties in implementing the penalty function methods to solve NLP.

On the other hand, when we apply traditional GA to solve NLP, a scheme of coding and decoding processes for optimizing variables is needed. Moreover, due to the complexity and differences of constraints in actual optimization problems, there is not a general coding method for all types of optimization problems. Meanwhile, the handling with system constraints, especially the formulation and evaluation of illegal chromosomes are critical techniques.

Based on the above analysis in this paper, we propose a special hybrid genetic algorithm with penalty function and gradient direction search to solve NLP. The basic idea may be described as follows. First, randomly produce an initial population with the size of *popsize* individuals. In the process of iteration, for an individual $x_i \notin Q$, give it a less fitness function by means of embedding information of illegal chromosomes into the evaluation process, so that it may have less chance than others to be selected as parents to reproduce children in the later generations. Each individual is selected to reproduce children by mutation along the weighted gradient direction, according to the selection probability which depends on its fitness function (objective function). As the generation increases, the individuals with less fitness functions die out gradually, namely, the individuals $x_i \in Q$ with less objective functions and individuals $x_i \notin Q$ die out gradually, and the individuals maintained in the population are the individuals with a high value of objective function. After a number of generations, the individuals' objective function values reach the optimal or near optimal from the two sides of feasible domain.

2.3. Weighted Gradient Direction

For an individual x, if $x \in Q$, then the objective function may be improved along the gradient direction of objective function $\nabla f(x)$.

For an individual x, if $x \notin Q$, it denotes that x is out of the feasible domain. Let

$$I^+ = \{i \mid g_i(x) > 0,\ x \in E_n\}.$$

For $i \in I^+$, if x moves along the negative gradient direction $-\nabla g_i(x)$, it may satisfy $g_i(x) \le 0$. The greater the weight is, the faster $g_i(x) \le 0$ may be achieved.

Based on the above idea, we construct a weighted gradient direction [4,9], denoted by $d(x)$, which is defined as follows:

$$d(x) = w_0 \nabla f(x) - \sum_{i=1}^{m} w_i \nabla g_i(x), \tag{2}$$

where w_i is the weight of the gradient direction, generally, $w_0 = 1$ and w_i is defined as

$$w_i = \begin{cases} 0, & g_i(x) \le 0, \\ \delta_i, & g_i(x) > 0, \end{cases} \tag{3}$$

$$\delta_i = \frac{1}{1 - (g_i(x))/(g_{\max}(x) + \delta)}, \tag{4}$$

$$g_{\max}(x) = \max\{g_i(x),\ i = 1, 2, \dots, m\}, \tag{5}$$

where δ is a very small positive number.

Formula (4) means that the weight of gradient direction increases as the increase of $g_i(x)$ when $g_i(x) > 0$.

Then $x_j^{(k+1)}$ generated from $x_i^{(k)}$ by mutation along the weighted gradient direction $d(x)$ can be described as

$$x_j^{(k+1)} = x_i^{(k)} + \beta^{(k)} d\left(x_i^{(k)}\right), \tag{6}$$

where $\beta^{(k)}$ is a step-length of the Erlang distribution random number with declining means, which is generated by the random number generator

$$E\left(\beta^{(k)}\right) = \frac{E\left(\beta^{(k-1)}\right)}{M_1}, \tag{7}$$

where M_1 is a declining coefficient.

From (2)–(7), it can be seen that for an individual $x \in Q$, $d(x) = \nabla f(x)$, it moves along the direction of $\nabla f(x)$, the objective function may be improved, so as to reach the near optimal; for some other individuals $x \notin Q$, the bigger the $g_i(x)$, the farther apart these are from the feasible domain Q, they may require the high weight w_i in order to reach or move into feasible domain.

2.4. Formulation and Evaluation for Illegal Points

In the application of GA to optimization problems with constraints, it is very important to formulate and evaluate the illegal chromosomes. In this section, we introduce some concepts of semifeasible direction and feasible degree of illegal points, etc.

