Measurement 115 (2018) 27–38

Contents lists available at ScienceDirect

Measurement

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

# A novel genetic algorithm based method for solving continuous nonlinear optimization problems through subdividing and labeling

Majid Esmaelian[a], Madjid Tavana[b,c,*], Francisco J. Santos-Arteaga[d], Masoumeh Vali[e]

[a] Department of Management, University of Isfahan, Isfahan, Iran
[b] Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, USA
[c] Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, Paderborn, Germany
[d] Faculty of Economics and Management, Free University of Bolzano, Bolzano, Italy
[e] Department of Industrial Management, Persian Gulf University, Bushehr, Iran

ARTICLE INFO

*Keywords:*
Genetic algorithm (GA)
Subdividing labeling
Nonlinear optimization
Fitness function

ABSTRACT

We introduce a novel method called subdividing labeling genetic algorithm (SLGA) to solve optimization problems involving $n$ – dimensional continuous nonlinear functions. SLGA is based on the mutation and crossover operators of genetic algorithms, which are applied on a subdivided search space where an integer label is defined on a polytope built on the $n$ – dimensional space. The SLGA method approaches a global optimal solution by reducing the feasible search region in each iteration. One of its main advantages is that it does not require computing the derivatives of the objective function to guarantee convergence. We apply the SLGA method to solve optimization problems involving complex combinatorial and large-scale systems and illustrate numerically how it outperforms several other competing algorithms such as Differential Evolution even when considering problems with a large number of elements.

## 1. Introduction

Finding the optimal solution constitutes one of the main objectives when undertaking practical projects. In this regard, optimization problems are a central element of several disciplines such as computational chemistry and biology, computer sciences, structural optimization frameworks, economics, operational research, and engineering design and control [3,7,10,27,30,31,37,40]. Global optimization problems have attracted a considerable amount of attention in the last decade. In particular, the implementation of nonlinear optimization environments is widely extended in the measurement literature, encompassing a large variety of problem settings whose solutions are determined by the behavior of nonlinear equations [9,14,28].

Evolutionary Computation (EC) is a general term that encompasses different types of optimization algorithms inspired by the evolving capacity of organisms that are able to adapt to their environment [22]. Usually categorized as EC algorithms (also known as Evolutionary Algorithms (EAs)) are the domains of genetic algorithms [16], evolutionary programming, evolutionary strategies, and genetic programming [13]. Compared to other optimization methods, Genetic Algorithms (GAs) behave as auto-adaptive global searching systems that search for the best solution more effectively by simulating biological evolution and the fittest principle of natural environments [16].

In other words, GAs are heuristic search techniques that mimic natural evolutionary processes such as selection, crossover and mutation. Despite the substantial improvements achieved in the latter years, the use of GAs in optimization problems remains quite time consuming, with further efforts being directed at overcoming this limitation. For instance, Storn and Price [32,33] introduced the Differential Evolution (DE) algorithm as a simple but powerful population-based stochastic search technique for solving global optimization problems. This technique constitutes one of the main benchmarks used to compare the performance of the Subdivision Labeling Genetic Algorithm (SLGA) introduced in the current paper.

This novel method is designed to solve optimization problems involving $n$ – dimensional continuous nonlinear functions. SLGA is based on the mutation and crossover operators of genetic algorithms, which are applied on a subdivided search space where an integer label is defined on a polytope built on the $n$ – dimensional space. The SLGA method approaches a global optimal solution by reducing the feasible search region per iteration.

Among its main advantages, we must highlight the fact that SLGA does not require computing the derivatives of the objective function to

* Corresponding author at: Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, USA.
*E-mail addresses:* M.Esmaelian@ase.ui.ac.ir (M. Esmaelian), tavana@lasalle.edu (M. Tavana), fsantosarteaga@unibz.it (F.J. Santos-Arteaga), m.vali@mehr.pgu.ac.ir (M. Vali).
*URL:* http://tavana.us/ (M. Tavana).

http://dx.doi.org/10.1016/j.measurement.2017.09.034
Received 1 June 2017; Received in revised form 25 July 2017; Accepted 19 September 2017
Available online 10 October 2017
0263-2241/ 

guarantee convergence. We apply the SLGA method to optimize the De Jong functions as well as to solve nonlinear constrained and unconstrained optimization problems with continuous, discrete and mixed variables. Experimental results show that SLGA has good performance when considering complex combinatorial and large-scale systems and illustrate how it outperforms several other competing algorithms by reducing the number of generations within the solution space in problems with a large number of elements.

The paper is organized as follows. The next section surveys the related literature. The proposed approach is introduced in Section 3, while several experimental results obtained by applying the SLGA method to continuous nonlinear optimization problems are presented in Section 4. The performance of SLGA is compared with that of several similar approaches in Section 5. Some concluding remarks are provided in Section 6.

## 2. Related works

Since John Holland [16] popularized GAs in the early 1970s, they have been applied to a wide array of problems, such as sensor network optimization [2,6], the traveling salesman problem [2,36], data mining [13,29], network expansion [23], and several types of chance constrained-based problems, including fractional programming [35], data envelopment analysis [34] and system reliability [8].

However, the global optimization of high-dimensional problems remains a major challenge of evolutionary computation and has been the subject of a substantial amount of studies. For example, Chu et al. [5] introduced a novel strategy to treat the complexity caused by high dimensionalities in randomization-based evolutionary algorithms. Their strategy featured a slope-based searching kernel and a scheme to maintain the capacity of the population to search over the entire space. Jiao et al. [19] defined a self-adaptive selection method that combined feasibility with multi-objective problem techniques. Wong et al. [38] proposed three methods to analyze the effect of locality and the synergy between temporal and spatial locality. Jahn [20] presented a multi-objective search algorithm based on a subdivision technique to obtain global solutions for multi-objective constrained optimization problems with non-continuous objective or constraint functions.

Regarding the implementation of GAs to solve different types of optimization problems, we must highlight the following studies. Qian et al. [36] proposed a GA to deal efficiently with constrained integer programming problems. Zhang et al. [41] introduced a GA that used the vertex label information of the individual simplex to design selection, crossover and mutation operators. Wright et al. [39] developed a dynamical system model of a GA that used gene pool crossover, proportional selection, and mutation. Finally, Cao et al. [3] presented a mixed-variable evolutionary programming technique for solving nonlinear optimization problems that proved to be superior to other techniques in terms of solution quality and algorithm robustness [4,15,29].

## 3. Subdividing labeling genetic algorithm

We propose a novel approach designed to optimize simple-bounded continuous functions, denoted by $f(x_1,x_2,...,x_n)$, in which $a_i \leqslant x_i \leqslant b_i (i = 1,2,...,n)$, for some constants $a_i$ and $b_i$. The main aim of the suggested approach is to find the optimum point based on a modified version of a genetic algorithm. In order to preserve the notation used in genetic-based methods, we refer to $f$ as the fitness function throughout the paper.

Fig. 1 provides an overview of the proposed subdividing labeling method based on genetic algorithms.

1. Similarly to standard genetic algorithms, the proposed approach starts from an initial population $P$, containing the vertices of the polytope built using the boundaries of $x_i$. The initial population delimits the search space of the algorithm by defining explicitly its bounds.
2. An initial best point $S$ is selected from this population using an elitism mechanism. For instance, if the goal is to minimize the fitness function, then the best point is the one leading to the lowest fitness value.
3. Then, a mutation operator is implemented in order to generate offspring from the initial population $P$. The main aim of this operator is to reduce the search space so as to approach the optimum point through the different iterations.
4. The resulting $n$ – dimensional points are labeled so as to obtain a completely labeled polytope. The labeling process provides directional guidelines describing the evolution of the algorithm through the different dimensions as it converges towards the optimum.
5. A crossover operation is then performed between the best point S and the completely labeled polytope. In particular, the crossover operation defines a weighted average between the best point S and the vertices and center point of the adjacent sides of the completely labeled polytope. As a result, a new population is generated consisting of the offspring obtained via crossover.
6. The best $n$ – dimensional offspring is selected from the population following an elitism mechanism, i.e. according to the value of the fitness function $f$. The best offspring obtained from the new population is denoted by $S'$.
7. If the algorithm meets the precision requirements, then the process ends and $S'$ is selected as the best (near-optimum) point.
8. However, if the algorithm does not meet the precision requirements, a new population is generated by subdividing the search space and implementing the mutation operator on the vertices of the new polytope.
9. The algorithm proceeds by labeling the new mutated polytope and repeating the above crossover operation between the best point $S'$ and the new completely labeled polytope so as to obtain a new potentially optimum point.

### 3.1. Steps of the proposed algorithm (SLGA)

Let $f(x_1,x_2,...,x_n)$ be a $n$ – dimensional simple-bounded continuous function with constraints of the type $a_i \leqslant x_i \leqslant b_i$, for $i = 1,2,...,n$. The proposed subdividing labeling optimization algorithm is defined through the following formal steps:

**Step 1 (initial population):** Set $P(0)$ as the initial population to the vertices of the polytope made by the boundaries $x_i = a_i$ and $x_i = b_i$, for $i = 1,2,...,n$. That is, the initial population is given by the intersection points obtained from drawing the lines $x_i = a_i$ and $x_i = b_i$, for $i = 1,2,...,n$. For example, consider the following test minimization problem, which will be further developed in Section 4.2

$$f(x) = x_1^2 + x_2^2 - 18\cos x_1 - 18\cos x_2$$
$$-1 \leqslant x_1 \leqslant 1, i = 1,2 \quad (1)$$

In this case, the search space is delimited by drawing the lines $x_1 = -1$, $x_1 = 1$, $x_2 = -1$ and $x_2 = 1$. The initial population is therefore given by the intersection points defined by the four lines, as Fig. 2 illustrates.

**Note:** When $n = 2$, the size of the initial population is equal to four, i.e. $P = [(a_1,a_2),(a_1,b_2),(b_1,a_2),(b_1,b_2)]$. Generally, the size of the initial population equals $2^n$.

