Information Sciences 241 (2013) 119–137

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

A rough penalty genetic algorithm for constrained optimization

Chih-Hao Lin *

Department of Information Management, Chung Yuan Christian University, Jhongli 320, Taiwan

ARTICLE INFO

Article history:
Received 21 November 2007
Received in revised form 24 March 2013
Accepted 1 April 2013
Available online 6 April 2013

Keywords:
Genetic algorithm
Penalty function
Rough set theory
Constrained optimization

ABSTRACT

Many real-world issues can be formulated as constrained optimization problems and solved using evolutionary algorithms with penalty functions. To effectively handle constraints, this study hybridizes a novel genetic algorithm with the rough set theory, called the rough penalty genetic algorithm (RPGA), with the aim to effectively achieve robust solutions and resolve constrained optimization problems. An infeasible solution is subjected to rough penalties according to its constraint violations. The crossover operation in the genetic algorithm incorporates a novel therapeutic approach and a parameter tuning policy to enhance evolutionary performance. The RPGA is evaluated on eleven benchmark problems and compared with several state-of-the-art algorithms in terms of solution accuracy and robustness. The performance analyses show this approach is a self-adaptive method for penalty adjustment. Remarkably, the method can address a variety of constrained optimization problems even though the initial population includes infeasible solutions.

1. Introduction

Most real-world optimization problems involve constraints. Since Holland first developed a genetic algorithm (GA) in 1975 [13], GAs have been successfully applied to a wide range of complex problems in science, engineering, and industry fields. The challenge of a constrained problem is how to optimize the objective function value against its constraint violations. However, traditional GAs may consume considerable computational energy in searching infeasible solutions because genetic operations do not always preserve feasibility [28]. Considerable research has focused on constraint-handling techniques for evolutionary algorithms (EAs) [37,38].

Traditionally, penalty-function methods are the most popular constraint-handling techniques [22]. Each infeasible solution is penalized by the magnitude of its constraint violations. Many studies attempt to manipulate penalty coefficients for balancing the objective function with constraint violations [4,7]. However, the static penalty method has difficulty in adjusting all the penalty factors empirically [14]. Although the dynamic penalty (DP) method uses evolutionary time to compute the corresponding penalty factors, it is still difficult to determine dynamic penalty functions appropriately [15]. The adaptive penalty (AP) method considers the feasibility ratios of sequential generations to determine penalties [8]. Nevertheless, the drawback of the AP method is the need to choose generational gap and coefficient adjustments.

This study attempts to answer the following questions: (1) Which constraints dominate an optimization problem? (2) What kind of fitness metrics can evaluate infeasible solutions in a meaningful way? (3) How can parameters be adjusted appropriately? Therefore, a novel penalty-adjustment method is proposed in this paper to guide genetic evolution for approaching the global optimal solution. Because the proposed method is inspired by Pawlak's rough set theory (RST) [24,25], it is named the rough penalty (RP) method, which serves as a new constraint-handling technique that aims to

* Tel.: +886 32655410; fax: +886 32655499.
E-mail address: linch@cycu.edu.tw

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.04.001

effectively apply the information granulation technique of RST in dealing with indiscernibility penalties. In this paper, the RP method is further incorporated with a GA, named RPGA, for solving constrained optimization problems with the following two goals: (1) adjust penalty coefficients according to their constraint violations and (2) analyze inefficient constraints by applying RST during evolution. The advantage of the proposed RPGA is that it can automatically adjust penalty coefficients for each optimization problem. Furthermore, the method does not require extra functional analysis to realize its solution space. The performance of the RPGA is evaluated by eleven well-known constrained problems. Experimental results show the proposed RPGA cannot only find optimal or close-to-optimal solutions but also obtain robust results for both linear and nonlinear constraint functions.

The remainder of this paper is organized as follows. Section 2 briefly describes several constraint-handling techniques. Section 3 introduces the proposed RST-based penalty function. Next, Section 4 describes the proposed RPGA and its genetic operations. In Section 5, we adopt Taguchi's quality-design method to analyze the best parameter setting for the proposed RPGA. Section 6 reports the computational results for 11 constrained optimization problems and comparisons with several well-known optimization algorithms. Finally, we summarize the findings and contributions of this study in Section 7.

2. Constraint handling techniques in evolutionary algorithms

Without a loss of generality, a general minimization problem with m constraints can be formulated as [30]

$$\text{minimize} \quad f(\vec{x}) = f(x_1, x_2, \ldots, x_n) \tag{1}$$

$$\text{subject to} \quad g_k(\vec{x}) \leqslant 0, \quad k = 1, \ldots, q \tag{2}$$

$$h_k(\vec{x}) = 0, \quad k = q+1, \ldots, m \tag{3}$$

$$\overrightarrow{B}_l \leqslant \vec{x} \leqslant \overrightarrow{B}_u, \tag{4}$$

where $f(\vec{x})$ is the objective function, $\vec{x} = (x_1, x_2, \ldots, x_n)$ is a vector of n decision variables, and $\overrightarrow{B}_l$ and $\overrightarrow{B}_u$ represent the lower bounds (LBs) $(B_{l1}, B_{l2}, \ldots, B_{ln})$ and the upper bounds (UBs) $(B_{u1}, B_{u2}, \ldots, B_{un})$ of all the variables, respectively. There are q inequality constraints $g_k(\vec{x})$ and $(m-q)$ equality constraints $h_k(\vec{x})$. A constrained optimization problem can be transformed into an unconstrained one by introducing penalty terms into the original objective function such that it becomes an extended objective function as in Eq. (5).

$$\psi(\vec{x}) = f(\vec{x}) + \sum_{k=1}^{m} (w_{tk} \times \max(0, \Phi_k(\vec{x}))^2), \tag{5}$$

where $\psi(\vec{x})$ is the expanded objective function. Penalty coefficients w_{tk} are adaptive with the magnitude of each constraint violation for the kth constraint in the tth generation. For a minimization problem, let

$$\Phi_k(\vec{x}) = \begin{cases} g_k(\vec{x}), & k = 1, \ldots, q \\ |h_k(\vec{x})| - \delta, & k = q+1, \ldots, m \end{cases} \tag{6}$$

where "$|\cdot|$" denotes an absolute operator, and the small tolerance (δ) is assigned to 0.0001 for equality constraints.

Generally, penalty-function approaches attempt to optimize the expended objective function and search the boundary between feasible and infeasible regions [7]. As infeasible solutions are penalized by Eq. (5), the search ability of EAs is significantly influenced by the values of the penalty terms [23]. A large penalty will reduce the net fitness of infeasible solutions. That discourages an EA to explore the infeasible region; thus, the EA may ignore solutions near the boundary of the feasible region. A low penalty misdirects an EA to explore the infeasible region because the penalty will be negligible with respect to its objective function [3]. Therefore, penalty coefficients should be adjusted according to the evolutionary situation [18,32].

Barbosa and Lemonge (denoted as BL) in 2003 [1] and Lemonge and Barbosa (abbreviated as LB) in 2004 [19] proposed two AP-based functions that assign different penalties to different constraints according to the average value of the objective function and the level of each constraint violation. Tessema and Yen (represented as TY) in 2006 [34] introduced a distance-based fitness function to the normalized fitness-constraint violation space. Two penalty values are applied to infeasible individuals to identify promising infeasible individuals and guide the search process toward finding the optimal solution. Farmani and Wright (indicated as FW) in 2003 [6] developed a two-stage penalty method in which the infeasibility values of an infeasibility solution are represented by its relationship to the worst, the best, and the highest values of the objective function in the current population. Although this method produced good results for most test functions, this two-stage penalty method has an unnecessarily high computational cost.

In this paper, the proposed RP method, which benefits from the concept of the RST and DP method, is utilized to adjust penalties based on their constraint violations. This proposed RPGA can achieve two primary advantages. First, parameter tuning in this work is self-adaptive. Second, the RPGA can obtain the global optimum starting from any infeasible solution.

3. Rough penalizing method

The rough set theory (RST) proposed by Pawlak in 1982 [24] is a mathematical approach for dealing with vagueness in which imprecision is expressed by a boundary region of solution space and imperfect searching is approximated by the main

components of rough set concepts to assist decision-making [20,26,27]. The RST has been employed to solve box-constrained multi-objective problems [9]. Santana-Quintero et al. combined hybrid EAs with RST as an effective way to approximate the Pareto front of a constrained multi-objective optimization problem [31].

3.1. Rough set theory (RST)

The primary concept of the RST is the indiscernibility relation, which is generated by the information of interested objects [25]. Because discerning knowledge is lacking, one cannot identify some objects based on the available information. The indiscernibility relation expresses this fact by considering granules of indiscernible objects as a fundamental basis. Some relevant concepts of the RST are as follows [17,24]:

Definition 1 (*Information system*). An information system (IS) is denoted as a triplet $T = (U,A,D)$, where U is a non-empty finite set of objects and A is a non-empty finite set of attributes. An information function D maps an object to its attribute, i.e., $D_a{:}U \to V_a \quad \forall a \in A$, where V_a is the domain of attribute a. A posteriori knowledge (denoted by d) is expressed by one distinguished attributed. A decision system is an IS with the form $DT = (U,A \cup \{d\},f)$, where $d \notin A$ is used as supervised learning. The elements of A are called conditional attributes.

