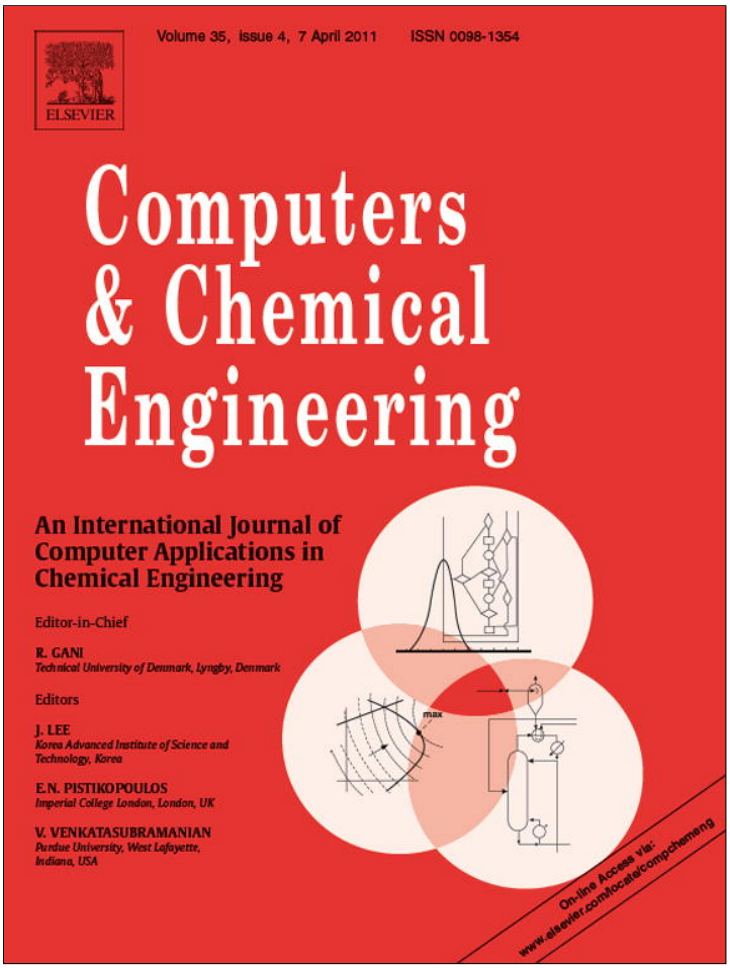

Computers and Chemical Engineering 35 (2011) 615–621

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

An improved genetic algorithm based on a novel selection strategy for nonlinear programming problems

Ke-Zong Tang*, Ting-Kai Sun, Jing-Yu Yang

School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China

ARTICLE INFO

Article history:
Received 16 October 2009
Received in revised form 8 June 2010
Accepted 24 June 2010
Available online 8 July 2010

Keywords:
Genetic algorithms
Nonlinear programming problems
Constraint-handling
Non-dominated solution
Optimization

ABSTRACT

Genetic algorithm is a heuristic population-based search method that incorporates three primary operators: crossover, mutation and selection. Selection operator plays a crucial role in finding optimal solution for constrained optimization problems. In this paper, an improved genetic algorithm (IGA) based on a novel selection strategy is presented to handle nonlinear programming problems. Each individual in selection process is represented as a three-dimensional feature vector composed of objective function value, the degree of constraints violations and the number of constraints violations. We can distinguish excellent individuals through two indices according to Pareto partial order. Additionally, IGA incorporates a local search (*LS*) process into selection operation so as to find feasible solutions located in neighboring areas of some infeasible solutions. Experimental results over a set of benchmark problems demonstrate that proposed IGA has better robustness, effectiveness and stableness than other algorithm reported in literature.

1. Introduction

Real world problems such as control and scheduling problems from the chemical engineering area, can be formulated as simple nonlinear programming (NLP) problems. In NLP problems, we define the single-objective optimization function as f: $R^n \to R$, where $n \geq 1$, n is the dimension of the decision solution vector, x. The aim of NLP problem is to search feasible solutions with better objective values. Formally speaking, it can be described as follows:

$$Min f(x) \tag{1}$$

$$s.t \quad g_i(x) \leq 0, \quad i = 1, 2, \ldots, m, \tag{2}$$

$$h_j(x) = 0, \quad j = 1, 2, \ldots, p, \tag{3}$$

where $x = [x_1, x_2, \ldots, x_n] \in R^n$, $f(x)$ is the objective function of the constrained problem and $f(x) \in R$. m is the number of inequality constraints and p is the number of equality constraints. Generally, equality constraints can be transformed into inequality constraints by using $|h_j(x)| - \phi \leq 0$ (ϕ is a very small value). For convenience, we denote by S the feasible search region and name the whole search space by Ω. Obviously, $S \subseteq \Omega$. When there are two solutions, $x^a = (x_1^a, x_2^a, \ldots, x_n^a)$ and $x^b = (x_1^b, x_2^b, \ldots, x_n^b)$ in S, the solution x^a is said to dominate (in the Pareto sense) the solution x^b (denoted by $x^a \prec x^b$) if $x_i^a \leq x_i^b \; \forall i \in \{1,2,\ldots,n\}$ and $x_i^a < x_j^b \; \exists j \in \{1,2,\ldots,n\}$. In addition, a solution $x^* \in S$ is the global optimal solution if $f(x^*) \leq f(x)$ for every $x \in S$.