**Step 2 (labeling):** Assume that the algorithm is searching for the point $x$ that makes the continuous function $f(x_1,x_2,...,x_n)$ achieve its minimum. The necessary and sufficient extreme point condition is that the gradient at $x$ equals zero, i.e. $\nabla f(x) = 0$. At the same time, $x \in R^n$ is a fixed point of $g$: $R^n \to R^n$ if $g(x) = x$. Thus, both optimality conditions can be related as follows, $g(x) - x = \nabla f(x) = 0$.

Assume now that the domain of $f(x_1,x_2,...,x_n)$ is of the type $a_1 \leqslant x_1 \leqslant b_1,...a_n \leqslant x_n \leqslant b_n$ and that it is divided into $2^n$ polytope vertices in the first iteration. As a result, we have $n$ groups of straight lines of the type $x_1^1 = a_1 + k_1^1 h_1^1, x_2^1 = a_2 + k_2^1 h_2^1, ..., x_n^1 = a_n + k_n^1 h_n^1$, where

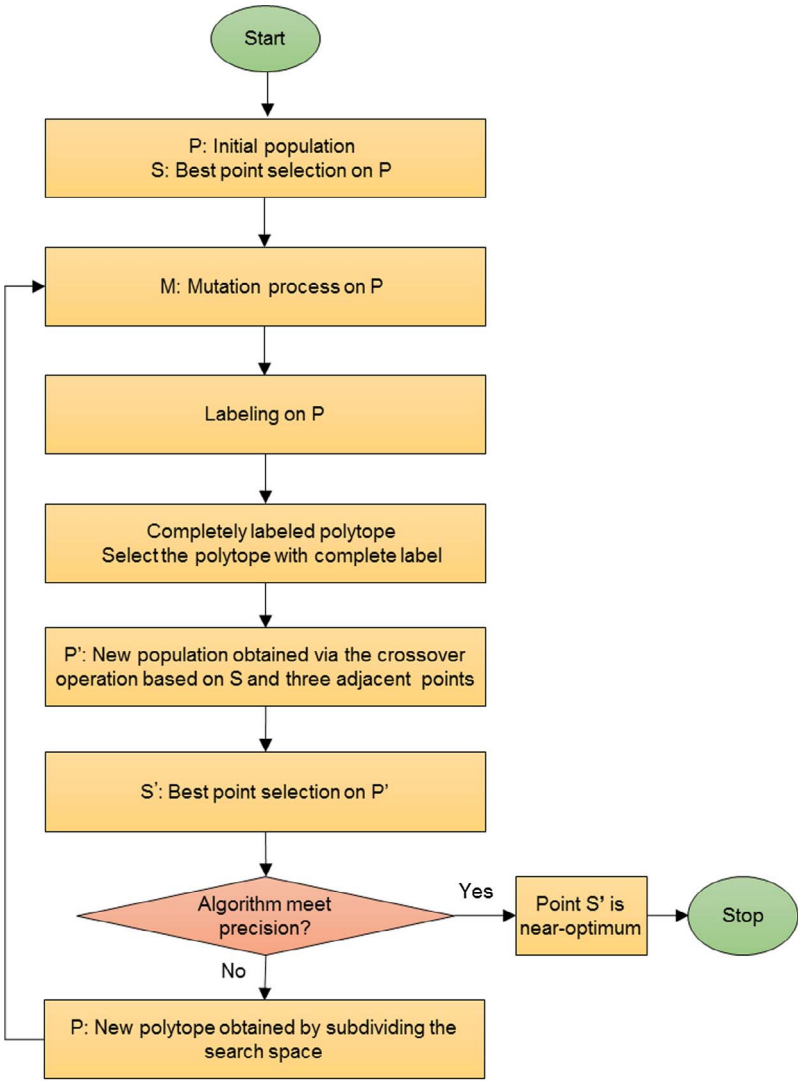


**Fig. 1.** Proposed SLGA algorithm flowchart.

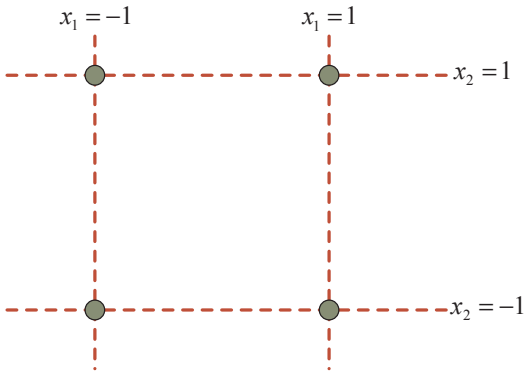


**Fig. 2.** Representation of the initial population.

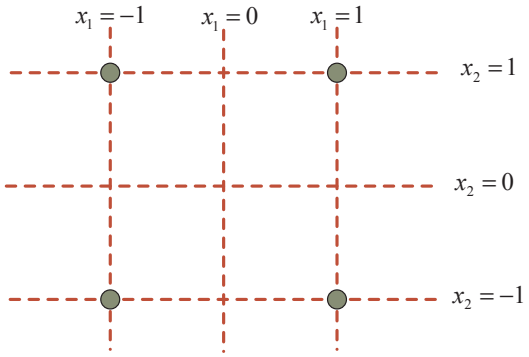


**Fig. 3.** Subdividing process.

$h_i^1 = b_i - a_i$ are positive quantities and $(k_1^1, k_2^1, \ldots, k_n^1)$ non-negative integers defining the relative coordinates of the points in the initial iteration.

In each iteration of the algorithm, the domain of $x_i^j$ is reduced as the search space is subdivided. Given the above minimization example, the initial subdivision of the search space is determined by $h_1^1 = 2$ (i.e. the distance between $x_1 = -1$ and $x_1 = 1$) and $h_2^1 = 2$ (i.e. the distance between $x_2 = -1$ and $x_2 = 1$). Thus, assuming that $k_1^1 = k_2^1 = \frac{1}{2}$, we have $x_1^1 = -1 + k_1^1 h_1^1 = -1 + \frac{1}{2} * 2 = 0$ and $x_2^1 = -1 + k_2^1 h_2^1 = -1 + \frac{1}{2} * 2 = 0$.

As Fig. 3 illustrates, the search space will be subdivided in four polytopes.

The new intersection points obtained per iteration after subdividing the space are mutated through $g(X^j) = (g_1(X^j), g_2(X^j), \ldots, g_n(X^j)) = (x_1^j + \alpha_1^{j+1}, x_2^j + \alpha_2^{j+1}, \ldots, x_n^j + \alpha_n^{j+1})$, with $\alpha_i^{j+1} \in [0, \pm h_i^{j+1}]$ and $i = 1, \ldots, n$. The mutation operator reduces its range per iteration at a rate $h_i^{j+1} = \frac{h_i^j}{2}$.

The intersection points are then labeled according to the difference between the mutated point in $g(X^j)$ and their value, $(x_1^j, x_2^j, \ldots, x_n^j)$, as

follows

$$l(x^j) = \begin{cases} 0, & g_1(X^j) - x_1^j \geqslant 0, \ldots, g_n(X^j) - x_n^j \geqslant 0 \\ 1, & g_1(X^j) - x_1^j < 0, g_2(X^j) - x_2^j \geqslant 0 \ldots, g_n(X^j) - x_n^j \geqslant 0 \\ 2, & g_2(X^j) - x_2^j < 0, g_3(X^j) - x_3^j \geqslant 0, \ldots, g_n(X^j) - x_n^j \geqslant 0 \\ \vdots & \\ n, & g_n(X^j) - x_n^j < 0 \end{cases} \quad (2)$$

A polytope with all its intersection points labeled is called a completely labeled polytope. After labeling, we select a completely labeled polytope, which will be used to perform the crossover operation. As can be already inferred from the description of the process, the main purpose of the subdividing and labeling operations is to classify the different areas of the search space, illustrate the direction followed by the algorithm through it and remove those areas that do not contain the best point.

**Step 3 (crossover):** Let the vertices $v_1$, $v_2$ and the center point $v_3$ be the closest vertices to *the best vertex S* in a completely labeled polytope. The crossover operation generates three new points, $S_1$, $S_2$ and $S_3$, as follows

$$S_1 = w_1 v_1 + (1 - w_1) S$$
$$S_2 = w_2 v_2 + (1 - w_2) S$$
$$S_3 = w_3 v_3 + (1 - w_3) S$$

where $w_1$, $w_2$ and $w_3$ determine the weights assigned to the parents $v_1$, $v_2$, $v_3$ and $S$ in the resulting offspring $S_1$, $S_2$ and $S_3$.

**Step 4 (selection):** Set $S'$ to be the best point of the offspring $S_1$, $S_2$ and $S_3$ in terms of the fitness function.

**Step 5:** $S'$ is the best (near-optimum) point of the fitness function if either $h_i^j \to 0$ within iterations or the algorithm meets the desired accuracy requirements; else go to Step 6.

**Step 6 (subdividing and mutation):** After subdividing the search space, the GA aims at improving each point coded by $(x_1^j, x_2^j, \ldots, x_n^j)$ in the $j-$ th iteration through a mutation operator that searches all the points surrounding it within a range determined by $\left(h_i^{j+1} = \frac{h_i^j}{2}\right)$.

For example, the $(x_1^j, x_2^j, \ldots, x_n^j)$ point obtained in the $j-$ th iteration will be mutated as $X^j = (x_1^j + \alpha_1^{j+1}, x_2^j + \alpha_2^{j+1}, \ldots, x_n^j + \alpha_n^{j+1})$ with $\alpha_i^{j+1} \in \{0, \pm h_i^{j+1}\}$ and $i = 1, \ldots, n$. The resulting polytope must be evaluated and labeled as described in Step 2 of the algorithm.

That is, the mutated population is required to label the polytope that will be used in the crossover operation together with the corresponding $S$ value to generate the offspring among which to select the next best point $S'$. Moreover, a new mutation process determined by the range of the next $h_i^j$ will be implemented after subdividing the search space. The resulting labeled polytope together with $S'$ will be used as parents in the subsequent crossover operation.