Definition 2 (*Indiscernibility*). For an attribute set $B \subseteq A$, the equivalence relation induced by B is called a *B-indiscernibility relation*, i.e., $IND_T(B) = \{(x,y) \in U^2 \mid \forall a \in B, f_a(x) = f_a(y)\}$. The equivalence classes of the B-indiscernibility relation are denoted as $I_B(x)$.

Definition 3 (*Set Approximation*). Let $X \subseteq U$and $B \subseteq A$ in an IS T, the *B-lower approximation of X* is the set of objects that belongs to X with certainty, i.e., $\underline{B}X = \{x \in U \mid I_B(x) \subseteq X\}$. The *B-upper*is the set of objects that possibly belongs to X, where $\overline{B}X = \{x \in U | I_B(x) \cap X \neq \phi\}$.

Definition 4 (*Reducts*). If $X^1_{DT}, X^2_{DT}, \ldots, X^r_{DT}$ are the decision classes of DT, the set $POS_B(d) = \underline{B}X_1 \cup \cdots \cup \underline{B}\,X_r$ is the *B-positive region of DT*. A subset $B \subseteq A$ is a set of relative reducts of DT if and only if $POS_B(d) = POS_C(d)$ and $POS_{B-\{b\}}(d) \neq POS_C(d)$, $\forall b \in B$.

3.2. Rough penalty classification

The proposed RP method adjusts the penalty coefficients of each infeasible individual such that a relatively high penalty is assigned to the coefficient of the most difficult constraint. According to the extent of the constraint violation and evolution time, the RP introduces a penalty term to each violation. The constrained optimization problem introduced by Hock and Schittkowski [12] with four nonlinear inequality constraints is utilized as an illustration for the basic concepts of the RP method.

Example 1.

$$\begin{aligned}
\text{minimize} \quad & f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 \\
& + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 && (7)\\
\text{subject to} \quad & g_1(\vec{x}) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leqslant 0, && (8)\\
& g_2(\vec{x}) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \leqslant 0, && (9)\\
& g_3(\vec{x}) = 23x_1 + x_2^2 + 6x_6^2 - x_7 - 196 \leqslant 0, && (10)\\
& g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leqslant 0, && (11)\\
& -10 \leqslant x_i \leqslant 10 \quad \forall i = 1, \ldots, 7. && (12)
\end{aligned}$$

Table 1
The constraint violations and objective function values of six illustrative solutions.

Solution	Constraint violation				Objective function value (f)
	g_1	g_2	g_3	g_4	
$\vec{x}_1$	−24.8731	−274.733	−179.8143	−6.3632	957.3873
$\vec{x}_2$	−122.9181	−275.0301	−180.7736	−5.7018	1016.0817
$\vec{x}_3$	−25.1208	−274.7438	−179.1476	−7.6799	994.7475
$\vec{x}_4$	−122.9181	−275.0301	−180.7641	−5.7149	1016.0725
$\vec{x}_5$	−27.688	−274.0137	−179.8143	−6.3632	930.1447
$\vec{x}_6$	−15.1156	−251.9141	−183.334	9.3996	1061.0061

The UB and LB of constraint $g_k(\vec{x})$ are pre-calculated and denoted as UB_k and LB_k, respectively. Table 1 lists six illustrative solutions (denoted as $\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4, \vec{x}_5,$ and $\vec{x}_6$) with their objective function values and constraint violations.

This work uses information granulation as a key function for implementing a divide-and-conquer strategy. Elementary information granules are indiscernibility classes of constraint violations. The information system is an information table of attribute values containing rows labeled by objects and columns labeled by attributes [27].

Remark 1. To treat continuous-type attributes as discrete values, this work quantizes continuous-type attributes into regions and then allocates attributes according to their classes. A partition granularity (ρ) of classification is defined for constraint violation magnitude. The design principle is that solution quality increases as its constraint penalty moves closer to zero. Therefore, this study uses a smaller range in near-zero regions than in other regions.

Example 2. If the partition granularity (ρ) is 6, each constraint violation in Example 1 is divided into 6 regions between the UB and LB of each constraint violation. The ranges of Regions 1–6 for constraint g_k are defined as the intervals $[LB_k, (3/7)LB_k]$, $((3/7)LB_k, (1/7)LB_k]$, $((1/7)LB_k, 0]$, $(0, (1/7)UB_k]$, $((1/7)UB_k, (3/7)UB_k]$, and $((3/7)UB_k, UB_k]$, respectively. The concept is depicted in Fig. 1. The ranges of Regions 1–3 decrease exponentially from 2^2 to 2^0 because the violation level of an inactive constraint is inversely related to a negative constraint value. Conversely, the ranges of Regions 4–6 increase exponentially from 2^0 to 2^2. Table 2 shows the corresponding region for each constraint violation (g_k) in Example 1.

3.3. Rough decision system

A *decision system* is an IS with the form $DT = (U, A \cup \{d\}, D)$ in which each individual is treated as an object of a non-empty finite set U. Attribute set $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ is a non-empty finite set of attributes where each *penalty multiplier* (α_k) corresponds to a conditional attribute. The elements of A are called conditional attributes or conditions in RST. The supervised knowledge is expressed by a decision attribute (denoted by $d \notin A$). An information function D maps each object to a decision attribute, i.e., $D: U \to V_d$ for $d \notin A$, where the domain of decision attribute $d$is $V_d = \{0, 1\}$. The decision value of an object is "good" if its objective function value is above average, and otherwise its value is "bad." For a minimization problem, the information function is designed as follows:

$$D(\vec{x}_j) = d_j = \begin{cases} \text{good}, & \text{if } f(\vec{x}_j) < f_{average} \\ \text{bad}, & \text{if } f(\vec{x}_j) \geqslant f_{average} \end{cases} \text{ where } f_{average} = \frac{1}{p}\sum_{j=1}^{p} f(\vec{x}_j). \tag{13}$$

Remark 2. To construct a decision table, a penalty multiplier (α_k) is adjusted according to the region of its constraint violation. The principle is that a penalty multiplier increases exponentially as the violation level increases. Therefore, a penalty multiplier (α_k) is assigned as $(\alpha)^{-2}$, $(\alpha)^{-1}$, $(\alpha)^0$, $(\alpha)^{+1}$, $(\alpha)^{+2}$, and $(\alpha)^{+3}$ for constraint Regions 1, 2, 3, 4, 5, and 6, respectively.

Example 3. In this example, the average of these six objective function values, i.e., $f_{average} = (1/6)\sum_{j=1}^{6} f(\vec{x}_j) = 995.9066$, is applied to evaluate the decision value of each solution. According to the violation regions and the objective qualities in Table 2, the decision table of the six illustrative solutions is depicted in Table 3 while the coefficient (α) is set to 1.005.

3.4. Significant penalty and attribute reduction

The indiscernibility relation reduces the amount of required data by identifying equivalence classes with respect to the selected attributes. Because only elements in each equivalence class are necessary, redundant attributes can be removed without degrading the classification process. For any attribute set $B \subseteq A$, the *B-indiscernibility relation* is derived as

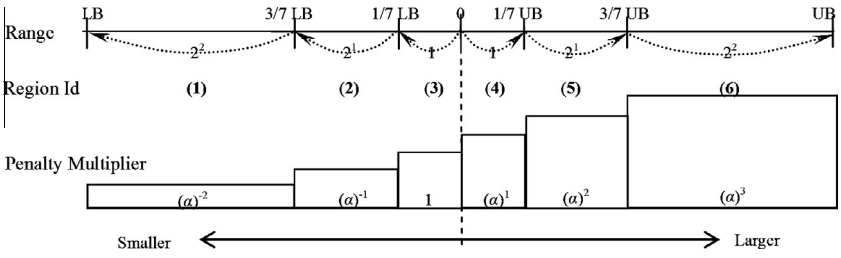

Fig. 1. Penalty multiplier classification.

Table 2
Information granulation of six illustrative solutions (No. of region = 6).

Solution	Violation region				Objective (quality)
	g_1	g_2	g_3	g_4	
$\vec{x}_1$	Region 3	Region 1	Region 2	Region 3	$f_1 < f_{mean}$
$\vec{x}_2$	Region 1	Region 1	Region 2	Region 3	$f_2 > f_{mean}$
$\vec{x}_3$	Region 3	Region 1	Region 2	Region 3	$f_3 < f_{mean}$
$\vec{x}_4$	Region 1	Region 1	Region 2	Region 3	$f_4 > f_{mean}$
$\vec{x}_5$	Region 2	Region 1	Region 2	Region 3	$f_5 < f_{mean}$
$\vec{x}_6$	Region 3	Region 1	Region 2	Region 4	$f_6 > f_{mean}$

Table 3
Decision table for Example 3.

Solution	α_1	α_2	α_3	α_4	Decision
$\vec{x}_1$	1	0.9901	0.995	1	Good
$\vec{x}_2$	0.9901	0.9901	0.995	1	Bad
$\vec{x}_3$	1	0.9901	0.995	1	Good
$\vec{x}_4$	0.9901	0.9901	0.995	1	Bad
$\vec{x}_5$	0.995	0.9901	0.995	1	Good
$\vec{x}_6$	1	0.9901	0.995	1.005	Bad

$IND(B) = \{(x,y) \in U^2 | \forall a \in B, a(x) = a(y)\}$. The relation $(x,y) \in IND(B)$ means that objects x and y are indiscernible by their selected attribute set.