DEFINITION 1. *The* **distance** *$d(x, Q)$ between point x and feasible domain Q is the maximum violation of constraints at point x.*

According to the definition, $d(x, Q)$ has the following formula:

$$d(x, Q) = \max\{0, g_{\max}(x)\}. \tag{8}$$

The distance reflects the information of the relationship between point x and feasible domain Q; if $d(x, Q) = 0$, this indicates that $x \in Q$; conversely, $d(x, Q) > 0$, and the greater the $d(x, Q)$ is, the worse the performance of x 'belong to' Q is, i.e., x is further from the feasible domain Q.

Let

$$\begin{aligned} G_i(x) &= \frac{g_i(x)}{\|\nabla g_i(x)\|}, \\ G_{\max}(x) &= \max\{G_i(x),\ i = 1, 2, \dots, m\}, \\ K &= \arg\{i \mid G_i(x) = G_{\max}(x),\ i = 1, 2, \dots, m\}. \end{aligned}$$

DEFINITION 2. *A nonzero vector $\nabla g_K(x)$ is called the* **dominated semifeasible direction.**

DEFINITION 3. *A nonzero vector z is defined as* **point to** *feasible domain Q at point $x \notin Q$, if it satisfies*

$$z^{\mathrm{T}}\left(-\nabla g_K(x)\right) > 0. \tag{9}$$

DEFINITION 4. *A nonzero vector z is defined as* **departure from** *feasible domain Q at point $x \notin Q$, if it satisfies*

$$z^{\mathrm{T}}\left(-\nabla g_K(x)\right) \leq 0. \tag{10}$$

DEFINITION 5. *A nonzero vector z is called the* **semifeasible direction** *at point $x \notin Q$, if it is a point to feasible domain.*

The relationship between weighted gradient direction, the dominated semifeasible direction and semifeasible direction is shown in Figure 1.

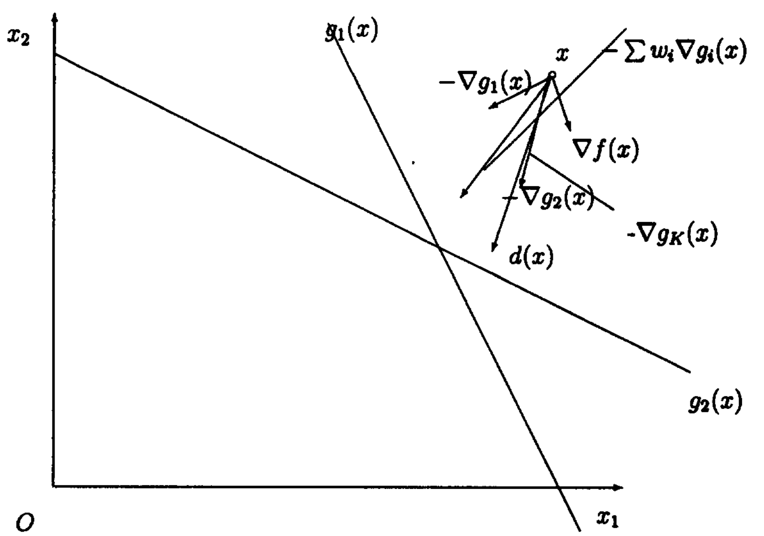

Figure 1. The illustration of semifeasible direction and weighted gradient direction.

THEOREM 1. *For* $\forall x \notin Q$, $z = -\sum_{i=1}^{m} w_i \nabla g_i(x)$ *is semifeasible direction.*

PROOF. If z satisfies (9), then z is the semifeasible direction; if not, then we may adjust the coefficient δ to obtain a semifeasible direction.

THEOREM 2. *For* $\forall x \notin Q$, *the weighted gradient direction* $d(x)$ *constructed by (2) is the semifeasible direction.*

PROOF. If $d(x)$ satisfies (9), then $d(x)$ is the semifeasible direction; if not, then we may adjust the coefficients w_0 or δ as follow to obtain a semifeasible direction.

(1) Let $w_0 = w_0/2$, and replace it in (2), test the subjection of (9); if it does not hold, repeat the process until it does.
(2) Let $\delta = \delta/2$, and replace it in (4), test the subjection of (9); if it does not hold, repeat the process until it does.