Note that the mutation and crossover operations act as a neighborhood search mechanism, making the SLGA method resemble a multiple restart local search algorithm. In this regard, it should be highlighted that our local search algorithm works intelligently, since it selects the restart point endowed with the best fitness function value per iteration. Though not a GA per se, the SLGA method uses the main operators of GAs to improve its search capabilities, preventing the limited size of the population considered from restricting its convergence capacity. Table 1 summarizes the notation used to explain the basic features of the SLGA algorithm.

## 4. Performance evaluations

In order to evaluate our proposed approach, we have first considered two test problems in Sections 4.2 and 4.3. Then, in Section 4.4, we describe the implementation of the proposed algorithm to constrained nonlinear optimization problems. In Sections 4.5–4.7 the SLGA is applied to solve specific constrained nonlinear problems.

**Table 1**
The notation used in explaining our algorithm.

| | |
|---|---|
| $a_i, b_i$ | Boundaries of $x$, defining the test area |
| $P$ | Population |
| $n$ | Dimension of the test function |
| $S$ | Best point of the population regarding the fitness function in a completely labeled polytope |
| $v$ | Point of polytope center |
| $w_i$ | Weighting parameters for the crossover operator |

### 4.1. Parameter initialization

In all the experiments of this paper, we have assigned a value of $w_1 = w_2 = w_3 = 0.5$ to the weight parameters of the crossover operation, as well as the following tolerance limits

- $\varepsilon_c = 0.001$ Z convergence tolerance for the fitness function $F$: $|F_{j+1} - F_j| < \varepsilon_c$;
- $\varepsilon_i = 0.01$ Z integer tolerance for the optimum point $x \in R^n$: $|x - round(x)| < \varepsilon_i$;
- $\varepsilon_f = 0.01$ Z feasible tolerance relative to the upper boundaries: $|\sum x_i - b_i| < \varepsilon_f$.

All the numerical results presented in the tables have been rounded to four decimal places.

### 4.2. Test problem 1

The first test problem has already been defined in Equation (1). The global minimum value of the function is equal to −36 and the minimum point is (0,0). Table 2 reports the results obtained from our proposed algorithm when minimizing Equation (1). As shown in Table 2, the SLGA converges in the 11th generation.

As can be observed in Table 3 and Fig. 4, the vertices defined by the lower and upper boundaries constitute the initial population $P(0)$, which has been denoted by $x^1$. Since the function is defined on $\Re^2$, four vertices {(1, 1), (−1, 1), (1, −1) and (−1, −1)} exist. The value of the function $f(x)$ is identical on all the vertices of the polytope, leading to the random selection of the initial best point. That is, $S = (1,1)$ is randomly selected as the best point of the initial population. Note that $h_i^1 = 2$, i=1,2, is determined by the width of the domain on which the $x_i$ variables are defined. Thus, $h_i^2 = 1$.

We implement now the mutation operator, leading to the first mutated population represented on the third column of Table 3. In order to provide additional intuition, we describe how the point (−1, 1) from the initial population becomes point (0, 0) after applying the mutation operator $\overset{M}{\to}$ with a step of length one (i.e. $h_i^2 = 1$, $i = 1,2$). Consider the set of mutated points obtained from (−1, 1) and described in the first column of Table 4. The first mutated point, (0, 2), generated after

**Table 2**
Results of our proposed algorithm (SLGA) for test problem 1.

| Gen. num. | Fitness | Best Point | |
|---|---|---|---|
| | | $x_1$ | $x_2$ |
| 1 | −35.4986 | 0.5000 | 0.5000 |
| 2 | −35.8747 | 0.2500 | 0.2500 |
| 3 | −35.9687 | 0.1250 | 0.1250 |
| 4 | −35.9922 | 0.0625 | 0.0625 |
| 5 | −35.9980 | 0.0313 | 0.0313 |
| 6 | −35.9995 | 0.0156 | 0.0156 |
| **7** | −36.0000 | 0.0078 | 0.0078 |
| 8 | −36.0000 | 0.0039 | 0.0039 |
| 9 | −36.0000 | 0.0020 | 0.0020 |
| 10 | −36.0000 | 0.0009 | 0.0009 |
| 11 | −36.0000 | 0.0004 | 0.0004 |

**Table 3**
First generation of *f*.

| $h_i^1 = 2\ x^1$ | $h_i^2 = 1$ | $g(X^1)$ | $l(x^1)$ | Crossover Points | Best point |
|---|---|---|---|---|---|
| (−1, 1) | $\overset{M}{\rightarrow}$ | (0, 0) | 2 | A = (0, 1) | C |
| (1, 1) | | (0, 0) | 2 | B = (1, 0) | |
| (−1, −1) | | (0, 0) | 0 | C = (0.5, 0.5) | |
| (1, −1) | | (0, 0) | 1 | | |

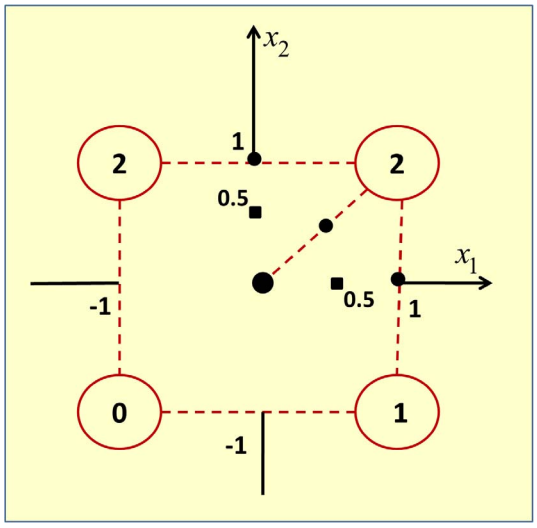


**Fig. 4.** First generation of *f*.

adding one unit to each entry of (−1, 1), achieves a fitness value equal to −6.44, as can be observed in the second column of Table 4. Thus, when accounting for the whole set of mutated points described in the first column, (0, 2), (−2, 0), (0, 1), (−1, 2), (−1, 0), (−2, 1), (0, 0) and (−2, 2), we select (0, 0) since it achieves the best (lowest) fitness value among them. The process is repeated for each point of the initial population until the set of best mutated points described in the third column of Table 3 is obtained.

Consider now the labeling operation, where each point of the initial population is compared with the image generated by the mutation operator. In the example above, $x^1 = (x_1^1, x_2^1) = (-1,1)$ and $g(X^1) = (g_1(X^1), g_2(X^1)) = (0,0)$, implying that

$$\left.\begin{aligned} g_1(X^1) - x_1^1 &= 0-(-1) = 1 > 0 \\ g_2(X^1) - x_2^1 &= 0-1 = -1 < 0 \end{aligned}\right\}$$

which is labeled as $l(x^1) = 2$, since the second component of $(g_1(X^1) - x_1^1, g_2(X^1) - x_2^1) = (1,-1)$ is negative. That is, the labeling of the vertices of the polytope provides a guideline that describes the direction in which the mutation operation is guiding the algorithm.

For example, consider the vector $(g_1(X^1) - x_1^1, g_2(X^1) - x_2^1, \ldots, g_n(X^1) - x_n^1) = (-1,\ 0,\ 1,\ -1,\ 1)$. We say that $l(x^1) = 4$ since the last negative component of the previous vector is the fourth one. In other words, the search space is being restricted on its fourth dimension. Thus, the set of labeled polytopes presented in Figs. 4–6 describes the direction in which the search space is being constrained so as to reach the best available solution located at (0, 0).