In most cases, different constraints are always in conflict, because in order to satisfy any of the constraints, maybe we would violate other constraints. One of the well-known constraint-handling techniques is the penalty function method, in which the violations of constraints of the solutions are incorporated into the objective function so that the original constrained problems with inequality constraints and equality constraints are transformed into unconstrained ones, in that case, we can apply any unconstrained optimization technique to the new unconstrained optimization problems. The penalty approach to constrained optimization problems is attributed to Courant (1962), and was developed and popularized by Fiacco and McCormick (1968). In recent years, a vast amount of work has been published on applications of penalty function method to solve constrained optimization problems in many engineering areas, such as constrained redundancy allocation problem of series system (Gupta, Bhunia, & Roy, 2009), traffic assignment problem (Shahpar, Aashtiani, & Babazadeh, 2008), and economic lot size scheduling problem (Sarker & Newton, 2002). As we know, a major disadvantage of the penalty approach is the choice of penalty factors. Consequently, in order to avoid the trial and error tuning process of penalty factors, Coello (2000) presented the notion of using co-evolution to adapt the penalty factors of a fitness function incorporated in a genetic algorithm (GA) for engineering optimization problems.

* Corresponding author.
E-mail address: tangkezong@126.com (K.-Z. Tang).

0098-1354/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2010.06.014

He and Wang (2007) proposed an effective co-evolutionary particle swarm optimization (PSO) for constrained problems, where PSO was applied to evolve both decision factors and penalty factors. In these methods, the penalty factors were treated as searching variables and evolved by GA or PSO to the optimal values.

Apart from the penalty function method, there are many other methods in literature for handling constraints. Coello and Montes (2002) proposed a dominance-based selection scheme to incorporate constraints into the fitness function of a genetic algorithm which was used for global optimization. Unlike the traditional mathematical programming methods, this approach does not require the use of a penalty function. Chootinan and Chen (2006) proposed a constraint-handling technique by taking a gradient-based repair method. The proposed technique is embedded into GA as a special operator.

During the past three decades, a global optimization technique, genetic algorithm, has been successfully applied to many engineering and chemical engineering problems, such as polymer design problem (Sundaram & Venkatasubramanian, 1998), batch beer fermentation (Carrillo-Ureta, Roberts, & Becerra, 2001), transportation problems (Li, Ida, & Gen, 1997), job scheduling problems (Park, Choi, & Kim, 2003), etc. However, in the implementation of GA to constraint problems, individuals (i.e., solutions to a problem) generated by applying crossover and mutation operators usually violate the system constraints resulting in infeasible individuals. Particularly, in the selection, although best individuals will be presented in the next generation with a higher probability, it does not search the space further. When a few infeasible individuals are located in neighboring areas of feasible individuals, if we attempt to produce descendants that are better than their parents, we have to find a way of estimating how close an infeasible individual is from the feasible region. This paper develops an improved genetic algorithm (IGA) where we concentrate on the selection operator, and give a novel strategy of individual selection where a new local search procedure is performed with the aim to further search feasible individuals located in neighboring areas of infeasible individuals. This local search procedure works by generating a temporary subset of solutions with a modified mutation method in differential evolution algorithm. The proposed algorithm, IGA, has a very different design concept, as compared with traditional GA. Firstly, it adopts the real coding instead of the genetic string implementation normally found in GAs. Since many complex optimization problems in real-life applications and scientific researches are represented in real variables, and it can obtain faster computation without conversion between binary and decimal system. Also, real coding is very appropriate to NLP problems with better precision and simpleness of operation, and easy to search in complex high-dimensional space. Secondly, it performs a local search procedure in the neighboring area of infeasible individuals while preserving non-dominated individuals in an external set in terms of corresponding feature vectors. Thirdly, IGA uses the concept of the $(\mu+\lambda)$-ES (Costa & Oliveira, 2001). Where μ is the assigned member of individuals, and it is a constant in each generation. The number of generated new individuals is the μ multiplied by a fixed ratio η. Hence, the number of offspring is an invariable value, and it is equal to λ (i.e., $\mu \times \eta$). Experimental results over a set of benchmark problems demonstrate that proposed IGA has better robustness, effectiveness and stableness than other algorithm reported in literature.

The aim of this paper is to present an improved genetic algorithm based on a novel selection strategy. It is designed to exploit the benefit of the existing genetic algorithms so as to be able to deal with various engineering and chemical engineering problems. The rest of this paper is organized as follows: Section 2 introduces related mechanics of GA in details. The proposed algorithm, IGA, is presented in Section 3. Our results are briefly discussed in Section 4 and our conclusions and some paths of future researches are provided in Section 5.

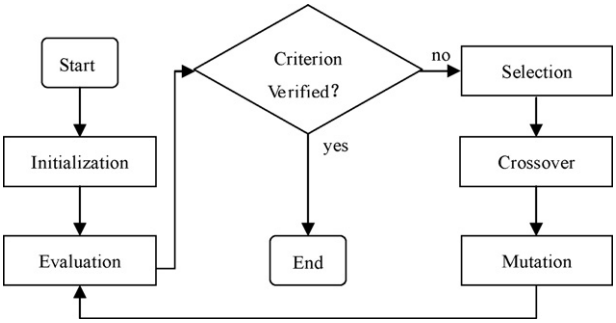

Fig. 1. Genetic algorithm flow chart.