DEFINITION 6. **Feasible Degree** FD_1 *of a semifeasible direction at point* $x \notin Q$ *is defined as*

$$\mathrm{FD}_1 = d(x)^{\mathsf{T}} \left(-\nabla g_K(x)\right). \tag{11}$$

Obviously, the feasible degree FD_1 of the semifeasible direction at point x reflects only the deviation of the weighted gradient direction from the dominated semifeasible direction, not the degree of point far from feasible domain.

In the follows, we formulate the degree of a point far from feasible domain in fuzzy methodology, by means of introducing the concept of feasible degree of an illegal point 'belonging to' feasible domain.

Let γ_i denote the degree of subjection to the i^{th} constraints at point x, which can be defined as

$$\gamma_i = \begin{cases} 1, & \text{if } g_i(x) \le 0, \\ 1 - \dfrac{g_i(x)}{d(x,Q)}, & 0 < g_i(x) \le d(x,Q), \\ 0, & \text{else.} \end{cases} \tag{12}$$

DEFINITION 7. *Feasible degree* FD_2 *of an illegal point (chromosome)* x *'belonging to' feasible domain* Q *is defined as*

$$\mathrm{FD}_2 = \sum_{i=1}^{m} \frac{\gamma_i}{m}. \tag{13}$$

FD_2 reflects the degree of point x 'belonging to' feasible domain. If $\mathrm{FD}_2 = 1$, it indicates that x belongs completely to the feasible domain, i.e., $x \in Q$; otherwise, if $\mathrm{FD}_2 = 0$, it implies that x completely does not belong to the feasible domain. If $0 < \mathrm{FD}_2 < 1$, it denotes that the membership degree of point x belonging to the feasible domain in the sense of fuzzy theory, although it does not belong to the feasible domain in the sense of crisp mathematics.

Based on the above analysis, the following three methods may be suggested to evaluate the illegal point (chromosome).

METHOD 1. Embedding the feasible degree of the semifeasible direction, i.e.,

$$\mathrm{eval}(x) = \frac{f(x)}{(1 + 1/\mathrm{FD}_1)^p}, \qquad p \ge 1. \tag{14}$$

METHOD 2. Embedding the feasible degree of the illegal point 'belonging to' feasible domain, i.e.,

$$\mathrm{eval}(x) = \frac{f(x)}{(1 + 1/\mathrm{FD}_2)^p}, \qquad p \ge 1. \tag{15}$$

METHOD 3. Embedding the distance far from the feasible domain, i.e.,

$$\mathrm{eval}(x) = \frac{f(x)}{(1 + g_{\max})^p}, \qquad p \ge 1. \tag{16}$$

2.5. Overall Scheme of Hybrid GA

1. CHROMOSOME REPRESENTATION. In this problem, the chromosome is schemed as a real representation, i.e., $x = (x_1, x_2, \dots, x_n)$.

2. INITIAL POPULATION. In order to speed up the search process, select upper (u_i) and lower bound (l_i) for each gene x_i, then the initial population is generated as follows.

Procedure: Initialize
begin
for $k \leftarrow 1$ **to** *pop_size* **do**
  for $i \leftarrow 1$ **to** n **do**
  $x_i^{(k)} \leftarrow$ *random* (l_i, u_i)
   $i \leftarrow i + 1;$
  end
   $k \leftarrow k + 1;$
end
end

where random (a, b) is a random number with unity distribution of (a, b).

3. CROSSOVER. Crossover is not the main operator, which may only be used in the later generation procedures, when the whole individuals are all in the feasible domain and the cases that there is not any improvement of objective function for any individuals in continuous more than a definite number *num* (in general, select num = 2) generations. In this case, the arithmetic combinatorial crossover operator is suggested. It is noted that even though in the case of the feasible domain Q is not convex, the effective of the algorithm is not affected, because p_c is selected as very small, moreover, the operator need not guarantee the feasibility of the offspring. In fact, the feasible chromosome is mostly obtained by mutation operator. The offspring x_i^{k+1} generated by the arithmetic combinatorial operator is described as follows:

$$x_i^{k+1} = \alpha x_i^k + (1 - \alpha) x_j^{k+1},$$

where x_i^k is selected as the strategy of Roulette Wheel, and x_j^k is selected randomly, $\alpha \in \mathsf{U}(0, 1)$.

4. MUTATION. Mutation is the main operator, which is also the characteristic of the hybrid GA. Mutation along the weighted gradient direction is performed as (6).