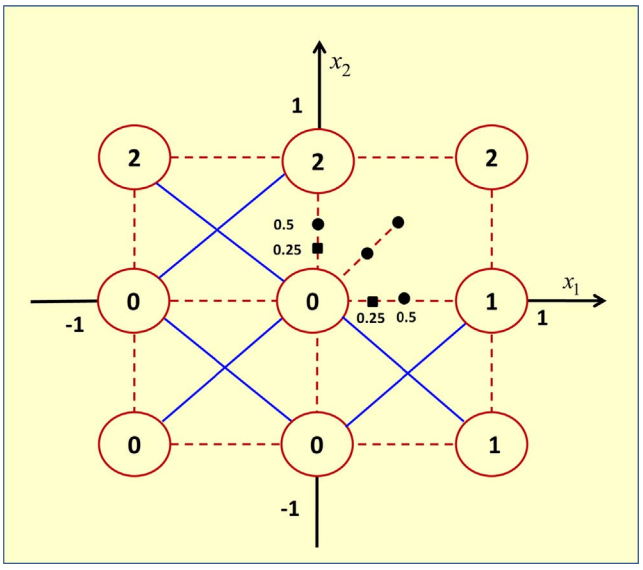


**Fig. 5.** Second generation of *f*.

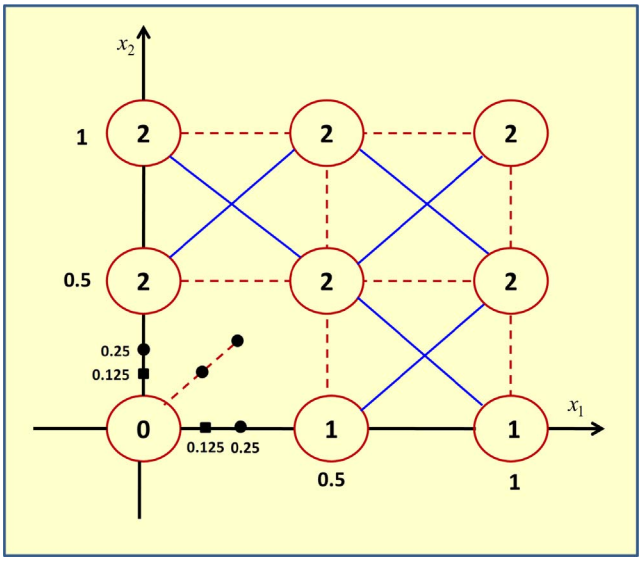


**Fig. 6.** Third generation of *f*.

**Table 4**
Implementing the mutation operator.

| Mutated points | Fitness function value |
|---|---|
| $(-1 + h_2, 1 + h_2) = (0, 2)$ | $f(0,2) = 0 + 4 - 18\cos 0 - 18\cos 2 = 4 - 18 + 18(0.42) = -14 + 7.56 = -6.44$ |
| $(-1 - h_2, 1 - h_2) = (-2, 0)$ | $f(-2,0) = 4 + 0 - 18\cos(-2) - 18\cos 0 = 4 - 18 + 18(0.42) = -6.44$ |
| $(-1 + h_2, 1) = (0, 1)$ | $f(0,1) = 0 + 1 - 18\cos 0 - 18\cos 1 = 1 - 18 - 18(0.54) = -17 - 9.72 = -26.72$ |
| $(-1, 1 + h_2) = (-1, 2)$ | $f(-1,2) = 1 + 4 - 18\cos(-1) - 18\cos 2 = 5 - 18(0.54) - 18(0.42) = 5 - 9.72 + 7.56 = 2.84$ |
| $(-1, 1 - h_2) = (-1, 0)$ | $f(-1,0) = 1 + 0 - 18\cos(-1) - 18\cos 0 = 1 - 18(0.54) - 18 = 1 - 9.72 - 18 = -26.72$ |
| $(-1 - h_2, 1) = (-2, 1)$ | $f(-2,1) = 4 + 1 - 18\cos(-2) - 18\cos 1 = 5 + 18(0.42) - 18(0.54) = 5 + 7.56 - 9.72 = 2.84$ |
| $(-1 + h_2, 1 - h_2) = (0, 0)$ | $f(0,0) = 0 + 0 - 18\cos 0 - 18\cos 0 = -36$ |
| $(-1 - h_2, 1 + h_2) = (-2, 2)$ | $f(-2,2) = 4 + 4 - 18\cos(-2) - 18\cos(-2) = 8 + 18(0.42) + 18(0.42) = 8 + 7.56 + 7.56 = 23.12$ |

**Table 5**
Second generation of $f$.

| $h_i^2 = 1\ x^2$ | $h_i^3 = 0.5$ | $g(X^2)$ | $l(x^2)$ | Crossover Points | Best point |
|---|---|---|---|---|---|
| (1, 0) | $\overset{M}{\rightarrow}$ | (0.5, 0) | 1 | | E |
| (0, 1) | | (0, 0.5) | 2 | D = (0, 0.5) | |
| (−1, 0) | | (−0.5, 0) | 0 | E = (0.25, 0.25) | |
| (0, −1) | | (0, −0.5) | 0 | F = (0.5, 0) | |
| (0, 0) | | (0, 0) | 0 | | |

After labeling the polytope, we implement the crossover operator between $S = (1,1)$ and the closest vertices – $v_1 = (-1,1)$, $v_2 = (1,-1)$ – together with their center point $v_3 = (0,0)$. Assuming that $w_1 = w_2 = w_3 = 0.5$, we obtain the following offspring from the crossover operation

$$S_1 = 0.5v_1 + 0.5S = 0.5(-1,1) + 0.5(1,1) = (0,1)$$

$$S_2 = 0.5v_2 + 0.5S = 0.5(1,-1) + 0.5(1,1) = (1,0)$$

$$S_3 = 0.5v + 0.5S = 0.5(0,0) + 0.5(1,1) = (0.5,0.5)$$

According to the value of the fitness function $f$, the best point of the first generation is given by (0.5, 0.5).

The design of the second generation is described in Table 5 and Fig. 5. We start by subdividing the search space and defining the resulting vertices in the first column of Table 5. After implementing the mutation operator – with $h_i^3 = 0.5$ – and labeling the polytope, the crossover operation based on $S' = (0.5,0.5)$ delivers the three offspring constituting the second generation. The best point of the second generation is given by (0.25, 0.25).

Fig. 5 illustrates how the subdivision implemented when designing the third generation will take place within the section defined by the completely labeled vertices {(0, 1), (1, 0), (1, 1), (0, 0)}, since the other labeled squares do not contain the minimum value of $f(x)$. That is, before implementing the subdividing and labeling operations per iteration, the sections of the search space that do not contain the best point are sequentially removed until the search space is constrained to a unique (optimal) point. The subdivision of the corresponding search space together with the mutation and crossover operations are described in Table 6 and Fig. 6. Ultimately, as can be observed in Fig. 7, the algorithm stops in the 11th generation at the near-optimum point (0.0004, 0.0004), which is the closest point to the ideal minimum (0, 0) with a precision of $\varepsilon_c = 0.001$.

### 4.3. Test problem 2

The second test is given by the continuous unconstrained nonlinear (CUCNL) problem defined in Equation (3). The global minimum value of the function is equal to 1, and the minimum point is $x = 2.5$. Table 7 describes the results derived from implementing the SLGA to minimize Equation (3).

$$\begin{aligned} &Min\ e^{(x-2.5)^2} \\ &-5 \leqslant x \leqslant 5 \end{aligned} \tag{3}$$

As shown in Table 7, our algorithm converges in the 12th generation.

**Table 6**
Third generation of $f$.

| $h_i^3 = 0.5$ $x^3$ | $h_i^4 = 0.25$ | $g(X^3)$ | $l(x^3)$ | Subdivision Points | Best point |
|---|---|---|---|---|---|
| (0.5, 0) | $\overset{M}{\rightarrow}$ | (0.25, 0) | 1 | | H |
| (0, 0.5) | | (0, 0.25) | 2 | G = (0, 0.25) | |
| (1, 0.5) | | (0.75, 0.25) | 2 | H = (0.125, 0.125) | |
| (0.5, 1) | | (0.25, 0.75) | 2 | K = (0.25, 0) | |
| (0.5, 0.5) | | (0.25, 0.25) | 2 | | |

The algorithm stops at the near-optimum point 2.5006, which is close to the ideal minimum point 2.5 with a precision of $\varepsilon_c = 0.001$. The initial population is defined by the lower and upper boundaries $-5$, $+5$. We select the point $+5$, since it delivers the minimum value of the function when compared to point $-5$.

After implementing the mutation operator, we obtain the first generation of offspring consisting of two points produced via crossover, i.e. 0 and 3.75. The point 3.75 is the best point of the first generation. Then, we restrict (subdivide) the search space to [0, 5] before applying the mutation operator and obtaining the next generation via crossover. The second generation consists of the points 2.5 and 3.125. As shown in Table 7, the point 3.125 is the best point of the second generation. Ultimately, the algorithm stops in the 12th generation at the minimum point 2.5006 with a precision of $\varepsilon_c = 0.001$.

The results obtained when solving the same test problem with an integer variable are summarized in Table 8. In this case, the global minimum value is equal to $e^{0.25} = 1.2840$, and the minimum point is given by $x = 3$ or $x = 2$.

### 4.4. Performance of the SLGA method on constrained nonlinear problems

In this section, we illustrate how to solve continuous and discrete constrained nonlinear problems using the proposed algorithm. The contents of this section and the next one build directly on the results obtained by Esmaelian et al. [12].

The novel SLGA approach introduced in the previous sections was applied to solve CUCNL optimization problems of the type

$$\begin{aligned} &Min\ f(x_1,x_2,...,x_n) \\ &s.\ t. \\ &a_j \leqslant x_j \leqslant b_j (j = 1,2,...,n) \end{aligned} \tag{4}$$

Consider model (4) as the standard form of the problem being analyzed. The continuous constrained nonlinear (CCNL) form of the problem is defined as follows:

$$\begin{aligned} &Min\ f(x_1,x_2,...,x_n) \\ &\text{s. t.} \\ &g_i(x_1,x_2,...,x_n)\begin{bmatrix}\leqslant\\=\\\geqslant\end{bmatrix}c_i \quad i = 1,2,...,m \\ &a_j \leqslant x_j \leqslant b_j \quad (j = 1,2,...,n) \end{aligned} \tag{5}$$

In order to transform the CCNL problem into its CUCNL form, we add the violation of each constraint to the objective function of the CCNL problem as follows

$$\begin{aligned} &Min\ f(x_1,x_2,...,x_n) + M_i v_i \\ &s.\ t. \\ &a_j \leqslant x_j \leqslant b_j (j = 1,2,...,n) \\ &M_i > 0\ ,\forall\ i \end{aligned} \tag{6}$$

where $M_i$ is the violation penalty associated to the $i^{th}$ constraint, while $v_i$ represents the extent to which the $i^{th}$ constraint is violated

$$g_i(x_1,x_2,...,x_n) \leqslant c_i \Rightarrow \begin{cases} if & g_i(x_1,x_2,...,x_n) \leqslant c_i \rightarrow v_i = 0 \\ if & g_i(x_1,x_2,...,x_n) > c_i \rightarrow v_i = g_i(x_1,x_2,...,x_n) - c_i \end{cases}$$

$$g_i(x_1,x_2,...,x_n) \geqslant c_i \Rightarrow \begin{cases} if & g_i(x_1,x_2,...,x_n) \geqslant c_i \rightarrow v_i = 0 \\ if & g_i(x_1,x_2,...,x_n) < c_i \rightarrow v_i = c_i - g_i(x_1,x_2,...,x_n) \end{cases}$$

$$g_i(x_1,x_2,...,x_n) = c_i \Rightarrow \begin{cases} if & g_i(x_1,x_2,...,x_n) = c_i \rightarrow v_i = 0 \\ if & g_i(x_1,x_2,...,x_n) \neq c_i \rightarrow v_i = |g_i(x_1,x_2,...,x_n) - c_i| \end{cases}$$

If $v_i \leqslant \varepsilon_f$ for all the constraints, then the solution obtained is feasible. However, if $v_i > \varepsilon_f (\max_i(v_i) > \varepsilon_f)$ for at least one constraint, then the solution obtained is non-feasible.