2. Description of GA

Genetic algorithm, initiated by Holland (1975) is one of the most important evolutionary computation techniques. It mimics the process of natural selection (Goldberg, 1989) and starts with artificial individuals (represented by a population of "chromosomes"). GA tries to evolve those individuals that are fitter and, by applying genetic operators (crossover and mutation), it attempts to produce descendants that are better than their parents in terms of a certain quantitative measure. In spite of their diversity, most of them are based on the same iterative procedure.

As a heuristic population-based method, GA is really like a "black box", completely independent from the characteristic of the problem. Fig. 1 presents the classical GA flow chart. An initial population of individuals is generated randomly. Each of these individuals is evaluated in terms of a certain "fitness function" that can "guide" GA to the desired region of the search space. Holland's three genetic operators (selection, crossover and mutation) are the main components to improve the GA's behavior. Selection is the process that mimics the "survival of the fittest" principle in the biological theory of evolution. Firstly, the selection operator assures that individuals are copied to the next generation with a probability associated to their fitness values. Although selection is implemented in a GA as a policy for determining the best candidate individuals that will be presented in the next generation with a higher probability, it does not search the space further, because it just copies the previous candidate individuals. The search results from the creation of new individuals from old ones. Secondly, the crossover operator is implemented in GA by exchanging chromosome segments between two randomly selected chromosomes. Crossover process provides a mechanism to allow new chromosomes to inherit the properties from old ones. Thirdly, mutation is a random perturbation to one or more genes in the chromosomes during evolutionary process. The purpose of the mutation operator is to provide a mechanism to avoid local optima by exploring the new regions of the search space, which selection and crossover could not fully guarantee. The searching process terminates when the predefined criterion is satisfied.

3. The proposed algorithm

This section firstly gives a novel strategy of individual selection, and briefly describes a new local search scheme we have implemented to neighboring areas of infeasible individuals in selection process. Finally, an improved genetic algorithm based on this novel selection scheme is described in details.

3.1. A novel selection scheme

Selection is the first genetic operator in GA. In this process, the better individuals are selected according to their fitness for which they will take part in genetic operations (crossover and mutation). Generally, there are three selection schemes in the GA community, i.e., Roulette wheel, rank-based and tournament, respectively. Firstly, Roulette wheel is the simplest proportional selection scheme. The basic concept of this scheme is that the individuals of the population are considered as slots of the Roulette wheel. Each slot is as wide as the probability for selection of the corresponded chromosome. Using the scaled fitness function, selection probabilities are calculated for each respective individual. This scheme is emphasized to the better individuals in the population and exerts a large pressure on the search process. Secondly, rank-based selection is similar to the proportional selection scheme. Calculating the selection probability is to use individual's rank instead of the fitness function. Thirdly, tournament selection models the tournament games. It uses the fitness function values to evaluate the prospective for reproduction individuals. According to this scheme, we randomly choose some individuals from population to form a tournament size. In a contest, only one individual that has the highest fitness value is the winner.

Based on selection schemes mentioned above, several researchers have attempted to derive good techniques to build better selection schemes for solving constraints problems. Jiménez and Verdegay (1999) proposed a min–max approach to handle constraints. The main idea of this approach is to apply a set of simple rules to decide the selection process. Coello and Montes (2002) proposed a dominance-based selection scheme to incorporate constraints into the fitness function. Their proposal is to try to reach the feasible region of the search space in two ways: finding non-dominated solutions and choosing those with a lower accumulation of constraints violations. Additionally, some individuals are probabilistically selected in the selection. These individuals can be either infeasible or dominated. The reason for this phenomenon is to avoid stagnates and prematurely converges of the proposed algorithm. To some extent, this method can keep the diversity required in the population to ensure that the search progresses. As an extension of the previous work, the number of constraints violations of non-dominated solutions should be considered in our study. Particularly, in such a condition that both function values (e.g., $f(x)$ and $p(x)$) of a solution are the same as those of another solution, then we may consider the number of constraints violations in order to give an appropriate evaluation between two solutions. In account of the equality and inequality constraints, we give the following definitions.

Definition 1 *(The degree of constraints violations).* The degree of constraints violations, $p(x)$, for an infeasible solution x is defined as follows:

$$p(x) = \sum_{i=1}^{m+p} q_i(x)^2, \tag{4}$$

$$q_j(x) = \begin{cases} \max(0, g_j(x)), & 1 \le j \le m \\ |h_j(x)|, & 1 \le j \le p \end{cases}. \tag{5}$$

Definition 2 *(The number of constraints violations).* The number of constraints violations, $s(x)$, for an infeasible solution x is defined as follows:

$$s(x) = \sum_{i=1}^{m+p} Num_j(x), \tag{6}$$

$$Num_j(x) = \begin{cases} 0, & \text{if } q_j(x) \le 0 \\ 1, & \text{otherwise} \end{cases}. \tag{7}$$

Definition 3 *(The representation of feature vector of an individual).* The feature vector $v(x)$ of each individual x consists of three elements: the value of objective function (f), the degree of constraints violations (p), and the number of constraints violations (s). For each random individual x in the population, its value of objective function is calculated by using the original objective function $f(x)$. Hence, $v(x) = (f, p, s)$.