A reduct of DT is a minimal set of attributes, $B \subseteq A$, which discerns objects in the same manner used for the full set of considered objects $IND(B) = IND(A)$. That is, a reduct is a minimal set of attributes from A that preserves the universe partitioning and can generate the same partition in the domain of the decision system. Although computing equivalence classes is straightforward, finding a minimal reduct (i.e., reduct with a minimal cardinality of attributes among all reducts) has been proven to be an NP-hard problem [16]. Reducts have been appropriately characterized by a *discernibility matrix* $M(DT)$, which is a symmetrical matrix with entries $m_{ij} = \{a \in A | a(\vec{x}_i) \neq a(\vec{x}_j)\} \quad \forall i,j = 1,\ldots,n$.

Example 4. The decision system expresses all knowledge about the decision table in Table 3. All equivalence classes in the relation $IND(A)$ are $U/IND(\{a_1,a_2,a_3,a_4\}) = \{\{\vec{x}_1,\vec{x}_3\},\{\vec{x}_5\},\{\vec{x}_2,\vec{x}_4\},\{\vec{x}_6\}\}$. The partition created by the equivalence relation can build new subsets of the universe. Table 4a is the discernibility matrix of the decision table (Table 3). The entries in Table 4a are the set of attributes from the entire attribute set A that can discern each class in $U/IND(\{a_1,a_2,a_3,a_4\})$.

By referring to decision attribute d of DT, this study constructs a decision-relative discernibility matrix $M^d(DT) = (m^d_{ij})$ assuming $m^d_{ij} = \phi$ if $d(\vec{x}_i) = d(\vec{x}_j)$, or $m^d_{ij} = m_{ij}$ otherwise. Differing from the discernibility matrix, entries in matrix $M^d(DT)$ are empty for objects that have the same decision value. Therefore, the corresponding decision-relative discernibility matrix $M^d(DT)$ of the decision table is depicted in Table 4b.

Several studies have shown that a rough reduct can increase classification accuracy for test data when only rules constructed by such relevant features are considered [16]. Based on the concept of attribute reduction, attributes may not be equally important, and some of them can be eliminated from a decision table without degrading information quality. Attribute reduction can be generalized by introducing attribute evaluation, which can express the merit of each attribute in the information table.

Remark 3. According to the reduct definition of RST, the minimal subset of penalized constraints is applied to distinguish above-average individuals (i.e., their decision values are "good") and below-average ones (i.e., their decision values are "bad"). Decision attribute d in DT also determines a partition $CLASS(d)$ of object set U, where $CLASS(d)$ is the object

Table 4
Two kinds of symmetrical discernibility matrices for Example 4.

m_{ij}	$\vec{x}_1$	$\vec{x}_3$	$\vec{x}_5$	$\vec{x}_2$	$\vec{x}_4$	$\vec{x}_6$
(a) Discernibility matrix						
$\vec{x}_1$	$\varnothing$					
$\vec{x}_3$	$\varnothing$	$\varnothing$				
$\vec{x}_5$	a_1	a_1	$\varnothing$			
$\vec{x}_2$	a_1	a_1	a_1	$\varnothing$		
$\vec{x}_4$	a_1	a_1	a_1	$\varnothing$	$\varnothing$	
$\vec{x}_6$	a_4	a_4	a_1, a_4	a_1, a_4	a_1, a_4	$\varnothing$

m^d_{ij}	$\vec{x}_1$	$\vec{x}_3$	$\vec{x}_5$	$\vec{x}_2$	$\vec{x}_4$	$\vec{x}_6$
(b) Decision-relative discernibility matrix						
$\vec{x}_1$	$\varnothing$					
$\vec{x}_3$	$\varnothing$	$\varnothing$				
$\vec{x}_5$	$\varnothing$	$\varnothing$	$\varnothing$			
$\vec{x}_2$	a_1	a_1	a_1	$\varnothing$		
$\vec{x}_4$	a_1	a_1	a_1	$\varnothing$	$\varnothing$	
$\vec{x}_6$	a_4	a_4	a_1, a_4	$\varnothing$	$\varnothing$	$\varnothing$

classification with respect to decision attribute d. The features of dynamic reducts are relevant attributes of each decision-relative class. The representative value of each relevant attribute is assigned as the attribute value with the maximum cardinality in the same class.

Example 5. The classified partitions for the decision table (in Table 3) are $U = X_{\text{good}} \cup X_{\text{bad}}$, where $X_{\text{good}} = \{\vec{x}_j \in U | d(\vec{x}_j) = \text{good}\} = \{\vec{x}_1, \vec{x}_3, \vec{x}_5\}$ and $X_{\text{bad}} = \{\vec{x}_j \in U | d(\vec{x}_j) = \text{bad}\} = \{\vec{x}_2, \vec{x}_4, \vec{x}_6\}$. For instance, the cardinalities of attribute value a_1 in class X_{good} are that $card(X_{\text{good}}^{a_1=1}) = card\{\vec{x}_1, \vec{x}_3\} = 2$ and $card(X_{\text{good}}^{a_1=0.995}) = card\{\vec{x}_5\} = 1$. Therefore, the representative value of attribute a_1 in class X_{good} is 1. For classes X_{good} and X_{bad}, the representative values of the attribute vectors are denoted as vectors $\vec{\gamma}$ and $\vec{\beta}$, respectively. Based on Table 3, $\vec{\gamma} = (\gamma_1, \gamma_2, \gamma_3, \gamma_4) = (1, 0.9901, 0.995, 1)$ for class X_{good}, and $\vec{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4) = (0.9901, 0.9901, 0.995, 1)$ for class X_{bad}, as shown in Table 5a. Furthermore, Table 5b presents the discernibility matrix of decision-relative classes. For simplification, the attribute set that can differentiate between classes X_{good} and X_{bad} is presented as a discernible mask $\vec{\mu} = (\mu_1, \mu_2, \mu_3, \mu_4) = (1, 0, 0, 0)$. If the kth constraint is discernible, μ_k is set to 1. That is, individual qualities can be differentiated by considering the functional value of the kth constraint. Otherwise, the ineffective constraint is not a differential feature, and μ_k is set to 0.

3.5. Self-adaptive penalty adjustment

The proposed RP method adjusts penalty terms according to both violation levels and evolution time to solve constrained optimization problems effectively. Each individual in generation t is evaluated using an expanded objective function (Eq. (14)):

$$\psi(\vec{x}) = f(\vec{x}) + \sum_{k=1}^{m}((C \times t)^{\pi(k,t)} \times \max(0, \Phi_k(\vec{x}))^2), \tag{14}$$

where C is a "severity" factor, m is the total number of constraints, and Φ_k is the violation level of constraint k. This fitness function combines a*coefficient* $(C \times t)$ with an*exponent* $\pi(k,t)$ to increase penalty pressure over time. For constraint k in generation t, the exponent $\pi(k,t)$ is a representative penalty multiplier that is initially assigned as 2 and then is tuned iteratively according to the discernible mask $\vec{\mu}$ and the representative attribute value γ_k of superior class X_{good}. The exponent $\pi(k,t)$ is defined as

$$\pi(k,t) = \begin{cases} \pi(k,t-1) \times \gamma_k, & \text{if } \mu_k = 1 \\ \pi(k,t-1), & \text{if } \mu_k = 0 \end{cases} \quad \forall k = 1, \ldots, m; \quad \forall t = 1, \ldots, MaxGeneration \tag{15}$$

$$\pi(k,0) = 2 \quad \forall k = 1, \ldots, m. \tag{16}$$

Remarkably, the discernible mask $\vec{\mu}$ can be used to enable $\pi(k,t)$ by differentiating significant characteristics between classes X_{good} and X_{bad}. If the k^{th} constraint is discernible (i.e., $\mu_k = 1$), the exponent $\pi(k,t)$ is adjusted by the representative attribute value (γ_k); otherwise, the exponent retains the same value as in the previous generation. The pseudo-code of the RP method is depicted in Fig. 2.

The proposed RP method not only accelerates feasible space exploitation by penalizing constraint violations but also enhances exploration ability by releasing the penalties of ineffective constraints during evolution. Therefore, the RP method is a self-adaptive approach that can measure infeasibility and can adjust each penalty coefficient automatically.

4. Rough penalty genetic algorithm

The proposed RPGA adopts the RP method to enhance the exploration and exploitation abilities of original GAs for handling constrained optimization problems. The flow chart of RPGA (in Fig. 3) consists of several genetic operations, such as initialization, selection, crossover, mutation, and replacement.

4.1. Population initialization

Decision variables of a test function are represented as a double vector data type to form a chromosome $\vec{x} = (x_1, x_2, \ldots, x_n) \in \Re^n$, where x_i is the value of the ith variable. The RPGA starts with a random population within the prob-

Table 5
Decision-relative equivalence classes and discernibility matrix for Example 5.