The discrete constrained nonlinear (DCNL) form of the problem is defined as follows:

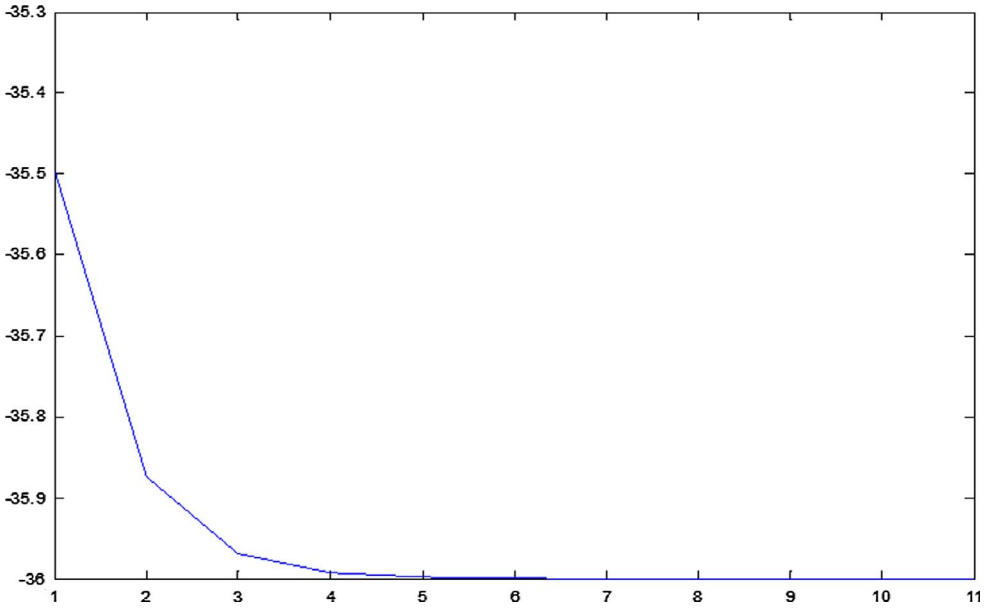


**Fig. 7.** Convergence graph for test problem 1.

**Table 7**
Results of our proposed algorithm for test problem 2.

| Gen. num. | Fitness | Best Point |
|---|---|---|
| | | $x$ |
| 1 | 4.7708 | 3.7500 |
| 2 | 1.4780 | 3.1250 |
| 3 | 1.1026 | 2.8125 |
| 4 | 1.0247 | 2.6563 |
| 5 | 1.0061 | 2.5781 |
| 6 | 1.0015 | 2.5391 |
| 7 | 1.0004 | 2.5195 |
| 8 | 1.0001 | 2.5098 |
| 9 | 1.0000 | 2.5049 |
| 10 | 1.0000 | 2.5024 |
| 11 | 1.0000 | 2.5012 |
| 12 | 1.0000 | 2.5006 |

**Table 8**
Results of our proposed algorithm for test problem 2 with an integer variable.

| Gen. num. | Fitness | Best Point |
|---|---|---|
| | | $x$ |
| 1 | 4.7707 | 3.7500 |
| 2 | 1.4779 | 3.1250 |
| 3 | 1.4779 | 3.1250 |
| 4 | 1.3486 | 3.0469 |
| 5 | 1.3486 | 3.0469 |
| 6 | 1.2942 | 3.0078 |
| 7 | 1.2942 | 3.0078 |
| 8 | 1.2878 | 3.0029 |
| 9 | 1.2866 | 3.0020 |
| 10 | 1.2862 | 3.0017 |
| 11 | 1.2847 | 3.0005 |

$$
\begin{aligned}
&Min\ f(x_1,x_2,...,x_n)\\
&\text{s. t.}\\
&g_i(x_1,x_2,...,x_n)\begin{bmatrix}\leqslant\\=\\\geqslant\end{bmatrix}c_i \quad i=1,2,...,m\\
&a_j \leqslant x_j \leqslant b_j \quad (j=1,2,...,n)\\
&x_j \quad \text{integer}
\end{aligned}
\tag{7}
$$

In this case, the violation of each constraint and each variable (this latter one takes place when differing from an integer value) are added to the objective function and the DCNL problem is transformed into the following CUCNL problem

$$
\begin{aligned}
&Min\ f(x_1,x_2,...,x_n)+M_iv_i+N_jw_j\\
&s.\ t.\\
&a_j \leqslant x_j \leqslant b_j (j=1,2,...,n)\\
&M_i>0\ ,\forall\ i\\
&N_j>0\ ,\forall\ j
\end{aligned}
\tag{8}
$$

where $N_j$ is the violation penalty associated to variable $x_j$, while $w_j$ is the extent to which the variable $x_j$ differs from an integer value

$$
\begin{cases}
if \quad x_j = round(x_j) \Rightarrow w_j = 0\\
if \quad x_j \neq round(x_j) \Rightarrow w_j = |x_j - round(x_j)|
\end{cases}
$$

Note that $M_i$ and $N_j$ are constant parameters used to transform CNL problems into CUCNL ones. Intuitively, they perform a similar role to that of the Lagrange multipliers in constrained optimization problems, adding a violation penalty per constraint to the objective function. Clearly, if the whole set of constraints is satisfied (i.e. $v_i = 0\ ,\forall\ i$, and $w_j = 0\ ,\forall\ j$), then the expression $f(x_1,x_2,...,x_n) + M_iv_i + N_jw_j$ simplifies to $f(x_1,x_2,...,x_n)$.

Finally, the discrete unconstrained nonlinear (DUCNL) form of the problem is defined as follows:

$$
\begin{aligned}
&Min\ f(x_1,x_2,...,x_n)\\
&s.\ t.\\
&a_j \leqslant x_j \leqslant b_j (j=1,2,...,n)\\
&x_j \text{integer}
\end{aligned}
\tag{9}
$$

The violation of each variable (when differing from an integer value) is added to the objective function and the DUCNL problem transformed into a CUCNL problem as follows

$$
\begin{aligned}
&Min\ f(x_1,x_2,...,x_n)+N_jw_j\\
&s.\ t.\\
&a_j \leqslant x_j \leqslant b_j (j=1,2,...,n)\\
&N_j>0\ ,\forall\ j
\end{aligned}
\tag{10}
$$

where $N_j$ is the violation penalty associated to variable $x_j$, while $w_j$ is the extent to which the variable $x_j$ differs from an integer value.

### 4.5. Applicable experiments

In order to further evaluate the proposed SLGA method, we consider three additional reference frameworks consisting of transportation, allocation, and mixed-variable problems, all of them commonly used in applied mathematics, management and industrial engineering.

#### 4.5.1. Test problem 3

The following constrained nonlinear setting with continuous $x_{ij}$ variables defines a typical transportation problem

$$
\begin{aligned}
&Min \sum_{i=1}^{2}\sum_{j=1}^{2} c_{ij}x_{ij}\\
&\text{s. t.}\\
&x_{11} + x_{12} = 15\\
&x_{21} + x_{22} = 15\\
&x_{11} + x_{21} = 20\\
&x_{12} + x_{22} = 10\\
&c_{11}(x_{11}) = 15x_{11} - 0.02x_{11}^2\\
&c_{21}(x_{21}) = 4x_{21} - 0.04x_{21}^2\\
&c_{12}(x_{12}) = 10x_{12} - 0.01x_{12}^2\\
&c_{22}(x_{22}) = 7x_{11} - 0.01x_{22}^2\\
&0 \leqslant x_{ij} \leqslant 15 \quad (i,j = 1,2)
\end{aligned}
\tag{11}
$$

Table 9 presents the results obtained after implementing our proposed algorithm to minimize Equation (11). The SLGA method converges in the 10th generation and stops at the near-optimum point ($x_{11} = 5.0098$, $x_{12} = 10$, $x_{21} = 15$, $x_{22} = 0.0098$) with precision $\varepsilon_i = 0.01$, $\varepsilon_c = 0.001$ and $\varepsilon_f = 0.01$.

#### 4.5.2. Test problem 4

Consider now the following constrained nonlinear setting with binary $x_{ij}$ variables, defining a standard allocation problem

$$
\begin{aligned}
&Min \sum_{i=1}^{2}\sum_{j=1}^{2} c_{ij}x_{ij}\\
&\text{s. t.}\\
&x_{11} + x_{12} = 1\\
&x_{21} + x_{22} = 1\\
&x_{11} + x_{21} = 1\\
&x_{12} + x_{22} = 1\\
&c_{11}(x_{11}) = 5x_{11}^2\\
&c_{21}(x_{21}) = 3x_{21}^2\\
&c_{12}(x_{12}) = 4x_{12}^2\\
&c_{22}(x_{22}) = 7x_{22}^2\\
&Binary\ x_{ij} \quad (i,j = 1,2)
\end{aligned}
\tag{12}
$$

Table 10 describes the results obtained after implementing the proposed SLGA method to minimize Eq. (12). The algorithm stops after 9 generations at the near-optimum point ($x_{11} = 0.0009$, $x_{12} = 0.999$, $x_{21} = 0.999$, $x_{22} = 0.0009$) with precision $\varepsilon_i = 0.01$, $\varepsilon_c = 0.001$ and $\varepsilon_f = 0.01$.