Based on the above concepts, when comparing two individuals in selection, according to the feature vectors of individuals we can have an appropriate evaluation. However, the comparison between two vectors is not the same as the comparison between two ordinary real-values. In general, we can compare two vectors by way of the Pareto partial order $\prec$. We give some related concepts for the set of individuals $Pop = (x_1, x_2, \ldots, x_n)$ as follows:

Definition 4 *(The intensity value of an individual).* Given $x_i \in Pop$, then $IV(x_i)$ represents the number of individuals that individual x_i dominates. We name $IV(x_i)$ as the intensity value of an individual.

$$IV(x_i) = \#\left\{x_j \in Pop \quad \text{and} \quad v(x_i) \prec v(x_j)\right\} \tag{8}$$

where # is a cardinal number of set.

Definition 5 *(The domination count of individual).* Given $x_i \in Pop$, then $DC(x_i)$ represents the number of individuals which dominate the individual x_i.

$$DC(x_i) = \#\left\{x_j | x_j \in Pop \quad \text{and} \quad v(x_j) \prec v(x_i)\right\} \tag{9}$$

We apply two indices (IV and DC) to compare two feasible or infeasible individuals. If IV of individual x_i is better than DC in a population, then the number of individuals that individual x_i dominates is more than the number of individuals which dominate the individual x_i, and vice versa. Obviously, if an individual is feasible, both feature components (p and s) of individual x_i are equal to 0, then comparing two individuals only by way of comparing the feature component, f. Conversely, if an individual is infeasible, both feature components (p and s) of individual x_i are usually unequal to 0, and the value of feature component f may be nonsense or a particular value. Under such circumstances, the feature component f of individual x_i is assigned as an infinite value. When comparing two individuals between x_a and x_b in selection, there are four possible situations:

(1) Both individuals are feasible. In this case, we make a decision in terms of corresponding IV between two individuals.
(2) One is feasible and the other is infeasible. Obviously, the feasible individual wins in terms of the partial order between corresponding feature vectors of two individuals.
(3) Both individuals are infeasible. We may compare two individuals in terms of the partial order between corresponding feature vectors of two individuals.
(4) Both individuals are infeasible and impossible to compare in terms of their feature vectors. An individual with less DC would wins, regardless of its IV.

The pseudo code of our approach is presented in Fig. 2. The following notation is adopted: *CPop* represents the current population, random is a function of generating a random positive integer and Feasibility is a test function which makes a decision with respect to the feasibility of individuals, *IVCompute* and *DCCompute* are applied to compute IV and DC of an individual, respectively.

```
function selection(CPop)
  begin
    i = random( )
    j = random( )
    if (Feasibility(CPop(i)) == true and Feasibility(CPop(j)) == true)
      IV_i = IVCompute(CPop(i))
      IV_j = IVCompute(CPop(j))
      if IV_i > IV_j
         winner = CPop(i)
      else
         winner = CPop(j)
      end
    else
       [v_i, v_j] = FeatureCompute(CPop(i), CPop(j))
    if v_i ≺ v_j
       winner = CPop(i)
    else if v_j ≺ v_i
       winner = CPop(j)
    else
       DC_i = DCCompute(CPop(i))
       DC_j = DCCompute(CPop(j))
       if DC_i < DC_j
         winner = CPop(i)
       else if DC_i < DC_j
         winner = CPop(j)
       end
     end
   end
end
```

Fig. 2. The representation of pseudo code for a novel selection process.

3.2. A new local search (LS) process embedded in selection

In traditional GA, some infeasible individuals would be directly discarded when comparing two individuals in selection. However, for a number of optimization problems in real world, some infeasible individuals may be helpful to effectively find non-dominated individuals since they are located in neighboring areas of feasible individuals. For this reason, we design a local search scheme in which neighboring areas of infeasible individuals is to be searched further with the aim of increasing the diversity of the individuals. In particular, this local search procedure works by generating a temporary subset (denoted by *TS*) of individuals with a modified mutation method in differential evolution. Individuals located in the *TS*, whose size *LS* is an assigned value, are considered as "firm candidates". Each one of these firm candidate individuals (x_i) is included in the *ES* (an external set of individuals) by using Pareto-dominance relations (Gil, Márquez, Baños, Montoya, & Gómez, 2007). The local search procedure is described as follows:

Step 1. Select a random individual x_j from *ES*, and compute the similarity degree between $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \ldots, x_{jn})$, according to the following formula:

$$S_d(x_i, x_j) = 1 - \frac{\sum_{k=1}^{n} x_{ik} \oplus x_{jk}}{n} \tag{10}$$

where $\oplus$ is an exclusive or operator. If x_{ik} is very close to x_{jk} (i.e., the absolute difference between x_{ik} and x_{jk} is less than a prescribed number σ), then the result of performing the operator is 0. Otherwise, its result is 1.

Step 2. If $S_d > \delta$ (δ is a prescribed number), then go to step 3. Otherwise, go to step 1.

Step 3. Performing the modified mutation to generate better solutions.

$$x_{ik}^{\text{new}} = x_{ik} + F(x_{ik} - x_{jk}) \tag{11}$$

where *F* is dynamic mutation parameter. Generate *LS* random numbers *F*, then get a subset $TS = (x_i^1, x_i^2, \ldots, x_i^{LS})$ according to Eq. (11).