5. EVALUATION OF CHROMOSOME. Evaluation of the chromosome is an important procedure of GA. In this paper, we suggest three new methods to evaluate illegal chromosomes. According to these methods, the information of illegal chromosomes is embedded into the evaluation function in order to measure the degree of illegal chromosomes far from feasible domain so that it could not be rejected as parents to reproduce children in the later generation process. Hence, the genetic search ensures the optimum from both feasible and infeasible domain.

Fitness function $F(x)$ is calculated as

$$F(x) = \begin{cases} f(x), & g_{\max}(x) \le 0, \\ \operatorname{eval}(x), & \text{else}, \end{cases} \tag{17}$$

where eval(x) is referred as (14)–(16).

6. STOP RULE. According to the degree of precision required, a maximum iteration number NG is determined to be the stop rule.

3. NUMERICAL EXAMPLE AND ANALYSIS

To clarify the effectiveness of the hybrid GA with penalty function and gradient direction search, in this section we give some test examples using linear constraint and nonlinear constraint as well as nonconvex constraint. The comparison results obtained by way of Penalty Function Method (PFM), traditional GA, and the hybrid GA are also given in this section.

3.1. The Design of the Test Problem

To justify the performance of the proposed HGA, we select the following three test problems as example of comparison, where test **P1** and **P2** are from [1] and quadratic problems with linear and nonlinear constraints, respectively, while the **P3** modified from [1] is a problem with nonlinear objective and nonconvex constraints.

TEST P1. (See [1, p. 413].)

$$\begin{aligned} \text{Max} \quad & f(x) = -2x_1^2 + 2x_1x_2 - 2x_2^2 + 4x_1 + 6x_2, \\ \text{s.t.} \quad & x_1 + x_2 \le 2, \\ & x_1 + 5x_2 \le 5, \\ & x_1, x_2 \ge 0. \end{aligned}$$

TEST P2. (See [1, p. 419].)

$$\begin{aligned} \text{Max} \quad & f(x) = -2x_1^2 + 2x_1x_2 - 2x_2^2 + 4x_1 + 6x_2, \\ \text{s.t.} \quad & 2x_1^2 - x_2 \le 0, \\ & x_1 + 5x_2 \le 5, \\ & x_1, x_2 \ge 0. \end{aligned}$$

TEST P3.

$$\begin{aligned} \text{Max} \quad & f(x) = -(1-x_1)^2 + 10\left(x_2 - x_1^2\right)^2 + x_1^2 - 2x_1x_2 + \exp(-x_1 - x_2), \\ \text{s.t.} \quad & x_1^2 + x_2^2 \ge 16, \\ & x_2 - x_1^2 \le 1, \\ & x_1 + x_2 \le 20, \\ & x_1, x_2 \ge 0. \end{aligned}$$

3.2. Implementation of PFM and Traditional GA

Penalty function method [1] is one of commonly used traditional optimization methods for NLP. In implementation of the PFM, the original problem NLP is transmitted into the following unconstrained optimization problem by means of penalty function:

$$\max f(x) - \sum_{i=1}^{m} \mu_i \max\{0, g_i(x)\}, \qquad x \in E_n,$$

and the steepest descent algorithm is applied to solve the above unconstrained problem. While the uniform crossover and position based mutation, as well as the constant penalty are applied in the implementation of traditional GA.

Table 1. Comparison results of test problems by way of PFM, GA, and HGA.

Test Problems	Algorithm	Optimal Solution	Best Solution	Best Error(%)	Mean Error(%)	Worst Error(%)
P1	PFM	7.160	6.995	2.30	8.25	10.3
	GA	7.160	7.10	0.84	15.94	21.4
	HGA	7.160	7.16085*	0.000	0.15	0.261
P2	PFM	6.613086	6.145	3.04	10.45	12.5
	GA	6.613086	6.387	3.45	14.48	19.68
	HGA	6.613086	6.61305	0.001	0.088	0.153
P3	PFM	1600039.0	1519431.5	5.037	11.35	14.30
	GA	1600039.0	1600039.0	0.0	61.60	93.33
	HGA	1600039.0	1600039.0	0.0	0.0181	0.0748

PFM—the results over 50 trials with initial point generated by random number generator.
HGA—the results as $NG = 50$, popsize $= 50$.
*: Best solution is better than the Optimal solution only due to the degree of precision.