**Table 9**
Results of our proposed algorithm for test problem 3.

| Gen. num. | Fitness | Best Point | | | |
|---|---|---|---|---|---|
| | | $x_{11}$ | $x_{12}$ | $x_{21}$ | $x_{22}$ |
| 1 | 275.8750 | 10 | 7.5000 | 10 | 2.5000 |
| 2 | 228.3750 | 6.2500 | 8.7500 | 11.2500 | 1.2500 |
| 3 | 222.8437 | 5.6250 | 9.3750 | 11.8750 | 0.6250 |
| 4 | 238.1133 | 5.6250 | 10 | 15 | 0.6250 |
| 5 | 231.3096 | 5.3125 | 10 | 15 | 0.3125 |
| 6 | 227.9066 | 5.1563 | 10 | 15 | 0.1563 |
| **7** | 226.2024 | 5.0781 | 10 | 15 | 0.0781 |
| 8 | 225.3523 | 5.0391 | 10 | 15 | 0.0391 |
| 9 | 224.9251 | 5.0195 | 10 | 15 | 0.0195 |
| 10 | 224.7136 | 5.0098 | 10 | 15 | 0.0098 |

**Table 10**
Results of our proposed algorithm for test problem 4.

| Gen. num. | Fitness | Best Point | | | |
|---|---|---|---|---|---|
| | | $x_{11}$ | $x_{12}$ | $x_{21}$ | $x_{22}$ |
| 1 | 4.6875 | 0.2500 | 0.7500 | 0.7500 | 0.2500 |
| 2 | 5.5469 | 0.1250 | 0.8750 | 0.8750 | 1.2500 |
| 3 | 6.1992 | 0.0625 | 0.9375 | 0.9375 | 0.0625 |
| 4 | 6.5818 | 0.0313 | 0.9688 | 0.9688 | 0.0313 |
| 5 | 6.7862 | 0.0156 | 0.9844 | 0.9844 | 0.0156 |
| 6 | 6.8920 | 0.0078 | 0.9922 | 0.9922 | 0.0078 |
| **7** | 6.9457 | 0.0039 | 0.9961 | 0.9961 | 0.0039 |
| 8 | 6.9721 | 0.0020 | 0.9980 | 0.9980 | 0.0020 |
| 9 | 6.9860 | 0.0009 | 0.9990 | 0.9990 | 0.0009 |

#### 4.5.3. Test problem 5

The last test performed is given by the following mixed-variable optimization problem

$$
\begin{aligned}
&Min\ Z = 3x_{11}^2 + 5x_{12}^2 + 7x_{21}^2 + 4x_{22}^2 + 80y_1 + 80y_2\\
&\text{s. t.}\\
&x_{11} + x_{12} = 10y_1\\
&x_{21} + x_{22} = 10y_2\\
&x_{11} + x_{21} = 10\\
&x_{12} + x_{22} = 10\\
&0 \leqslant x_{ij} \leqslant 10 \quad (i,j = 1,2)\\
&y_i = 0,1 \quad (i = 1,2)
\end{aligned}
\tag{13}
$$

Table 11 provides the results obtained from the SLGA when minimizing Equation (13). Similarly to Problem (12), the algorithm converges in the 9th generation and stops at the optimum point ($x_{11} = 9.990$, $x_{12} = 0.0098$, $x_{21} = 0.0098$, $x_{22} = 9.990$, $y_1 = 0.999$, and $y_2 = 0.999$) with a precision of $\varepsilon_i = 0.01$, $\varepsilon_c = 0.001$ and $\varepsilon_f = 0.01$.

## 5. Comparison with other methods

In this section a comprehensive evaluation of SLGA is performed by comparing the proposed approach with several similar methods. We start by defining a set of test functions that are suitable to compare the different approaches.

### 5.1. Test functions used in simulation studies

We test our proposed method using the De Jong functions [11], with $\varepsilon_c = 0.001$. The De Jong functions are quite popular in the function optimization literature [18,21,23,24]. The choice of this class of functions is justified by their ability to account for some of the most

**Table 11**
Results of our proposed algorithm for test problem 5.

| Gen. num. | Fitness | Best Point | | | | | |
|---|---|---|---|---|---|---|---|
| | | $x_{11}$ | $x_{12}$ | $x_{21}$ | $x_{22}$ | $y_1$ | $y_2$ |
| 1 | 558.7500 | 7.5000 | 2.5000 | 2.5000 | 7.5000 | 0.7500 | 0.7500 |
| 2 | 677.1875 | 8.7500 | 1.2500 | 1.2500 | 8.7500 | 0.8750 | 0.8750 |
| 3 | 760.5469 | 9.3750 | 0.6250 | 0.6250 | 9.3750 | 0.9375 | 0.9375 |
| 4 | 808.3450 | 9.6880 | 0.3125 | 0.3125 | 9.6880 | 0.9688 | 0.9688 |
| 5 | 833.6697 | 9.8440 | 0.1561 | 0.1561 | 9.8440 | 0.9844 | 0.9844 |
| 6 | 846.7097 | 9.9220 | 0.0782 | 0.0782 | 9.9220 | 0.9922 | 0.9922 |
| **7** | 853.3234 | 9.9610 | 0.0391 | 0.0391 | 9.9610 | 0.9961 | 0.9961 |
| 8 | 856.5680 | 9.9800 | 0.0196 | 0.0196 | 9.9800 | 0.9980 | 0.9980 |
| 9 | 858.2820 | 9.9900 | 0.0098 | 0.0098 | 9.9900 | 0.9990 | 0.9990 |

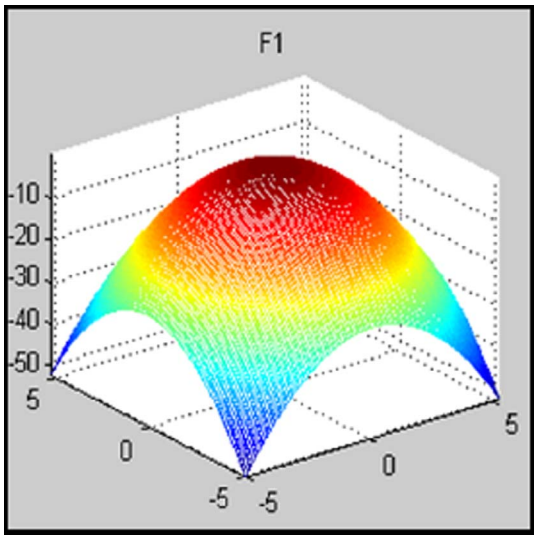


**Fig. 8.** F1 Sphere function.

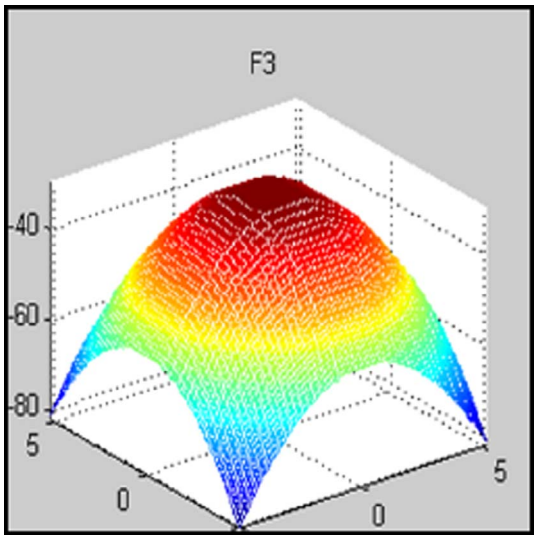


**Fig. 10.** F3 Step function.

common difficulties affecting optimization problems. Hence, the use of these functions allows for fair judgments when analyzing the strengths and weaknesses of particular algorithms.

The De Jong functions are represented in Figs. 8–15 while a survey of their main characteristics is presented in Table 12. The first De Jong function provides one of the simplest test benchmarks available. This function is continuous, convex, three-dimensional and unimodal, i.e. it has one solution. The test area is usually restricted to the polytope $-5.12 \leqslant x_i \leqslant 5.12, i = 1,2,3$, and its global minimum $f(x) = 0$ is obtained for $x_i = 0$.

The second De Jong function, known as Rosenbrock's valley or banana function, represents a classic optimization problem. Despite the fact that this function is unimodal and two-dimensional, the global optimum is to be found in a long, narrow, parabolic shaped flat valley. As a consequence, the convergence process to the global optimum is rather difficult and involved. The test area is usually restricted to the square $-2.048 \leqslant x_i \leqslant 2.048, i = 1,2$, while the global minimum $f(x) = 0$ is obtained for $x_i = 1$.

The third De Jong function is a step function typically used to model the problem of flat surfaces. Flat surfaces cannot provide information on the most favorable direction to follow. This fact imposes addition obstacles on an optimization algorithm that can get stuck on one of the