Step 4. If $x_i^t \prec x_j$ using their feature vectors, $t \in (1,2,\ldots,LS)$, then replace x_j by x_i^t. Otherwise, if x_i^t is better than any solution in *ES* by using their feature vectors, then replace the dominated individual with x_i^t. Otherwise, if x_i^t is better than any solution in *CPop* using their feature vectors, then replace the dominated individual with x_i^t.

3.3. Summary of the IGA

In our study, we incorporate the discrete crossover and Gaussian mutation (Sarker, Liang, & Newton, 2002) into the IGA. The proposed algorithm is described as follows:

Step 1. Form an initial population of q individuals randomly. The size of external set, *ES*, is arbitrary.
Step 2. Collect non-dominated individuals into *ES* from the population.
Step 3. Select μ individuals from the current population randomly. Perform Gaussian mutation and discrete crossover to generate $\mu \times \eta$ new individuals.
Step 4. Perform selection operator for new individuals. Select an individual into *ES*, and perform the local search process if another compared individual is infeasible.
Step 5. Execute repeatedly step 3 and step 4 until getting n individuals, and then replace old population.
Step 6. Go to step 2 if the termination criteria are not met.

In step 2, although the IGA collects a fixed number of non-dominated solutions into *ES*, the size of the external storage is not restricted. In step 3, we control the number of individuals from the current population so as to decrease the computation time of performing these operators (mutation and crossover). Hence the total time to run IGA is under control. To search better individuals in selection process, we execute the local searching process in neighboring area of infeasible individual in step 4.

In proposed algorithm, we adopt the real number representation instead of the binary string implementation normally found in GAs. There have been a number of different schemes of real coding so far (Zhou et al., 2003). For example, simulated binary crossover (denoted by SBX), blend crossover (denoted by BLX), unimodal normal distribution crossover (denoted by UNDX) and simplex crossover (denoted by SPX).

4. Numerical experiment

Many engineering and chemical engineering problems are presented in simple mathematical formulation. In the following experiments, five benchmark optimization problems *P*1–*P*5 (Zhou, Li, Wang, & Kang, 2003) and *P*6 (Costa & Oliveira, 2001) chosen from the chemical engineering area will be applied to show the way in which the proposed algorithm works, and those are all high-dimensional optimization functions with multiple inequality or equality constraints. These benchmark problems have been previously solved by applying a variety of other algorithms. The initial population size, q, is set to 200, which is a common value found in many GAs experiments. The maximum number of generations is 2500. The size of *TS* is 30. The selected number of individuals from the current population, μ, is 20, and the ratio of offspring and parents, η, is 2. Crossover rate is equal to 0.6. For each variable, the mutation rate is $1/n$, where n is the total number of variables.

$$\begin{aligned} \textit{Test P1} \quad \textit{Min} \quad & f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3 \\ s.t \quad & x_4 - x_3 + 0.55 \geq 0, \quad -x_4 + x_3 + 0.55 \geq 0, \\ & 10^3 \sin(-x_3 - 0.25) + 10^3 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0, \\ & 10^3 \sin(x_3 - 0.25) + 10^3 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0, \\ & 10^3 \sin(x_4 - 0.25) + 10^3 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0, \\ & 0 \leq x_i \leq 1200 (i = 1, 2), \quad -0.55 \leq x_i \leq 0.55 (i = 3, 4). \end{aligned}$$

Test P2 $Min \quad f(x) = e^{x_1 x_2 x_3 x_4 x_5}$

$$s.t \quad \sum_{i=1}^{5} x_i^2 - 10 = 0, \quad x_2 x_3 - 5x_4 x_5 = 0, \quad x_1^3 + x_2^3 + 1 = 0,$$
$$-2.3 \le x_i \le 2.3\,(i = 1, 2), \quad -3.2 \le x_i \le 3.2\,(i = 3, 4, 5).$$

Test P3 $Min \quad f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6 x_7 - 10x_6 - 8x_7$

$$s.t \quad 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \ge 0, \quad 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \ge 0,$$
$$196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \ge 0, \quad -4x_1^2 - x_2^2 + 3x_1 x_2 - 2x_3^2 - 5x_6 + 11x_7 \ge 0,$$
$$-10.0 \le x_i \le 10.0\,(i = 1, 2, \ldots, 7).$$

Test P4 $Min \quad f(x) = x_1 + x_2 + x_3$

$$s.t \quad -1 + 0.0025(x_4 + x_6) \le 0, \quad -1 + 0.0025(x_5 + x_7 - x_4) \le 0,$$
$$-1 + 0.01(x_8 - x_5) \le 0, \quad -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \le 0,$$
$$-x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \le 0, \quad -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \le 0,$$
$$100 \le x_1 \le 10^4, \quad 10^3 \le x_i \le 10^4\,(i = 2, 3), \quad 10 \le x_i \le 10^3\,(i = 4, 5, \ldots, 8).$$

Test P5 $Min \quad f(x) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 +$
$+5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$

$$s.t \quad 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \ge 0, \quad -10x_1 + 8x_2 + 17x_7 - 2x_8 \ge 0,$$
$$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \ge 0, \quad -x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 \ge 0,$$
$$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \ge 0, \quad -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \ge 0.$$
$$-10.0 \le x_i \le 10.0\,(i = 1, 2, \ldots, 10).$$