Table 2. Comparison results of the test problems by way of different evaluation functions for HGA.

Test	Method Solution	Best error(%)	Best error(%)	Mean error(%)	Worst (%)	Inf_per (%)
P1	**Method 0**	7.15286	0.71	40.04	80.01	60–80
	Method 1	7.16085	0.00	13.08	50.00	05–25
	Method 2	7.15799	0.028	23.56	66.6	0–15
	Method 3	7.15791	0.029	0.150	0.261	0–5
P2	**Method 0**	6.61295	0.002	28.00	80.00	60–80
	Method 1	6.61299	0.001	11.15	50.01	0–15
	Method 2	6.61300	0.000	4.00	66.1	0–15
	Method 3	6.61305	0.000	0.088	0.153	0–5
P3	**Method 0**	1600039.0	0.0	61.60	93.33	60–80
	Method 1	1600039.0	0.0	41.15	74.99	50–80
	Method 2	1600039.0	0.0	47.07	83.44	50–80
	Method 3	1600039.0	0.0	0.0181	0.0748	30–50

Note: The results as $NG = 100$, popsize $= 50$; Inf_per—percent of infeasible chromosomes in the last genertaion; **Method 0**—a large constant penalty as evaluation function.

3.3. The Analysis of the Results for Test Problems by HGA

The comparison results of the test problems by way of the Penalty Function Method (PFM), traditional GA, and the proposed hybrid genetic algorithm (HGA) are shown in Table 1. The comparison results of the test problems by way of the three new evaluation function methods and constant penalty method, and the best solution as different population size and maximum generations are shown in Tables 2 and 3, respectively. The best, mean, and the worst solution, as well the number of infeasible chromosomes at each iteration for **P2** by HGA are shown in Table 4.

From Table 1, we can see that HGA is much more effective than PFM in view of performances of the best error, mean error, and worst error, especially when there are no distinguished differences

Table 3. The best solution as different population size and maximum generations.

Test Problem	Popsize	Best* Solution	Test Problem	NG	Best** Solution
P1	10	7.15395	P1	15	3.06378
	20	7.15759		25	3.06355
	30	7.15820		40	3.06355
	40	7.15666		50	7.16085
	50	7.16085		75	7.15715
	60	7.15732		100	7.15868
	80	7.15798		125	7.15877
	100	7.15789		150	7.15887
P2	10	6.61299	P2	15	3.06387
	20	6.61292		25	3.06387
	30	6.61289		40	6.61287
	40	6.61306		50	6.61295
	50	6.61308		75	6.61296
	60	6.61308		100	6.613080
	80	6.61307		125	6.613080
	100	6.61308		150	6.613083
P3	10	1600039.0	P3	15	1600039.0
	20	1600039.0		25	1600039.0
	30	1600039.0		40	1600039.0
	40	1600039.0		50	1600039.0
	50	1600039.0		75	1600039.0
	60	1600039.0		100	1600039.0
	80	1600039.0		125	1600039.0
	100	1600039.0		150	1600039.0

*: the best solution as $NG = 100$; **: the best solution as popsize $= 50$.

between the best error, mean error, and worst error for HGA, while there exist great differences between them by way of traditional heuristic method, i.e., PFM, which indicate that PFM is greatly affected by the selection of initial points. As well, it is shown from Table 1 that HGA is more effective than traditional GA.

Table 2 is the comparison results of the test problems between the new evaluation function methods and traditional constant penalty method, from which we can see that the new evaluation function methods are all superior than constant penalty method from the point of both the best error, mean error and the worst error.

It can be seen from Table 3 that the best solution is slightly affected by the population size popsize, and is greatly affected by the maximum generation NG untill it reaches a suitable level, such as $NG = 50$.

It can be seen from Table 4 that the best error becomes smaller and smaller as the increase of generations and there is almost no difference in the best error from the generation of 80^{th}, which indicates the convergency of HGA. In the meantime, the difference between the best error and the worst error approaches zero as the increase of generation, which implies that solutions in the last generation are all in the near domain of the optimal solution, i.e., they are all accepted as a satisfactory solution. Additional, it can reflects the percentage of infeasible chromosomes in the population at each iteration, the number of infeasible chromosomes decreases as the increases of

Table 4. The best, mean, the worst solution and number of infeasible solutions for **P2** at each iteration by HGA.