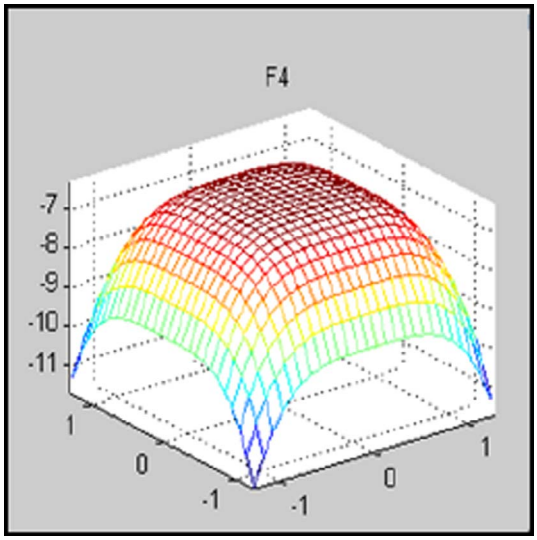


**Fig. 11.** F4 Stochastic function.

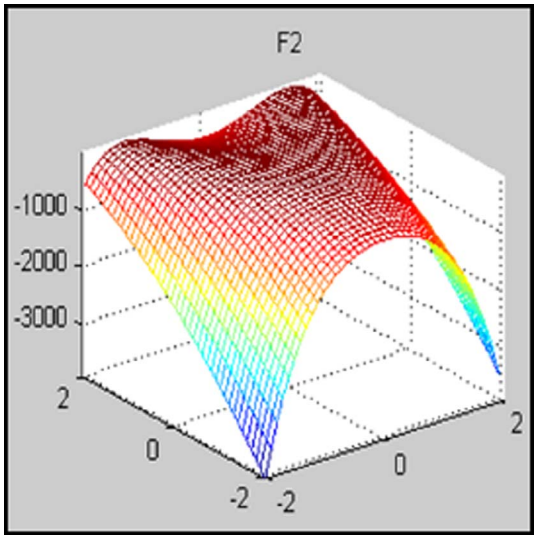


**Fig. 9.** F2 Rosenbrock's function.

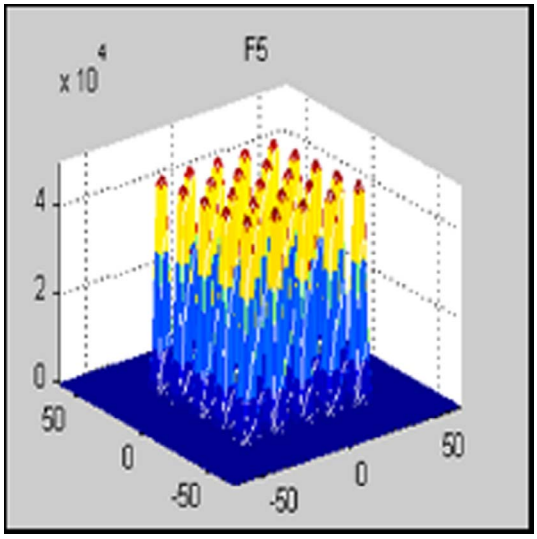


**Fig. 12.** F5 Foxholes function.

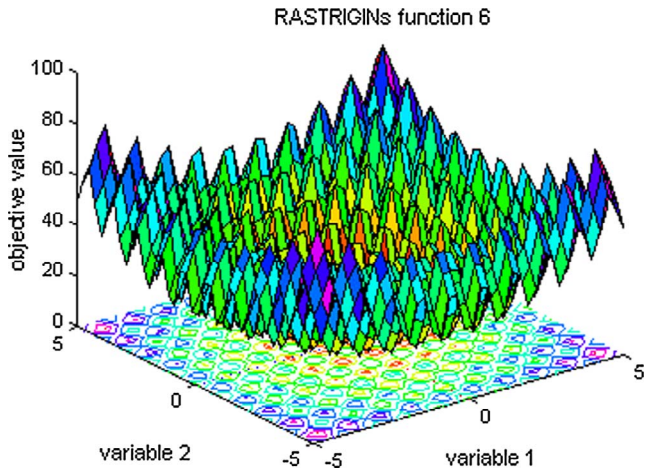


**Fig. 13.** F6 Rastrigin's function.

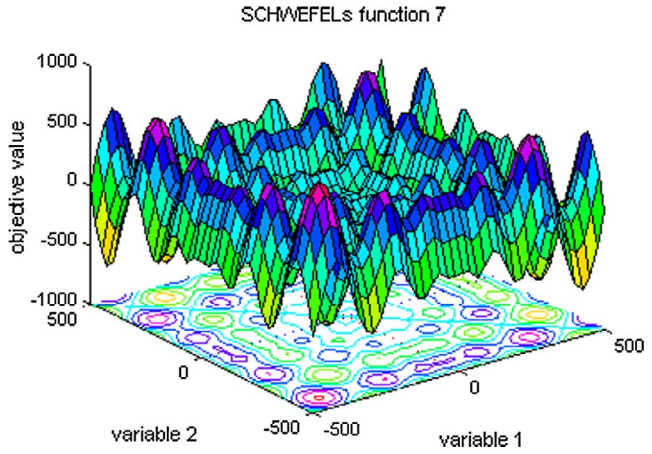


**Fig. 14.** F7 Schwefel's function.

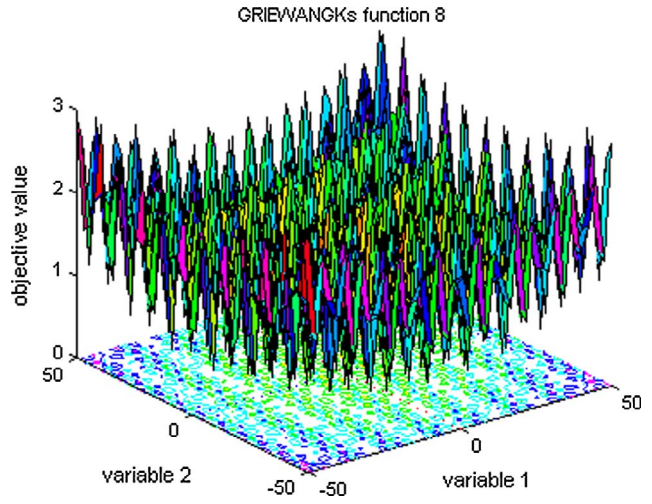


**Fig. 15.** F8 Griewangk's function.

flat plateaus unless it allows for variable step sizes. The third De Jong function is multidimensional, multimodal and admits an uncertain number of local optimums. The test area is usually restricted to the polytope $-5.12 \leqslant x_i \leqslant 5.12, i = 1,2,3,4,5$. Its global minimum $f(x) = 0$ is obtained for $x_i = -5-\varepsilon$, where $\varepsilon \in [0,0.12]$. Since the SLGA method has been designed to solve nonlinear problems, we consider the following step function: $F_3 = 30 + \sum_{i=1}^{5} \lfloor x_i^3 \rfloor, -5.12 \leqslant x_i \leqslant 5.12$.

The fourth De Jong function is a stochastic thirty-dimensional function. Using this type of functions, the values returned by multiple implementations of the algorithm on a certain point are different. Without the added Gaussian noise, the function is a simple unimodal function whose global minimum $f(x) = 0$ is obtained for $x_i = 0$. Adding the noise gives place to a multimodal function attaining an uncertain number of local optimums. We limit the test area to the polytope $-1.28 \leqslant x_i \leqslant 1.28, i = 1,2,...,30$.

The fifth De Jong function, also known as the Foxholes function, is a multimodal test function admitting 25 local minima in the square $-65.536 \leqslant x_i \leqslant 65.536, i = 1,2$. Even though many standard optimization algorithms get stuck in the first peak they find, its global optimum is given by $f(x) = 1$ for $x_i = -32$.

The sixth De Jong function, also called Rastrigin's function, is a multimodal test function that adds cosine modulation in order to produce many local minima regularly distributed within the square $-5.12 \leqslant x_i \leqslant 5.12, i = 1,2$. Its global optimum is given by $f(x) = 0$ for $x_i = 0$.

The seventh De Jong function, also known as Schwefel's function, is deceptive since its global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, search algorithms are potentially prone to convergence in the wrong direction. Its global optimum is given by $f(x) = -n.\ 418.9829$ for $x_i = 420.9687$ with $i = 1,...,n$.

The eighth De Jong function, also called Griewangk's function, is similar to Rastrigin's in that it has many widespread local minima whose location is regularly distributed. Its global optimum is given by $f(x) = 0$ for $x_i = 0$ with $i = 1,2$.

The last three De Jong functions have been introduced in the analysis to illustrate the fact that the SLGA method can handle, to a certain extent, problems with very rugged search spaces. Thus, even though the next section will focus on performing comparisons with other methods for the first five De Jong functions, which have few local optima, a numerical analysis illustrating the performance of the SLGA method when considering multiple local optima will also be provided.

### 5.2. Experimental results on the De Jong functions

The performance of our SLGA approach on the set of De Jong functions was evaluated and compared with that of several other similar methods.

Table 13 presents the numerical results provided by our method as well as the time required to reach them. The alternative genetic algorithms taken in consideration for the performance analysis include: parallel genetic algorithm [23], Grefensstette [26], Eshelman [26], differential evolution [23], genetic algorithm [17], simulated annealing [25], and imperialist competitive algorithm (ICA) [1]. For a more detailed description of the single algorithms please refer to the cited Refs. Table 14 reports the number of generations needed for the algorithms to converge to the near-optimum point of the first five De Jong functions.

The symbol "—" in Table 14 means that the algorithm was unable to converge. According to the obtained results, the simulated annealing method cannot be used to attain a solution for step function (F3), stochastic function (F4) or multimodal function (F5). Moreover, the ICA does not converge in the step function case.

Except for the convex function (F1) case where ICA performs better, all the other values in Table 14 imply a significantly better performance of our approach with respect to the other methods, requiring a much lower number of generations to reach the corresponding optima.

Note also that the SLGA method manages to reach the near-optimum value of the Rastrigin function, while this is not the case for the Schwefel and Griewangk ones. Thus, the SLGA method can handle, to a certain extent, problems with very rugged search spaces, though further improvements are clearly required to increase its accuracy.

**Table 12**
The De Jong Functions.

| | Type | Function | Limits | Dim. | Number of Local Minima |
|---|---|---|---|---|---|
| F1 | Convex Function | $\sum_{i=1}^{3} x_i^2$ | $-5.12 \leqslant x_i \leqslant 5.12$ | 3 | 1 |
| F2 | Banana Function | $100(x_1^2-x_2)^2 + (1-x_1)^2$ | $-2.048 \leqslant x_i \leqslant 2.048$ | 2 | 1 |
| F3 | Step Function | $F_3 = 30 + \sum_{i=1}^{5} \lfloor x_i^3 \rfloor$ | $-5.12 \leqslant x_i \leqslant 5.12$ | 5 | 1 |
| F4 | Stochastic Function | $\sum_{i=1}^{30} i x_i^4 + Gauss(0,1)$ | $-1.28 \leqslant x_i \leqslant 1-28$ | 30 | Uncertain |
| F5 | Foxholes Function | $\frac{1}{0.002 + \sum_{i=1}^{2} \frac{1}{i + \sum_{j=0}^{1} (x_i - a_{ij})^6}}$ | $-65.536 \leqslant x_i \leqslant 65.536$ | 2 | 25 |
| F6 | Rastrigin's function | $10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2.\ \pi.\ x_i))$ | $-5.12 \leqslant x_i \leqslant 5.12$ | 2 | Uncertain |
| F7 | Schwefel's function | $\sum_{i=1}^{n} -x_i.\ \sin(\sqrt{\lvert x_i \rvert})$ | $-500 \leqslant x_i \leqslant 500$ | 2 | Uncertain |
| F8 | Griewangk's function | $\sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | $-600 \leqslant x_i \leqslant 600$ | 2 | Uncertain |

**Table 13**
Optimizing the test functions by SLGA.

| | Gen. num. | Optimal Point | Function value | Time (Sec.) |
|---|---|---|---|---|
| F1 | 14 | (3.125E−04, 3.125E−04, 3.125E−04) | 0.0009 | 1.7605 |
| F2 | 9 | (1, 1) | 0 | 0.2242 |
| F3 | 11 | (−5.1050, −5.1050, −5.1050, −5.1050, −5.1050) | −640 | 301.3480 |
| F4 | 12 | $\underbrace{(0,\ldots,0)}_{30}$ | $Gauss(0,1)$ | 21.345 |
| F5 | 13 | (−32.0101, −32.0101) | 0.9980 | 0.2631 |
| F6 | 14 | (0.0006, 0.0006) | 0.00014 | 0.2549 |
| F7 | 11 | (−0.00049, −0.999) | 3.00007 | 0.2052 |
| F8 | 18 | (3.1414, 3.1414) | −1 | 0.3014 |