Class (d)	Notation	a_1	a_2	a_3	a_4	Attribute ($\vec{\mu}$)	X_{Good}	X_{Bad}
(a) Decision-relative equivalence classes						*(b) Discernibility matrix*		
X_{Good}	$\vec{\gamma}$	1	0.9901	0.995	1	X_{Good}	$\varnothing$	
X_{Bad}	$\vec{\beta}$	0.9901	0.9901	0.995	1	X_{Bad}	a_1	$\varnothing$

```
conN is mean constraint numbers
popnum is population numbers
IT is information table, which is (conN+1)* popnum matrix
gp is good population which decision variable =1 from IT
bp is bad population which decision variable =0 from IT
gc_k is good characteristic from gp
bc_k is bad characteristic from bp
t is the number of current generation
π_k(t) is a RP exponent for the t generation
λ_k is a RP coefficient
C is constraints weight
τ is next generation penalty coefficient
Begin
    Create information table IT
    //Divide IT to two group gp and bp
    For i = 1 to popnum
        IF decision variable =1  Pick IT(i) to good population gp
        Else  Pick IT(i) to bad population bp
        End if
    End for
    //Find characteristic form gp and bp
  For k=1 to conN
        gc_k=mod all gp_k
        bc_k=mod all bp_k
    End for
    //Find RP coefficient μ
  For k =1 to conN
        If gc_k=bc_k  λ_k= gc_k
        Else  λ_k=1
        End if
    End for
    //Modify RP penalty exponent π(k,t)
    For k =1 to conN
        π_k(t) =π_k (t-1)* λ_k
        τ_k= (Ct)^ π_k(t) , for all k=1 to conN
    End for
End begin
```

Fig. 2. Pseudo-code of the RP method.

lem-defined solution space $[\vec{B}_l, \vec{B}_u]$, no matter whether these individuals in the constrained problem are feasible or not. The emphasis is that the RPGA can find the global optimum for constrained problems even though the initial population is infeasible.

4.2. Selection operation

A selection operation uses fitness to determine the solution quality and to select high-quality chromosomes for the recombination operation [42]. The RPGA employs a stochastic universal selection to create selection pressure toward the global optimal solution. The measurement of a chromosome's fitness is its value of the expanded objective function in Eq. (14).

4.3. Therapeutic crossover operation

A novel therapeutic crossover is proposed to incorporate a gene-therapy method with a conventional uniform crossover scheme. The therapeutic crossover gives each locus an equal chance of being a crossover point. A random number distributed uniformly in interval [0, 1] is generated for each locus in a chromosome. If the random number is less than a pre-defined therapeutic rate$p$, this gene locus belongs to the therapeutic genome (i.e., $i \in G_c$ where $G_c \in \{1, 2, \ldots, N\}$); otherwise, no crossover occurs at this locus (i.e., $i \notin G_c$).

Each time the selection operation chooses two crossover parents from the population of generation t. Let the parent with superior fitness be the best parent $\vec{x}_b(t)$ and the other is the worst parent $\vec{x}_w(t)$. The proposed gene-therapy method measures the merit of two selected genes by comparing the changes in chromosome fitness before and after interchanging the genes

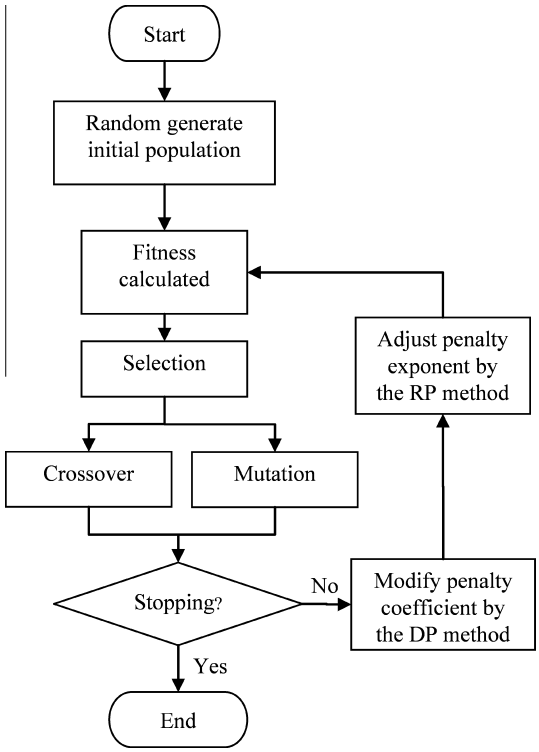

Fig. 3. Flow chart of the RPGA.

with the other mating chromosome. A temporary chromosome is generated by cloning genes from one parent and replacing the therapy gene with that in the other parent. For example, the temporary chromosome for a therapeutic locus $i \in G_c$ is $\vec{x}_{ri} = (x_{b1}, \ldots, x_{b(i-1)}, x_{wi}, x_{b(i+1)}, \ldots, x_{bN})$. Comparing the fitness change before and after the genome replacement (i.e., $f_b = \psi(\vec{x}_b)$ vs. $f_{wi} = \psi(\vec{x}_{ri})$) can determine the substitution effect and can be used to represent the relative merit of the genes x_{bi} and x_{wi} with respect to the genetic material of parent $\vec{x}_b$.

According to their relative merit, these two genes in both parents linearly combine to generate a new gene for their offspring. The principle of this linear combination is to retain the favorable schemata in the evolution process. Therefore, offspring inherit more genetic material from the better gene than from the worse one. The crossover operation also keeps some genetic material from the other gene to enhance population diversity. These two therapeutic genes (i.e., x_{bi} and x_{wi}) can be reproduced by Eq. (17) to generate a new offspring gene [21].

$$x_{ci} = \begin{cases} x_{bi}, & \text{if } (i \notin G_c) \\ x_{bi} \times coef + x_{wi} \times (1 - coef), & \text{if } (i \in G_c) \text{ and } (f_b \text{ dominates or equals to } f_{wi}) \\ x_{bi} \times (1 - coef) + x_{wi} \times coef, & \text{if } (i \in G_c) \text{ and} (f_b \text{ is dominated by } f_{wi}) \end{cases} \tag{17}$$

$$coef = 1.0 - \frac{randn}{5}\left(1 - \frac{current\_generation}{total\_generation}\right) \tag{18}$$

Therefore, the characteristics of the proposed therapeutic crossover are as follows: (1) the quality of each gene is determined individually; (2) genetic diversity is ensured by keeping promising genes in inferior chromosomes; and (3) some defective genomes in the superior parent are cured using the genetic material from the other parent.

4.4. Mutation operation

A mutation operation used in GAs can increase population diversity to enhance its exploration ability [10]. To balance exploration ability and convergence ability, this study combines traditional uniform with Gaussian mutations as a two-stage mutation that can adapt automatically based on the convergent situation. When fitness values over an entire population highly fluctuate, a traditional uniform mutation is employed to explore the search space. Otherwise, a Gaussian mutation is adopted to concentrate on exploiting potential optimal areas and to speed up the convergent effect. Furthermore, the proposed adaptive mutation adjusts a mutation rate for each chromosome with respect to the fitness distribution of its population. An individual with above-average performance has a smaller mutation rate than those that perform below average.

4.5. Reproduction operation and evolutionary termination

The proposed RPGA adopts a replacement-with-elitism method to prevent best solutions from being lost through a selection process. A successive population is produced from three sources: (1) the replacement-with-elitism method selects the best three chromosomes into the next generation; (2) the crossover operation recombines 80% of child chromosomes; and (3) the mutation operation constructs other child chromosomes.

5. Experimental analyses and parameter setting

5.1. Test constrained optimization problems

In this paper, the proposed RPGA is evaluated by solving eleven well-known benchmark constrained functions described in the Appendix. These test cases have various test functions, which involve both maximization and minimization problems with different objective functions (e.g., linear, quadratic, cubic, nonlinear, or polynomial functions) and constraints (e.g., linear inequality (LI), nonlinear inequality (NI), and nonlinear equality (NE)). Table 6 lists the characteristics of these test functions, such as the number of variables (n), objective function type, number of each constraint type, number of active constraints in the optimum solution, and the feasibility ratio. Without a loss of generality, the maximization problems (i.e., g02, g03, and g08) in the 11 benchmark functions are transformed into minimization problems using $-f(x)$. The nonlinear inequality constraints in three test functions (i.e., g03, g05, and g11) are converted into inequality constraints as $|h_k(\vec{x})| - \delta \leqslant 0$, where the tolerance of violation (δ) is a small value ($\delta = 0.0001$). The feasibility ratio and constraint types can be used to define the difficulty of each constraint-satisfaction problem.

5.2. Taguchi-based experimental design

It is well known that the performance of GAs significantly depends on the configuration of the operating parameters. To investigate the impact of various parameter settings on the performance, this study applies the Taguchi method to systematically find the optimal setting of the operating parameters [2]. The Taguchi method proposed by Taguchi and Konishi in 1987 is a robust design technique for quality engineering that improves the quality of solutions and minimizes the number of required experiments simultaneously [33,40]. The Taguchi method consists of two tools: (1) signal-to-noise ratio (SNR) for measuring quality and (2) orthogonal arrays for determining which combination of factor levels is suitable for each experimental run.

In this paper, the Taguchi method consists of the following steps: (1) specify the settings of the operating parameters; (2) determine the levels for each parameter; (3) select a suitable orthogonal array; and (4) conduct experiments on all the constrained problems to find the optimal parameter setting for each problem.

In the proposed RPGA, the Taguchi method is used to experiment on four operating parameters: two parameters of the RP method (penalty range and initial penalty) and two parameters for the therapeutic crossover (crossover rate and therapeutic rate). The value of each parameter can be divided into three levels, as listed in Table 7. A standard $L_9(3^4)$ orthogonal array (in Table 8) is used as the Taguchi matrix to study the cross-impact of four 3-level factors. The Taguchi experiments operate on eleven test functions with 30 independent runs. After the Taguchi experiments are complete, the optimal settings for all the test problems are identified, and the settings of the four parameters are shown in Table 9. The detailed process of the Taguchi experiments can also be found in the work of Tsai et al. [35].