Test P6 $Min \quad f(x) = (x_1 - 1)^2 + (x_2 - 1)^2 + (x_3 - 3)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 + (x_6 - 1)^2 - \ln(x_7 + 1)$

$$s.t \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 - 5 \le 0, \quad x_1^2 + x_2^2 + x_3^2 + x_6^2 - 5.5 \le 0, \quad x_1 + x_4 - 1.2 \le 0,$$
$$x_2 + x_5 - 1.8 \le 0, \quad x_3 + x_6 - 2.5 \le 0, \quad x_1 + x_7 - 1.2 \le 0, \quad x_2^2 + x_5^2 - 1.64 \le 0, \quad x_3^2 + x_6^2 - 4.25 \le 0,$$
$$x_3^2 + x_5^2 - 4.64 \le 0, \quad x_i \ge 0\,(i = 1, 2, 3), \quad x_i \in \{0, 1\}\,(i = 4, 5, 6, 7).$$

To evaluate the performance of the IGA, we compare IGA with other three kinds of algorithms. These algorithms are Pareto strength evolutionary algorithm (Zhou et al., 2003), stochastic ranking algorithm (Runarason & Yao, 2000) and homomorphous mapping method (Koziel & Michalewicz, 1999), respectively. Those are denoted by ZW, RY and KM, respectively. Each problem is executed 20 times, and the number of iteration is 1000 each time. We calculate the best solution, worst solution and mean solution by means of having a statistical computation for each running of the IGA and other algorithms. The experimental results obtained by other algorithms are provided in the literature (Zhou et al., 2003). Simulations are performed in MATLAB 7.0 with a 2 GHz Pentium Pc. The experimental results are given in Tables 1–3.

Table 1 shows the available optimal solutions and corresponding global minimum for all test problems. Table 2 shows the comparison of the test results between the IGA and the known three algorithms. The results indicate that IGA reports a slight improvement of the precision in localizing the global solution with a similar reliability, but presents a higher probability to get the best solution with a reduced number of trials. For problems *P*2, *P*4, *P*5, and *P*6, we can see that best solutions obtained by IGA are better than other three algorithms since those solutions are much close to the true optimal solutions. For problems *P*1, and *P*3, our algorithm finds a similar solution to ZW in terms of all three criteria: the best, mean, and worst results. Both algorithms performed well. The simulation results with *P*1 show that no feasible solutions are obtained at the end of the iterations using the KM. Note that the IGA can find the best solution with *P*4 for 18 times. We can observe a very robust behavior of the proposed algorithm with *P*4. Simulation results with other test problems also show that the proposed algorithm is very effective.

To examine the efficiency of the IGA further, we execute the IGA and other three algorithms for 100 runs, respectively. At each run, if the feasible solution x satisfies that $|f(x) - f^*| < 1E\text{-}3$ (f^* represents the global minimum), then simulation is considered as successful. Simulation results by using different algorithms for five test problems are shown in Table 3, where Sr represents the successful rate, and time and number of iterations are the average value of finding optimal solution and corresponding number of iterations, respectively.

Table 1
The available global optimal solutions and corresponding global minimum with all test problems.

Design variables	Optimal solution					
	*P*1	*P*2	*P*3	*P*4	*P*5	*P*6
x_1	679.9453	−1.717143	2.330499	579.3167	2.171996	0.2
x_2	1026.067	1.595709	1.951372	1359.943	2.363683	1.280624
x_3	0.1188764	1.827247	−0.477541	5110.071	8.773926	1.954483
x_4	−0.396234	−0.763641	4.365726	182.0174	5.095984	1
x_5	–	−0.763645	−0.624487	295.5985	0.9906548	0
x_6	–	–	1.038131	217.9799	1.430574	0
x_7	–	–	1.594227	286.4162	1.321644	1
x_8	–	–	–	395.5979	9.828726	–
x_9	–	–	–	–	8.280092	–
x_{10}	–	–	–	–	8.375927	–
Global minimum	5126.4981	0.0539498	680.63006	7049.3307	24.306209	3.557463

Table 2
The comparison among IGA, ZW, RY and KM (20 independent runs).