**Table 14**
Number of generation required for optimizing the De Jong functions.

| Algorithm | F1 | F2 | F3 | F4 | F5 |
|---|---|---|---|---|---|
| Parallel Genetic Algorithm ($\lambda = 4$) [23] | 1170 | 1235 | 3481 | 3194 | 1256 |
| Parallel Genetic Algorithm ($\lambda = 8$) [23] | 1526 | 1671 | 3634 | 5243 | 2076 |
| Grefensstette [23] | 2210 | 14229 | 2259 | 3070 | 4334 |
| Eshelman [23] | 1538 | 9477 | 1740 | 4137 | 3004 |
| Differential Evolution [23] | 260 | 670 | 125 | 2300 | 1200 |
| Genetic Algorithm [17] | 202 | 87 | 51 | 525 | 1300 |
| Simulated Annealing [25] | 8959 | 420 | – | – | – |
| Imperialist Competitive Algorithm [1] | 12 | 997 | – | 700 | 44 |
| SLGA | 14 | 9 | 11 | 12 | 13 |

## 6. Conclusion

We have defined a genetic algorithm in a $n$ – dimensional space that combines subdividing and labeling techniques with mutation and crossover operators. Among the properties of the SLGA, we highlight the following. First, creating a network on the search space provides an integer coding system that simplifies the process of locating all individuals from the present and future generations. Moreover, the algorithm can start from a fixed point located in the boundary of the domain, overcoming the weakness of manmade initial points. Second, completely labeling the polytopes prevents the algorithm from getting stuck in misleading local solutions. Third, the mutation and crossover operators work systematically in order to find better solutions. Consequently, the SLGA technique eliminates unneeded generations and calculations, leading to a quicker and more effective performance.

We have evaluated the proposed algorithm on the De Jong functions and several other nonlinear optimization problems where it is applicable. Given the fact that the selected benchmark functions have a very large number of local minima, finding satisfactory solutions becomes a considerable challenge. It therefore follows from the results obtained that the proposed approach converges quickly to a global minimum, improving upon the solution processes of several other known algorithms.

As stated in the introduction, nonlinear optimization environments are common to the measurement literature, which allows the SLGA method to provide substantial contributions within the corresponding research areas. Finally, it should be noted that the SLGA method introduced in the current paper is deterministic, implying that stochastic environments should be accounted for and incorporated in any immediate extension of the model.

## Acknowledgements

The authors would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions. An earlier and very preliminary version of this paper was presented at the 2017 IEEE Congress on Evolutionary Computation in San Sebastián, Spain, and published in the corresponding proceedings.

## References

[1] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, IEEE Congr. Evol. Comput. 7 (2007) 4661–4666.
[2] A.L. Buczak, H. Wang, H. Darabi, M.A. Jafari, Genetic algorithm convergence study for sensor network optimization, Inf. Sci. 133 (3–4) (2001) 267–282.
[3] Y.J. Cao, L. Jiang, Q.H. Wu, An evolutionary programming approach to mixed – variable optimization problems, Appl. Math. Model. 24 (2000) 931–942.
[4] J. Cha, R. Mayne, Optimization with discrete variables via recursive quadratic programming: part II, Trans. ASME 111 (1989) 130–136.
[5] W. Chu, X. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high- dimensional problems, Inf. Sci. 181 (15) (2011) 4909–4927.
[6] K.-M. Chiu, J.-S. Liu, Robot Routing Using Clustering-Based Parallel Genetic Algorithm with Migration, in: IEEE Workshop on Merging Fields of Computational Intelligence and Sensor Technology (CompSens), 2011, pp. 42–49.
[7] E. Castillo, I. Gallego, José MaríaUreña, José MaríaCoronado, Timetabling optimization of a mixed double- and single-tracked railway network, Appl. Math. Modell. 35 (2) (2011) 859–878.
[8] V. Charles, A. Udhayakumar, Genetic algorithm for chance constrained reliability stochastic optimisation problems, Int. J. Oper. Res. 14 (4) (2012) 417–432.
[9] C.-C. Chen, W.-H. Wu, M.-R. Leu, G. Lai, Tension determination of stay cable or external tendon with complicated constraints using multiple vibration measurements, Measurement 86 (2016) 182–195.
[10] W. Chu, X. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high- dimensional problems, Inf. Sci. 181 (2011) 4909–4927.
[11] K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, [Doctoral Dissertation] University of Michigan, Ann Arbor, MI, USA, 1975.
[12] M. Esmaelian, F.J. Santos-Arteaga, M. Tavana, M. Vali, Subdividing Labeling Genetic Algorithm: A new method for solving continuous nonlinear optimization problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, San Sebastián, Spain, 2017, pp. 773–780.
[13] L.J. Fogel, A.J. Owens, M.J. Walgh, Artificial Intelligence Through Simulated Evolution, John Wiley, New York, USA, 1996.
[14] X. Guo, C. Song, R. Yan, Optimization of multilayer cylindrical coils in a wireless localization system to track a capsule-shaped micro-device, Measurement 46 (1) (2013) 117–124.
[15] P. Hajela, C. Shih, Multi objective optimum design in mixed-integer and discrete design variable problems, AIAA J. 28 (1989) 670–675.

[16] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, USA, 1975.
[17] C. Houck, J. Joines, M. Kay, A genetic algorithm for function optimization: a matlab implementation, NCSU-IE Tech. Rep. 95–09 (1995).
[18] K. Juneja, N.S. Gill, Optimization of De Jong function using GA under different selection algorithms, Int. J. Comput. Appl. 64 (7) (2013) 28–33.
[19] L. Jiao, L. Li, R. Shang, F. Liu, R. Stolkin, A nonel selection evolutionary strategy for constrained optimization, Inf. Sci. 239 (2013) 122–141.
[20] J. Jahn, Multi-objective search algorithm with subdivision technique, Comput. Optim. Appl. 35 (2) (2006) 161–175.
[21] M. Kapoor, V. Wadhwa, Optimization of De Jong's function using genetic algorithm approach, Int. J. Adv. Res. Comput. Sci. Electron. Eng. 1 (5) (2012) 35–38.
[22] D.H. Kim, A. Abraham, J.H. Cho, A hybrid genetic algorithm and bacterial foraging approach for global optimization, Inf. Sci. 177 (18) (2007) 3918–3937.
[23] D. Karaboga, S. Okdem, A simple and global optimization algorithm for engineering problems: differential evolution algorithm, Turkish J. Electr. Eng. Comput. Sci. 12 (1) (2004) 53–60.
[24] H-S. Kim, K.J. Mun, J.H. Park, G.-H. Hwang, Application of real-type tabusearch in function optimization problems, in: Proceedings of the IEEE International Symposium on Industrial Electronics, 2001, pp. 613–618.
[25] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[26] H. Muhlenbein, M. Schomisch, J. Born, The Parallel Genetic Algorithm as Function Optimizer, in: Proceedings of The Fourth International Conference on Genetic Algorithms, University of California, San Diego, 1991, pp. 270–278.
[27] C. Özgüven, Y. Yavuz, L. Özbakır, Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times, Appl. Math. Model. 36 (2) (2012) 846–858.
[28] Y. Peng, J. Cheng, Y. Yang, B. Li, Adaptive sparsest narrow-band decomposition method and its applications to rotor fault diagnosis, Measurement 91 (2016) 451–459.
[29] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (1990) 223–229.
[30] A.F. da Silva, F.A.S. Marins, J.A.B. Montevechi, Multi-choice mixed integer goal programming optimization for real problems in a sugar and ethanol milling company, Appl. Math. Model. 37 (9) (2013) 6146–6162.
[31] I.N. da Silva, W.C. do Amaral, L.V. de Arruda, A novel approach based on recurrent neural networks applied to nonlinear systems optimization, Appl. Math. Model. 31 (2007) 78–92.
[32] R. Storn, System design by constraint adaptation and differential evolution, IEEE Trans. Evol. Comput. 3 (1) (1999) 22–34.
[33] R. Storn, and K. Price, Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.
[34] A. Udhayakumar, V. Charles, M. Kumar, Stochastic simulation based genetic algorithm for chance constrained data envelopment analysis problems, Omega 39 (4) (2011) 387–397.
[35] A. Udhayakumar, V. Charles, V. Rhymend Uthariaraj, Stochastic simulation-based genetic algorithm for chance constrained fractional programming problem, Int. J. Oper. Res. 9 (1) (2010) 23–38.
[36] A. Ugur, Path planning on a cuboid using genetic algorithms, Inf. Sci. 178 (16) (2008) 3275–3287.
[37] J. Wang, Z. Wang, C. Yang, N. Wang, X. Yu, Optimization of the number of components in the mixed model using multi-criteria decision-making, Appl. Math. Model. 36 (9) (2012) 4227–4240.
[38] Ka-Chun Wong, Chun-Ho Wu, Ricky K.P. Mok, C. Peng, Z. Zhang, Evolutionary multimodal optimization using the principle of locality, Inf. Sci. 194 (2012) 138–170.
[39] A.H. Wright, J.E. Rowe, C.R. Stephens, R. Poli, Bistability in a Gene Pool GA with Mutation, In Foundations of genetic algorithms-7, San Mateo, 2003. Morgan Kaufmann.
[40] Chia-Huang Wu, Jau-Chuanke, Multi-server machine repair problems under a (V, R) synchronous single vacation policy, Appl. Math. Model. 38 (7–8) (2014) 2180–2189.
[41] J. Zhang, H. Wang, R. Geo, Study of an improved genetic algorithm based on fixed point theory and J1 triangulation in euclidean space, J. Comput. 6 (10) (2011) 2173–2179.