5.3. Parameter settings

In all cases, up to 350,000 fitness function evaluations (FFEs) are conducted before stopping the experiment. Therefore, the total generation in this paper is up to 350,000/($PopulationSize \times (1 + CrossoverRate)$) because the therapeutic crossover

Table 6
Eleven benchmark constrained functions [29,34].

Function	n	Type of function	LI	NE	NI	Active Constraint	Feasibility Ratio (%)
g01	13	Quadratic	9	0	0	6	0.0003
g02	20	Nonlinear	1	0	1	1	99.9965
g03	10	Nonlinear	0	1	0	1	0.0000
g04	5	Quadratic	0	0	6	2	26.9356
g05	4	Nonlinear	2	3	0	3	0.0000
g06	2	Nonlinear	0	0	2	2	0.0064
g07	10	Quadratic	3	0	5	6	0.0003
g08	2	Nonlinear	0	0	2	0	0.8640
g09	7	Nonlinear	0	0	4	2	0.5256
g10	8	Linear	3	0	3	3	0.0005
g11	2	Quadratic	0	1	0	1	0.0000

Table 7
Corresponding values of four 3-level factors for Taguchi experiment.

Value	Factor A (penalty range)	Factor B (initial penalty)	Factor C (crossover rate)	Factor D (therapeutic rate)
Level 1	0.85	5	0.6	0.2
Level 2	0.9	50	0.7	0.3
Level 3	0.95	500	0.8	0.4

Table 8
Orthogonal array L_9 (3^4) for Taguchi experiment.

Experiment number	Factors			
	A	B	C	D
1	Level 1	Level 1	Level 1	Level 1
2	Level 1	Level 2	Level 2	Level 2
3	Level 1	Level 3	Level 3	Level 3
4	Level 2	Level 1	Level 2	Level 3
5	Level 2	Level 2	Level 3	Level 1
6	Level 2	Level 3	Level 1	Level 2
7	Level 3	Level 1	Level 3	Level 2
8	Level 3	Level 2	Level 1	Level 3
9	Level 3	Level 3	Level 2	Level 1

Table 9
Optimal settings of the four factors for the 11 test functions.

Optimal level	Factor A (penalty range)	Factor B (initial penalty)	Factor C (crossover rate)	Factor D (therapeutic rate)
Function g01	Level 2	Level 1	Level 1	Level 3
Function g02	Level 2	Level 3	Level 3	Level 3
Function g03	Level 1	Level 3	Level 2	Level 3
Function g04	Level 2	Level 2	Level 3	Level 3
Function g05	Level 3	Level 3	Level 2	Level 3
Function g06	Level 1	Level 1	Level 3	Level 3
Function g07	Level 3	Level 3	Level 2	Level 3
Function g08	Level 2	Level 1	Level 1	Level 3
Function g09	Level 2	Level 3	Level 2	Level 2
Function g10	Level 3	Level 2	Level 2	Level 3
Function g11	Level 1	Level 2	Level 2	Level 1

Table 10
Experimental results obtained by the RPGA over 30 independent runs.

Function	Optimal	Best	Median	Mean	Worst	St. Dev.	Infeasible Run	G_m	Gap (%)
g01	−15.000	**−15.000**	**−15.000**	**−15.000**	**−15.000**	0	0	500	0
g02	−0.803619	−0.803612	−0.794644	−0.794453	−0. 780,826	0.008188437	0	1400	8.710E−04
g03	−1.000	**−1.000**	**−1.000**	**−1.000**	**−1.000**	8.83097E−05	0	1400	0
g04	−30665.539	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	2.07846E−05	0	1000	0
g05	5126.498	5126.544	5233.654	5352.188	5888.510	246.1587486	0	1400	8.973E−04
g06	−6961.814	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	1.0452E−11	0	1400	0
g07	24.306	24.333	24.392	24.387	24.427	0.02801179	0	1400	1.110E−01
g08	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	2.14065E−17	0	100	0
g09	680.630	680.631	680.634	680.634	680.637	0.001665788	0	1400	1.469E−04
g10	7049.248	7049.861	7104.898	7131.084	7263.461	67.24232044	0	1400	8.696E−03
g11	0.750	**0.749**[a]	**0.749**[a]	**0.749**[a]	**0.749**[a]	1.16538E−07	0	1000	1.333E−01

[a] The best result was even better than the optimal solution because of using inequalities to approximate each equality constraint.

operation employs extra function evaluations for gene-therapy purposes. This evaluation budget is similar to several existing studies [35]. The population size is 200, the number of elite individuals is 3, and the severity factor of the RP method in Eq. (14) is 50. For each test problem, 30 independent runs with different seeds are performed using the MATLAB environment.

6. Computational results and comparisons

The experimental results obtained by the RPGA are shown in Table 10. The optimal results known to date for all the test problems are obtained from Ref. [29]. In Table 10, the experimental results include the best, median, mean, and worst

Table 11
Comparison of the best experimental results with other penalty-function methods.

Function	Optimum Value	BL [1]	FW [6]	LB [19]	TY [34]	RPGA
g01	−15.000	**−15.0**	**−15.0000**	**−15.00**	**−15.000**	**−15.000**
g02	−0.803619	NA	−0.802970	−0.7724640	−0.803202	−0.803612
g03	−1.00	**−1.0010096**	**−1.0000**	**−0.9939078**	**−1.000**	**−1.000**
g04	−30665.539	−30665.403	−30665.500	−30665.24	−30665.401	**−30665.539**
g05	5126.498	5126.868508	5126.9890	5126.571	5126.907	5126.544
g06	−6961.814	NA	−6961.800	−6961.796	−6961.046	**−6961.814**
g07	24.306	NA	24.480	24.86371	24.838	24.333
g08	−0.095825	**−0.0958250**	**−0.095825**	**−0.0958250**	**−0.095825**	**−0.095825**
g09	680.630	680.6668170	680.64	680.7590	680.773	680.631
g10	7049.248	NA	7061.34	7086.404	7069.981	7049.861
g11	0.750	**0.749**	**0.750**	**0.750**	**0.749**	**0.749**

Table 12
Comparison of the mean experimental results with other penalty-function methods.

Function	Optimum Value	BL [1]	FW [6]	LB [19]	TY [34]	RPGA
g01	−15.000	**−15.0**	**−15.000**	**−15.00**	−14.552	**−15.000**
g02	−0.803619	NA	−0.79010	−0.7031971	−0.755798	−0.794453
g03	−1.00	**−0.9937453**	**−0.9999**	−0.9757277	−0.964	**−1.000**
g04	−30665.539	−30662.388	−30665.2	−30663.40	−30659. 221	**−30665.539**
g05	5126.498	5458.360696	5432.08	5389.364	5214.232	5352.188
g06	−6961.814	NA	−6961.8	−6961.789	−6953.061	**−6961.814**
g07	24.306	NA	26.580	29.86465	27.328	24.387
g08	−0.095825	−0.0957742	**−0.095825**	−0.0926157	−0.095635	**−0.095825**
g09	680.630	681.2387964	680.72	681.4076	681.246	680.634
g10	7049.248	NA	7627.89	8161.997	7238.964	7131.084
g11	0.750	**0.7490065**	**0.75**	**0.7503349**	0.751	**0.749**

Table 13
Comparison of the worst experimental results with other penalty-function methods.

Function	Optimum value	BL [1]	FW [6]	LB [19]	TY [34]	RPGA
g01	−15.000	**−15.0**	**−15.000**	**−15.00**	−13.097	**−15.000**
g02	−0.803619	NA	−0.76043	−0.6002978	−0.745712	−0.780826
g03	−1.00	−0.9805778	**−0.9997**	−0.9391756	−0.887	**−1.000**
g04	−30665.539	−30655.388	−30663.30	−30660.76	−30656. 471	**−30665.539**
g05	5126.498	5992.370532	6089.4300	6040.595	5564.642	5888.510
g06	−6961.814	NA	−6961.800	−6961.779	−6943.304	**−6961.814**
g07	24.306	NA	28.40	42.01618	33.095	24.427
g08	−0.095825	−0.0954687	**−0.095825**	−0.0725015	−0.092697	**−0.095825**
g09	680.630	682.2892937	680.87	682.1562	682.081	680.637
g10	7049.248	NA	8288.79	10002.93	7489.406	7263.461
g11	0.750	**0.7490317**	**0.750**	0.7579745	0.757	**0.749**

objective function values in 30 independent runs. The results also list the standard deviations, the number of infeasible runs, and the median of the final generation number (G_m) in 30 trials. The solution quality is measured by the gap between the "known" optimal solution and the best result, i.e., [(best result-optimal solution)/optimal solution] $\times$ 100% for minimization problems. The exact/near optimal solutions found by algorithms are highlighted in **boldface**.

The first thing we can observe from Table 10 is the number of infeasible runs for all eleven functions is zero. That is, the proposed RPGA successfully finds feasible solutions in all 30 runs for all test functions even though the feasibility ratios of most of the test functions are less than 1% and the RPGA starts with infeasible individuals.

Secondly, the proposed algorithm obtained the "known" optimal solutions for six of the eleven test functions (g01, g03, g04, g06, g08, and g11). The results are exactly equal to the "known" optimal solutions in Table 10. The best result for the g11 problem was even better than its optimal solution because each equality constraint in g11 was approximated by two inequalities with a tolerance of 0.0001. Notably, all the best, median, mean, and worst experimental results for functions g01, g03, g04, g06, g08 and g11 were the same as the optimal values, revealing that the proposed RPGA can consistently converge on the optimal solutions in 30 independent runs.