Test problem	Method	Best solution	Best error (%)	Mean solution	Mean error (%)	Worst solution	Worst error (%)
*P*1	IGA	5126.498272	0.00000	5126.528315	0.00059	5127.159164	0.01290
	ZW	5126.59811	0.00195	5126.52654	0.00055	5127.15641	0.01284
	RY	5126.497	0.00002	5128.881	0.04648	5142.472	0.31159
	KM	–		–	–	–	–
*P*2	IGA	0.053949816	0.00003	0.053950126	0.00060	0.053952358	0.00474
	ZW	0.053949831	0.00006	0.053950257	0.00085	0.053972292	0.04169
	RY	0.053957	0.01335	0.057006	5.66490	0.216915	302.06822
	KM	0.054	0.09305	0.064	18.62880	0.557	932.44127
*P*3	IGA	680.6301361	0.00001	680.6302536	0.00003	680.6304037	0.00005
	ZW	680.6300573	0.00000	680.6300573	0.00000	680.6300573	0.00000
	RY	680.630	0.00001	680.656	0.00381	680.763	0.01953
	KM	680.91	0.04113	681.16	0.07786	683.18	0.37464
*P*4	IGA	7049.3307150	0.00000	7049.3307510	0.00000	7049.3310130	0.00000
	ZW	7049.2480205	0.00117	7051.2874292	0.02776	7058.2353585	0.12632
	RY	7054.316	0.07072	7559.192	7.23276	8835.665	25.34048
	KM	7147.9	1.39828	8163.6	15.80674	9659.3	37.02436
*P*5	IGA	24.30620913	0.00000	24.30620926	0.00000	24.30621013	0.00000
	ZW	24.306209068	0.00000	24.325487652	0.07932	24.362999860	0.23365
	RY	24.307	0.00325	24.374	0.27890	24.642	1.38150
	KM	24.620	1.29099	24.826	2.13851	25.069	3.13826
*P*6	IGA	3.557471	0.00022	3.643247	2.41138	3.731238	4.88480
	ZW	3.559345	0.05290	4.245433	19.33878	4.632134	30.20891
	RY	3.762318	5.75845	4.678341	31.50779	5.198211	46.12129
	KM	3.912343	9.97565	5.012315	40.89577	6.783529	90.68446

Of the six test problems, there are three successful rates of the IGA, which is larger (*P*4, *P*5 and *P*6) than those of the proposed algorithm (especially in *P*4), and only one (*P*3) is equal to that of ZW. The successful rate of the proposed algorithm is a little smaller than that of ZW in tests *P*1 and *P*2. Although the IGA does not outperform the ZW with *P*1 and *P*2 in terms of Sr, both algorithms is very close in terms of running time, and IGA is much superior to the other two algorithms, i.e., RY and KM. Note that the successful rates of six test problems are all above 80% using IGA, and obtained optimal solutions are all very close to the real optimal solutions. From Table 3, it can be seen that the IGA significantly outperforms the other two algorithms (RY and KM) with six test problems from the viewpoints of the Sr, time and f^*.

The number of iteration to find optimal solution indicates computational efficiency for an evolutionary optimization technique. This metric is normally adopted in evolutionary computation since the number of iteration is independent of the hardware used for the experiments. Furthermore, from Table 3, it also could be seen that the IGA is able to reach all global optima of the six test functions during the less number of iterations, whereas the other three algorithms may not be able to reach them.

5. Discussion of results

Since the IGA is a stochastic algorithm based on population, that is, each separate run of the IGA could result in a different result. It is desired to know the performance of stability one may obtain if the IGA is adopted in engineering and chemical engineering area. To some extent, it is directly relating to the quality of worst solution. We conduct the worst-case analysis by comparing the worst solution and optimum solution. Table 2 summarizes the experimental results we obtained after 20 independent runs. For *P*4 and *P*5, worst error is 0, which means our algorithm was capable of finding better solutions than other algorithms. For *P*3, worst error is 0.00005 which is very approximate to 0, which mean the stability of obtaining best solution is also very good. For *P*1, *P*2, and *P*6, their worst errors are 0.01290, 0.00474 and 4.88480, respectively. In comparison with the results obtained by other algorithms, these results obtained by our approach are significantly better for all problems. The analysis from the worst solution is very useful when the user requests a guarantee for the quality of the optimum solution.

There are a few things about our approach that deserve to be mentioned. Firstly, we have empirically shown the feasibility of using the feature vector to compare individuals. The feature vector of individual plays an important role in our approach, since it is responsible for selecting better individual between different individuals. When comparing two vectors, we can have four possible situations. During these situations, some individuals are selected according to the Pareto partial order. These individuals can be either infeasible or dominated. Our approach tries to

Table 3
The comparison among IGA, ZW, RY and KM (100 independent runs).

Test problem	Method	Sr (%)	Time (s)	f^*	Number of iterations
*P*1	IGA	80	40.62	5126.49821	318
	ZW	100	45.35	5126.49813	876
	RY	30	60.78	5127.52751	1213
	KM	–	–	–	–
*P*2	IGA	83	31.04	0.0539501	603
	ZW	95	29.26	0.0539498	1232
	RY	60	40.65	0.0053956	1550
	KM	57	35.27	0.0541862	1479
*P*3	IGA	95	43.35	680.6300573	703
	ZW	95	40.51	680.6300573	910
	RY	80	50.63	680.6300913	1078
	KM	20	45.36	683.108472	1802
*P*4	IGA	100	35.26	7049.3307	701
	ZW	46	39.48	7049.3648	1323
	RY	15	41.62	7159.4294	1832
	KM	26	40.14	7084.9617	2312
*P*5	IGA	100	50.89	24.3062091	523
	ZW	98	56.24	24.3062091	1283
	RY	90	64.61	24.3062091	1658
	KM	51	60.78	24.3068061	1753
*P*6	IGA	100	45.34	3.557471	504
	ZW	85	50.32	3.559345	1123
	RY	70	57.36	3.762318	1453
	KM	50	58.21	3.912343	2097

find non-dominated individuals and choosing those with a lower accumulation of constraint violation. It is worth emphasizing that traditional evolutionary optimization techniques normally cannot be used directly to handle constraints because their emphasis is to drive the GAs towards the feasible region, but not necessarily to the global optimum.