ItNo	Infeasible Number	Best Solution	Mean Solution	Worst Solution	ItNo	Infeasible Number	Best Solution	Mean Solution	Worst Solution
1	48	3.063550	−1.963721	−14.507540	51	32	6.602537	6.540881	6.351209
2	46	3.923996	−8.951345	−18.406640	52	24	6.603416	6.560880	6.407640
3	41	0.00000	−5.939373	−17.596200	53	42	6.601602	6.535637	6.439693
4	40	3.940441	−2.193960	−11.424870	54	11	6.609656	6.585100	6.475861
5	27	3.517756	−1.104784	−6.317472	55	31	6.612067	6.583463	6.499317
6	32	3.882819	−0.024352	−2.313398	56	33	6.610638	6.576521	6.501623
7	2	3.946857	0.202615	0.000000	57	16	6.610380	6.599028	6.577125
8	36	0.000000	−0.263667	−0.625713	58	42	6.610638	6.579856	6.518928
9	50	0.470409	−0.974160	−2.644845	59	9	6.611909	6.600630	6.545513
10	26	3.958498	0.669238	−3.865976	60	32	6.612026	6.597582	6.559607
11	28	0.665742	0.097536	−0.242842	61	30	6.610950	6.597780	6.560095
12	10	0.822507	−0.077686	−0.873723	62	34	6.611289	6.597525	6.565237
13	9	3.967764	1.639398	0.000000	63	21	6.612717	6.604111	6.583730
14	36	1.376387	0.558994	0.000000	64	20	6.612951	6.606650	6.578149
15	50	1.089149	0.303759	−0.428773	65	36	6.611935	6.603707	6.589171
16	14	6.334483	1.344784	0.000000	66	29	6.612157	6.607095	6.595370
17	42	3.797592	1.444057	0.000000	67	27	6.612237	6.608098	6.598180
18	40	2.286361	0.717346	0.000000	68	22	6.612535	6.607291	6.597160
19	10	4.378037	2.420234	0.000000	69	20	6.612622	6.610566	6.601147
20	50	3.814319	2.213592	0.701667	70	33	6.612540	6.609030	6.601996
21	49	3.691129	1.650333	0.571107	71	32	6.612876	6.609776	6.602560
22	12	5.486464	3.196831	1.742391	72	13	6.612928	6.611180	6.605491
23	49	5.952384	3.171359	1.514548	73	36	6.612814	6.610794	6.605368
24	47	4.652226	3.268739	1.609322	74	10	6.612985	6.611877	6.608120
25	4	5.761828	3.606856	2.181158	75	17	6.612953	6.611842	6.608649
26	43	6.093210	4.325548	2.618724	76	37	6.612969	6.611675	6.608157
27	20	6.262978	4.172750	2.970258	77	29	6.612937	6.612007	6.609680
28	33	6.580002	5.281199	2.736286	78	27	6.612792	6.612007	6.610425
29	35	6.402060	4.791120	2.995108	79	18	6.612975	6.612590	6.611485
30	27	6.321744	4.869431	3.473816	80	30	6.612936	6.612047	6.610817
31	25	6.444650	5.520216	3.933339	81	17	6.613067	6.612520	6.611129
32	34	6.330513	5.036617	3.791852	82	15	6.613078	6.612683	6.611564
33	20	6.557756	5.895343	4.400737	83	11	6.613064	6.612794	6.611892
34	36	6.589750	5.734088	4.151553	84	13	6.613031	6.612749	6.611916
35	29	6.415943	5.654635	4.436711	85	14	6.613067	6.612813	6.612110
36	23	6.483630	5.857147	4.916823	86	19	6.613071	6.612784	6.612139
37	32	6.553962	6.013317	5.000106	87	8	6.613067	6.612915	6.612493
38	24	6.553856	6.173467	5.151730	88	6	6.613069	6.612909	6.612663
39	34	6.571466	6.274135	5.542740	89	5	6.613073	6.612957	6.612496
40	34	6.578858	6.211339	5.312654	90	6	6.613079	6.612958	6.612568
41	27	6.566411	6.288830	5.718469	91	4	6.613083	6.612985	6.612669
42	29	6.599164	6.340402	5.750092	92	3	6.613069	6.612998	6.612786
43	28	6.561859	6.382557	5.717593	93	0	6.613081	6.613004	6.612864
44	23	6.603125	6.465948	6.131230	94	0	6.613081	6.613040	6.612833
45	26	6.602304	6.442135	6.034008	95	0	6.613068	6.613035	6.612905
46	37	6.557187	6.366481	5.971469	96	0	6.613079	6.613052	6.612929