Finally, the RPGA found values near the optimal solutions for the remaining five test functions (g02, g05, g07, g09, and g10). The obtained results approach the "known" optimal values with small differences from 0.00089% to 0.13%, which demonstrate the proposed algorithm can obtain near optimal solutions for these five optimization functions with several nonlinear and quadratic constraints.

Table 14
Comparison of the best experimental results with search-bias and multi-objective optimization methods.

Function	Optimum value	Search-bias methods			Multiobjective optimization methods				RPGA
		SR [29]	ISR [30]	HS [11]	TC [5]	YK [41]	VY [36]	ATMES [39]	
g01	−15.000	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−14.9999**	**−15.000**	**−15.000**
g02	−0.803619	−0.803515	**−0.803619**	−0.803602	−0.803615	−0.803503	−0.803190	−0.803388	−0.803612
g03	−1.00	**−1.000**	**−1.001**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**
g04	−30665.539	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	−30665.5312	**−30665.539**	**−30665.539**
g05	5126.498	**5126.497**	**5126.497**	**5126.499**	**5126.498**	**5126.497**	5126.5096	**5126.498**	5126.544
g06	−6961.814	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	−6961.1785	**−6961.814**	**−6961.814**
g07	24.306	24.307	**24.306**	24.307	24.740	24.307	24.410977	**24.306**	24.333
g08	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
g09	680.630	**680.630**	**680.630**	680.632	680.631	**680.630**	680.762228	**680.630**	680.631
g10	7049.248	7054.316	**7054.248**	7063.312	7080.265	7050.965	7060.55288	7052.253	7049.861
g11	0.750	**0.750**	**0.750**	**0.750**	**0.75**	**0.750**	**0.7490**	**0.75**	**0.749**

Table 15
Comparison of the mean experimental results with search-bias and multi-objective optimization methods.

Function	Optimum value	Search-bias methods			Multiobjective optimization methods				RPGA
		SR [29]	ISR [30]	HS [11]	TC [5]	YK [41]	VY [36]	ATMES [39]	
g01	−15.000	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	−14.997	**−15.000**	**−15.000**
g02	−0.803619	−0.781975	−0.782715	−0.777351	−0.796036	−0.790615	−0.755332	−0.790148	−0.794453
g03	−1.00	**−1.000**	**−1.001**	**−1.000**	**−1.000**	−0.999	−0.94899	**−1.000**	**−1.000**
g04	−30665.539	**−30665.539**	**−30665.539**	−30665.228	−30665.531	**−30665.539**	−30663.3642	**−30665.539**	**−30665.539**
g05	5126.498	5128.881	**5126.497**	5473.997	5288.127	5128.664	5170.5294	5127.648	5352.188
g06	−6961.814	−6875.940	**−6961.814**	−6896.354	**−6961.814**	−6894.914	−6959.5683	**−6961.814**	**−6961.814**
g07	24.306	24.374	**24.306**	24.417	25.987	24.323	26.735666	24.316	24.387
g08	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	−0.095825	**−0.095825**	**−0.095825**
g09	680.630	680.656	**680.630**	680.663	680.663	680.635	681.706290	680.639	680.634
g10	7049.248	7559.192	7049.250	7365.964	7892.470	7181.072	7723.16672	7250.437	7131.084
g11	0.750	**0.75**	0.756	0.827	**0.75**	**0.750**	0.7493	**0.75**	**0.749**

Table 16
Comparison of the worst experimental results with search-bias and multi-objective optimization methods.

Function	Optimum value	Search-bias methods			Multiobjective optimization methods				RPGA
		SR [29]	ISR [30]	HS [11]	TC [5]	YK [41]	VY [36]	ATMES [39]	
g01	−15.000	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	−11.999	**−15.000**	**−15.000**
g02	−0.803619	−0.726288	−0.723591	−0.712177	−0.777435	−0.740292	−0. 672169	−0.756986	−0.780826
g03	−1.00	**−1.000**	**−1.001**	**−1.000**	**−1.000**	**−0.998**	−0.785582	**−1.000**	**−1.000**
g04	−30665.539	**−30665.539**	**−30665.539**	−30659.007	−30663.829	**−30665.539**	−30651.9595	**−30665.539**	**−30665.539**
g05	5126.498	5142.472	**5126.497**	6080.091	5562.850	5150.440	6112.2231	5135.256	5888.510
g06	−6961.814	−6350.262	**−6961.814**	−6566.977	**−6961.814**	−6453.066	−6954.3186	**−6961.814**	**−6961.814**
g07	24.306	24.642	**24.306**	25.004	27.659	24.379	35.88193	24.359	24.427
g08	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
g09	680.630	680.763	**680.630**	680.863	680.707	680.648	684.131429	680.673	680.637
g10	7049.248	8835.655	7049.270	8220.442	9847.039	7526.509	12097.4078	7560.224	7263.461
g11	0.750	**0.75**	0.774	0.957	**0.75**	**0.750**	0.8094	**0.75**	**0.749**

6.1. Comparison with other penalty-function methods

The performance of the RPGA is compared with four state-of-the-art penalty-function methods, denoted as BL [1], FW [6], LB [19], and TY [34]. The comparison of the best experimental results for the 11 constrained problems is shown in Table 11. Obviously, the proposed RPGA found the exact optimal solutions for six of the eleven test functions (g01, g03, g04, g06, g08, and g11); however, the other four algorithms optimally solved only four functions (g01, g03, g08, and g11). For the remaining five test functions (g02, g05, g07, g09, and g10), the RPGA also obtained better solutions than the other four methods.

Tables 12 and 13 are the "mean" and "worst" experimental results obtained by these five algorithms, respectively. Both the "mean" and "worst" experimental results obtained by the proposed RPGA are better than the results of the other four methods on all the test functions except g05, for which the TY algorithm performed slightly better than the RPGA. However, the search ability of the RPGA is more consistent than the TY algorithm with respect to all test functions. That is, the proposed RPGA outperformed the four penalty-function methods for these eleven test functions.

6.2. Comparison with search-bias and multi-objective optimization methods

The performance of the RPGA is also compared with seven other state-of-the-art algorithms: four search-bias algorithms and three multi-objective optimization methods. The four search-bias algorithms are the Stochastic Ranking (SR) in 2000 [29], Improved SR (ISR) in 2005 [30], Ho & Shimizu (HS) in 2007 [11], and Triangular Crossover (TC) in 2008 [5]. The three multi-objective optimization algorithms are Yuchi and Kim (YK) in 2004 [41], Venkatraman and Yen (VY) in 2005 [36], and the Adaptive Tradeoff Model with Evolutionary Strategy (ATMES) in 2008 [39].

Table 14 shows the "best" experimental results obtained by the proposed RPGA and the seven algorithms. The ISR outperformed the other three search-bias algorithms, and the ATMES was the best multi-objective optimization algorithm for these constrained problems. The RPGA was superior to the VY algorithm on almost all the problems. The RPGA, SR, HS, TC, and YK algorithms can find all the optimal solutions for six test functions (g01, g03, g04, g06, g08, and g11). The best solutions obtained by the RPGA are similar to those of the SR, HS, TC, and YK algorithms for the g02, g07, and g09 test functions.

Tables 15 and 16 show the comparison of the "mean" and "worst" experimental results of 30 independent runs, respectively. The proposed algorithm achieved superior "mean" and "worst" results than five algorithms (SR, HS, TC, YK, and VY) in all the test problems. The RPGA also performed better than the ATMES algorithm for all but 2 test functions, g05 and g07.

For the standard deviations in 30 independent runs, the proposed algorithm achieved superior results compared with the SR, HS, YK, and VY algorithms in all the test problems. The proposed RPGA achieved better standard deviations than those obtained by the TC for all the problems except for g05. Although the RPGA had slightly worse standard deviations for functions g05 and g07 than the ATMES, the proposed RPGA outperformed the ATMES for the remaining 9 problems. The proposed algorithm also had better standard deviations for four functions (g01, g02, g08, and g11) than the ISR algorithm. Notably, the RPGA found feasible solutions during each run of all the test functions, while the other algorithms had difficulties solving problem g10. This finding implies the proposed algorithm is robust in solving constrained problems and can perform as good as or in some cases better than other algorithms.

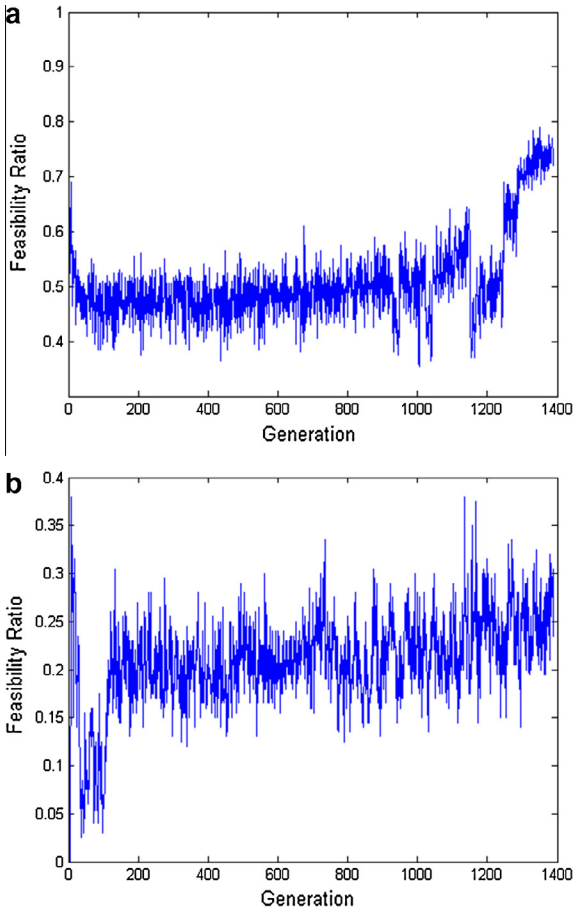

Fig. 4. The feasibility ratios for test functions (a) g02 and (b) g07 over all evolutionary generations.