A significant difference of our approach with respect to other algorithms (ZW, RY and KM) is the way in which a local search process is performed at the neighboring areas of infeasible individuals. The reason for this is to avoid that our algorithm stagnates and prematurely converges to a local optimum. This mechanism is thus responsible for keeping the diversity required in the population to move the search towards the global optimum of the problem.

Further, throughout our study, the IGA is faster for several of the test problems, but for other cases the ZW is faster. It is important to emphasize that the performance of any evolutionary algorithm for constrained optimization is determined by the constraint-handling technique used, as well as the natures of problems. This implies that the any evolutionary algorithm has not absolutely superiority over other methods. Further work is needed to understand that how the natures of problems exert influence on different constraint-handling technique used in terms of starting population, inequality constraints and equality constraints, crossover rate and mutation rate.

In summary, the proposed approach performed well in different NLP test problems both in terms of the quality of the solutions found and in terms of the successful rate. IGA has higher efficiency in solving NLP problems for reaching the near-optimal solutions with less error. It shows that our approach has a significant universality to some extent. In addition, it is also noted that the stability of any algorithm depends on the parameter selection to some extent. Currently the parameters can only be chosen heuristically, and more study is needed to investigate on this problem.

6. Conclusion

In this paper, an improved genetic algorithm based on a novel selection scheme was introduced for nonlinear programming problems. In selection process, each individual is represented as a three-dimensional feature vector. The proposed algorithm improves the selection strategy of individuals. We can distinguish excellent individuals through two indices according to Pareto partial order. On the other hand, a new local search process is embedded in selection process with the aim to improve the quality of the solutions further. The performance of this new algorithm has been compared with three well-known algorithms on five single-objective benchmark problems. In most cases, the proposed algorithm outperforms other approaches reported in the literature in terms of a set of metrics.

For future work, we intend to examine the proposed algorithm and improve its performance for single-objective optimization problems and other combinatorial optimization problem. In addition, we are interested in the hybridation of IGA and other heuristic population-based search method such as particle warm optimization, ant colony optimization, and simulated annealing. The combination of the IGA and other heuristic method should offer the advantages of mutual optimization methods while offsetting their disadvantages.

Acknowledgement

This paper is supported by the National Natural Science Foundation of China (Nos. 60803049 and 60472060).

References

Courant, R. (1962). *Calculus of variations and supplementary notes and exercises*. Supplementary notes by Kruskal, M. and Rubin, R., revised and amended by Moser, J. New York: New York University.

Fiacco, A. V., & McCormick, G. P. (1968). *Nonlinear programming: sequential unconstrained minimization technique*. New York: John Wiley and Sons.

Gupta, R. K., Bhunia, A. K., & Roy, D. (2009). A GA based penalty function technique for solving constrained redundancy allocation problem of series system with interval valued reliability of components. *Journal of Computational and Applied Mathematics*, *232*, 275–284.

Shahpar, A. H., Aashtiani, H. Z., & Babazadeh, A. (2008). Dynamic penalty function method for the side constrained traffic assignment problem. *Applied Mathematics and Computation*, *206*, 332–345.

Sarker, R., & Newton, C. (2002). A genetic algorithm for solving economic lot size scheduling problem. *Computers and Industrial Engineering*, *42*, 189–198.

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

He, Q., & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, *20*, 89–99.

Jiménez, F., & Verdegay, J. L. (1999). Evolutionary techniques for constrained optimization problems. In *Seventh European congress on intelligent techniques and soft computing (EUFIT'99)*. Berlin: Springer.

Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, *16*, 193–203.

Chootinan, P., & Chen, A. (2006). Constraint handling in genetic algorithms using a gradient-based repair method. *Computers and Operations Research*, *33*, 2263–2281.

Sundaram, A., & Venkatasubramanian, V. (1998). Parametric sensitivity and search-space characterization studies of genetic algorithms for computer-aided polymer design. *Journal of chemical Information and Computer Science*, *38*, 1177–1191.

Carrillo-Ureta, G. E., Roberts, P. D., & Becerra, V. M. (2001). Genetic algorithms for optimal control of beer fermentation. In *Proceedings of the 2001 IEEE international symposium on intelligent control* (pp. 391–396).

Li, Y., Ida, K., & Gen, M. (1997). Improved genetic algorithm for solving multiobjective solid transportation problem with fuzzy numbers. *Computers and Industrial Engineering*, *33*, 589–592.

Park, B. J., Choi, H. R., & Kim, H. S. (2003). A hybrid genetic algorithm for the job shop scheduling problems. *Computers and Industrial Engineering*, *45*, 597–613.

Zhou, Y. R., Li, Y. X., Wang, Y., & Kang, L. S. (2003). A Pareto strength evolutionary algorithm for constrained optimization. *Journal of Software (in china)*, *14*, 1243–1249.

Runarason, T., & Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, *4*, 284–294.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, *7*, 19–44.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

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

Gil, G., Márquez, A., Baños, R., Montoya, M. G., & Gómez, J. (2007). A hybrid method for solving multi-objective global optimization problem. *Journal of Global Optimization*, *38*, 265–281.

Sarker, R., Liang, K. H., & Newton, C. (2002). A new multiobjective evolutionary algorithm. *European Journal of Operational Research*, *140*, 12–23.

Costa, L., & Oliveira, P. (2001). Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers and Chemical Engineering*, *25*, 257–266.