Table 4. (cont.)

ItNo	Infeasible Number	Best Solution	Mean Solution	Worst Solution	ItNo	Infeasible Number	Best Solution	Mean Solution	Worst Solution
47	21	6.602222	6.520334	6.311711	97	0	6.613083	6.613043	6.612932
48	37	6.603067	6.495034	6.151949	98	0	6.613084	6.613045	6.612927
49	22	6.611194	6.528231	6.246696	99	0	6.613082	6.613054	6.612963
50	32	6.610987	6.532200	6.358410	100	0	6.613081	6.613061	6.612971

Popsize = 50, optimal solution is 6.13086; ItNo–iteration number, infeasible number–number of infeasible chromosome.

generation, which implies that the genetic search can converge to the optimal or near optimal solution from both sides of the feasible and infeasible domain due to the introduction of new evaluation function methods.

4. CONCLUSION

By means of embedding a penalty function method and a gradient direction search into the GA, this paper developed a special hybrid genetic algorithm (HGA) with mutation along the weighted gradient direction for a type of NLP. It is the first time that these new concepts of semifeasible direction, feasible degree (FD_1) of semifeasible direction, feasible degree (FD_2) of illegal points 'belonging to' feasible domain, etc., have been proposed for formulating and measuring the illegal point. Three new kinds of evaluation functions are also proposed in this paper. Convergency analysis and some simulation results for some test problems shown that the HGA is effective for the differentiable NLP. In comparison with the traditional genetic algorithm, the HGA has the following characteristics.

(1) It uses a special mutation along the weighted gradient direction as the main operator, and the arithmetic combinative crossover is only used in the later generation process, in which the whole individuals are all in the feasible domain.

(2) Embedding the information of illegal chromosomes into the evaluation process to develop three new kinds of evaluation function.

(3) It can converge to the optimum from both sides of the feasible domain and the infeasible domain.

REFERENCES

1. M.S. Bazaraa and L.M. Shetty, *Non-Linear Programming: Theory and Algorithms*, John Wiley & Sons, New York, (1985).
2. Y. Li and Mitsuo Gen, Non-linear mixed integer programming problems using genetic algorithm and penalty function, In *Proceedings of 1996 IEEE Int. Conf. on SMC*, pp. 2677–2682.
3. Y. Takao, G. Mitsuo, T. Takeaki and Y. Li, A method for interval 0-1 number non-linear programming problems using genetic algorithm, *Computers & Industrial Engineering* **29**, 531–535, (1995).
4. J. Tang and D. Wang, A new genetic algorithm for non-linear programming problems, In the *Proceedings of the 36th IEEE Conference on Decision and Control*, pp. 4906–4907, (1997).
5. T.C. Lin *et al.*, Applying the genetic approach to simulated annealing in solving some NP-hard problems, *IEEE Trans. on SMC* **23**, 1752–1766, (1993).
6. F. Glover *et al.*, Genetic algorithm and Tabu search: Hybrid for optimizations, *Computers & Operations Research* **22**, 111–134, (1995).
7. J.H. Holland, Adaptation in natural and artificial systems, The University of Michigan Press, (1975).
8. Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, In *Evolutionary programming IV*, pp. 135–155, MIT Press, (1995).
9. J. Tang and D. Wang, An interactive approach based on GA for a type of quadratic programming problems with fuzzy objective and resources, *Computers & Operations Research* **24** (5), 413–422, (1997).
10. G.P. McCormick, *Nonlinear Programming: Theory, Algorithms and Applications*, John Wiley & Son, New York, (1983).