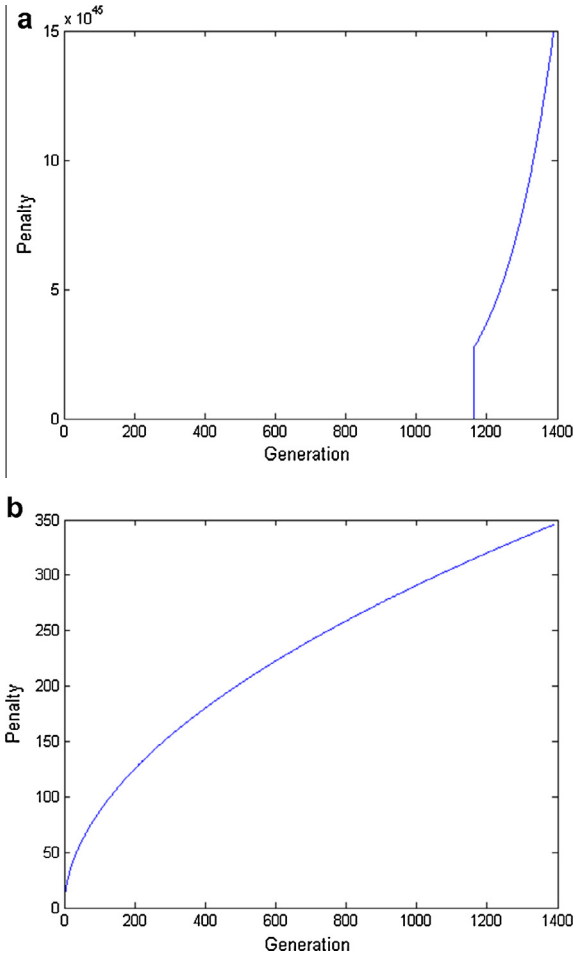

Fig. 5. The penalty coefficients for two constraints (a) g1 and (b) g2 in test function g02 over all evolutionary generations.

6.3. Performance analyses

The extent to which the proposed RP method influences the RPGA in striking a balance between the objective function and the constraint violations is of interest. To address constrained problems, a feasibility ratio (i.e., ratio of the number of feasible individuals to total population size) plays an important role in the evolutionary process and directly influences the tradeoff between the objective function and the constraint violations.

To visualize the progress of feasibility ratios, Fig. 4a and b plot the feasibility ratios for test functions g02 and g07 over all the generations of a single run, respectively. In Fig. 4a, test function g02 starts with a high feasibility ratio (0.5–0.6) because the problem has a large feasible region compared to its defined search space. Afterward, a relatively stable feasibility ratio (0.4–0.6) gradually improves to 0.8 as the number of generations increases. A moderate feasibility ratio helps the algorithm locate additional diverse solutions around the boundary of feasible space and avoid being trapped in local optima. Conversely, the feasibility ratio for problem g07 (in Fig. 4b) is 0 in the beginning due to its relatively small feasible region. As the evolution progresses, the feasibility ratio increases rapidly because the therapeutic crossover is applied. Furthermore, the feasibility ratio was bound within a small range. This phenomenon is reasonable because the RP method plays an important role in balancing between the objective function and the constraint violations.

Interestingly, the coefficients in penalty terms maintained by the RP method are consistent with active constraints. For test function g02, Fig. 5a and b plot the penalty coefficients of two constraints, $g_1(\vec{x})$ and $g_2(\vec{x})$, over different generations in a single run, respectively. The penalty of $g_1(\vec{x})$ in Fig. 5a increases exponentially from 50 to 15×10^{46}. The penalty of $g_2(\vec{x})$ in Fig. 5b increases slightly from 50 to 350 over 1400 generations. According to the principle of the RP method, constraints that are difficult to satisfy should have a relatively higher penalty coefficient. The RP method exponentially increased the penalty coefficients of active constraints during the evolution. Conversely, the penalty multipliers of feasible constraints would be assigned smaller values than the infeasible constraints to enhance genetic diversity. Because the number of active

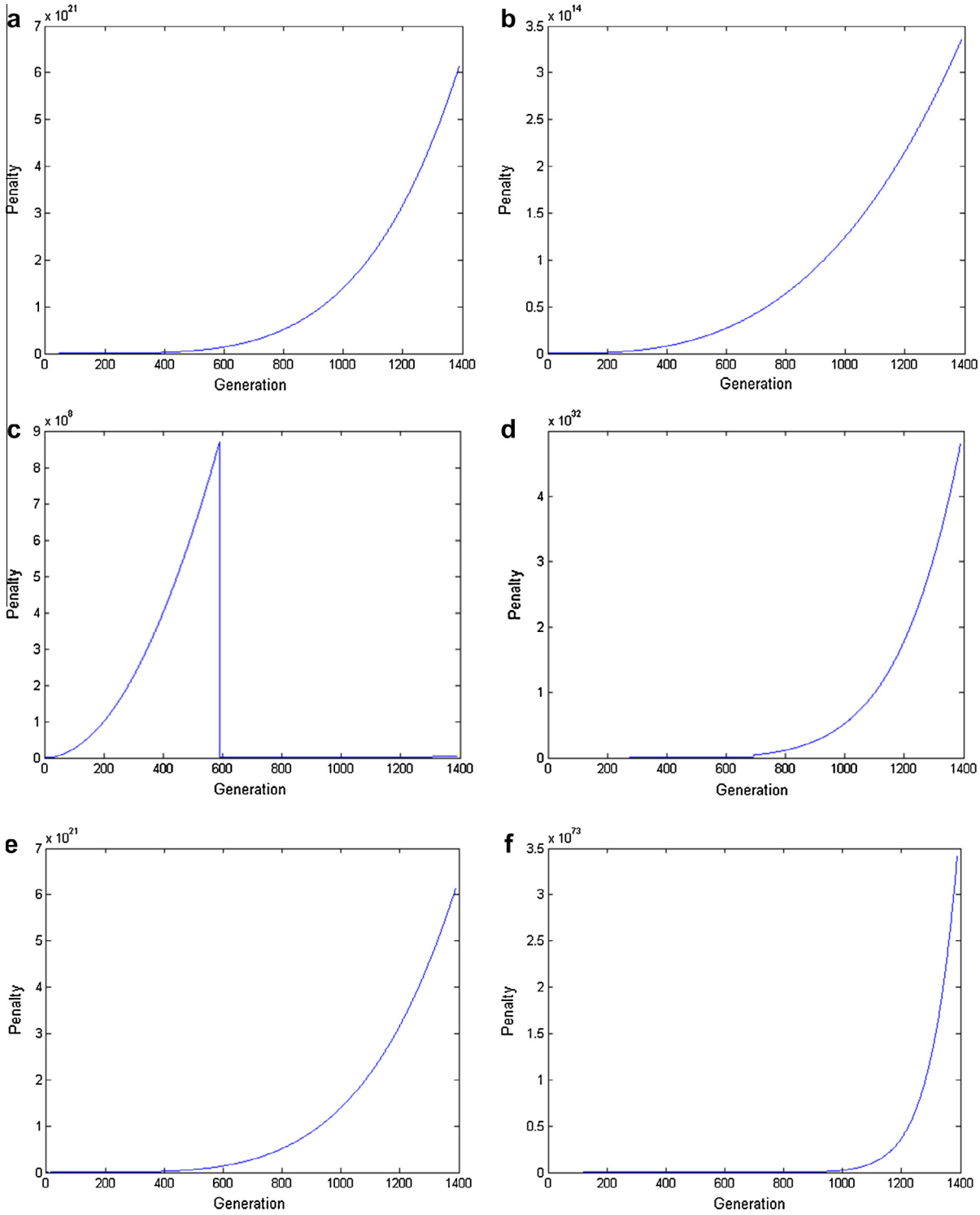

Fig. 6. The penalty coefficients for eight constraints from (a) g1 to (h) g8 in test function g07 over all evolutionary generations.

constraints for function g02 in the optimal solution is one, only a single penalty coefficient in g02 increases dramatically. That is, this RP method can automatically adjust penalty functions to achieve global optimal solutions effectively.

Fig. 6a–h show the penalty coefficients for eight constraints from $g_1(\vec{x})$ to $g_8(\vec{x})$ in test function g07 over all evolutionary generations in an example run, respectively. For the optimal solution of function g07, the number of active constraints is 6 of

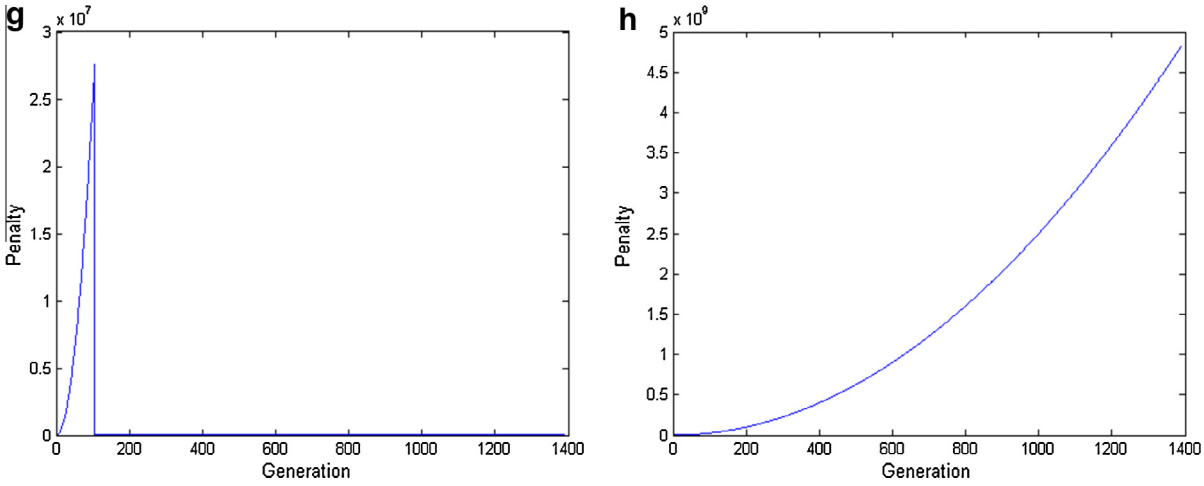

Fig. 6. (*continued*)

the 8 constraints, which are constraints $g_1(\vec{x}), g_2(\vec{x}), g_4(\vec{x}), g_5(\vec{x}), g_6(\vec{x})$, and $g_8(\vec{x})$. Because the feasibility ratio is still bounded within a small range of 25%, most infeasible individuals around the optimal solution may violate these six constraints. Thus, penalty coefficients of the 6 active constraints increase exponentially up to 10^{73}. The other two penalty coefficients consistently remain at a relatively lower value and do not increase iteratively. The low feasibility ratio of test function g07 also causes the RP method to increase the penalty coefficient more than for test function g02. Therefore, performance analyses indicate the proposed RP method is highly capable for adapting penalty coefficients to evolution situations for handling various constrained optimization problems, even though they consist of both linear and nonlinear objective functions with equality and inequality constraints.

7. Conclusions

To the best of our knowledge, the proposed RPGA is the first RST-based constraint-handling approach, which applies the information granulation of RST to address the indiscernibility relation among penalty coefficients for GAs. The proposed RP method cooperates with GAs for dealing with constrained optimization problems. The principle of the RP method is that the RPGA releases/enforces some penalties on inefficient/efficient constraints during evolution. Importantly, this approach can find the optimum or near-optimal solutions even though the initial population includes infeasible solutions. Because of its self-adaptive penalty adjustment method, the RPGA does not need prior knowledge about the solution space of a constrained problem. Moreover, the RPGA can solve a range of constrained optimization problems that have both linear and nonlinear constraints.

The performance of the proposed algorithm was measured using 11 benchmark functions. The performance was also compared with 11 state-of-the-art algorithms. Experimental results indicate the proposed RPGA finds most near-optimal solutions and outperforms four existing penalty-function algorithms. The proposed algorithm is also robust in obtaining consistent results and performed as well as, or in some cases better than, four search-bias algorithms and three multi-objective optimization algorithms. Notably, the proposed algorithm can find feasible solutions in all runs of all the test functions even though some difficult problems have smaller feasible ratios.

In conclusion, empirical analyses show the proposed RP method can address a variety of constrained optimization problems that have both linear and nonlinear objective functions with equality and inequality constraints. A performance assessment also demonstrates the proposed RP method has a remarkable capability to balance the objective function and the constraint violations for handling constrained optimization problems.

Acknowledgments

The author would like to express my deep gratitude to Professor Witold Pedrycz and the anonymous reviewers for their valuable and constructive suggestions on this manuscript.

Appendix A. Numerical test functions

(g01) The problem (Floudas and Pardalos, 1987) is

$$\begin{aligned}
\text{minimize}\quad & g01(\vec{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i,\\
\text{subject to}\quad & 2x_1 + 2x_2 + x_{10} + x_{11} \leqslant 10, \quad 2x_1 + 2x_3 + x_{10} + x_{12} \leqslant 10, \quad 2x_2 + 2x_3 + x_{11} + x_{12} \leqslant 10,\\
& -8x_1 + x_{10} \leqslant 0, \quad -8x_2 + x_{11} \leqslant 0, \quad -8x_3 + x_{12} \leqslant 0,\\
& -2x_4 - x_5 + x_{10} \leqslant 0, \quad -2x_6 - x_7 + x_{11} \leqslant 0, \quad -2x_8 - x_9 + x_{12} \leqslant 0,
\end{aligned}$$

And bounds $0 \leqslant x_i \leqslant 1, i = 1, \ldots, 9, 0 \leqslant x_i \leqslant 100, i = 10, 11, 12, 0 \leqslant x_{13} \leqslant 1$.

(g02) The problem (Keane, 1994) is

$$\begin{aligned}
\text{minimize}\quad & g02(\vec{x}) = \left| \left(\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i) \right) \Big/ \sqrt{\sum_{i=1}^{n} i x_i^2} \right|,\\
\text{subject to}\quad & \prod_{i=1}^{n} x_i \geqslant 0.75, \quad \sum_{i=1}^{n} x_i \leqslant 7.5n,\\
& \textit{And bounds } 0 \leqslant x_i \leqslant 10, \textit{ for } 1 \leqslant i \leqslant n.
\end{aligned}$$

(g03) The problem (Michalewicz et al., 1996; Schoenauer and Michalewicz, 1996) is

$$\begin{aligned}
\text{minimize}\quad & g03(\vec{x}) = \left(\sqrt{n}\right)^n \cdot \prod_{i=1}^{n} x_i,\\
\text{subject to}\quad & \sum_{i=1}^{n} x_i^2 = 1 \quad \text{and } 0 \leqslant x_i \leqslant 1, \text{ for } 1 \leqslant i \leqslant n.
\end{aligned}$$

(g04) The problem (Himmelblau, 1992) is

$$\begin{aligned}
\text{minimize}\quad & g04(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141,\\
\text{subject to}\quad & 0 \leqslant 80.51249 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leqslant 92,\\
& 90 \leqslant 80.51429 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leqslant 110,\\
& 20 \leqslant 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leqslant 25,\\
& \textit{And bounds } 78 \leqslant x_1 \leqslant 102, \quad 33 \leqslant x_2 \leqslant 45, \quad 27 \leqslant x_i \leqslant 45, \quad i = 3, 4, 5.
\end{aligned}$$

(g05) The problem (Hock and Schittkowski, 1981) is

$$\begin{aligned}
\text{minimize}\quad & g05(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + 0.000002/3x_2^3,\\
\text{subject to}\quad & x_4 - x_3 + 0.55 \geqslant 0, \quad x_3 - x_4 + 0.55 \geqslant 0,\\
& 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0,\\
& 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x2 = 0,\\
& 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0,\\
& \textit{And bounds } 0 \leqslant x_i \leqslant 1200, \quad i = 1, 2, \quad \text{and } -0.55 \leqslant x_i \leqslant 0.55, \quad i = 3, 4.
\end{aligned}$$

(g06) The problem (Floudas and Pardalos, 1987) is

$$\begin{aligned}
\text{minimize}\quad & g06(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3,\\
\text{subject to}\quad & (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geqslant 0, \quad -(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geqslant 0,\\
& \textit{And bounds } 13 \leqslant x_1 \leqslant 100 \quad \text{and } 0 \leqslant x_2 \leqslant 100.
\end{aligned}$$

(g07) The problem (Michalewicz, 1995) is

$$\begin{aligned}
\text{minimize}\quad & g07(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2\\
& + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,\\
\text{subject to}\quad & 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geqslant 0, \quad -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geqslant 0,\\
& -10x_1 + 8x_2 + 17x_7 - 2x_8 \geqslant 0, -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geqslant 0,\\
& 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geqslant 0, -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geqslant 0,\\
& 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geqslant 0, -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geqslant 0,\\
& \textit{And bounds } -10 \leqslant x_i \leqslant 10, \quad i = 1, \ldots, 10.
\end{aligned}$$

(g08) The problem (Schoenauer and Xanthakis, 1993) is

$$\begin{aligned}
\text{minimize} \quad & g08(\vec{x}) = \frac{\sin^3(2\pi x_1)\cdot\sin(2\pi x_2)}{x_1^3\cdot(x_1+x_2)},\\
\text{subject to} \quad & x_1^2 + x_2 + 1 \leqslant 0, \quad 1 - x_1 + (x_2-4)^2 \leqslant 0,\\
& \textit{And bounds } 0 \leqslant x_1 \leqslant 10 \quad \text{and } 0 \leqslant x_2 \leqslant 10.
\end{aligned}$$

(g09) The problem (Hock and Schittkowski, 1981) is

$$\begin{aligned}
\text{minimize} \quad & g09(\vec{x}) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6\\
& + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7,\\
\text{subject to} \quad & 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geqslant 0, \quad 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geqslant 0,\\
& 196 - 23x_1 - x_2^2 - 6x_6^2 + x_7 \geqslant 0, \quad -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geqslant 0,\\
& \textit{And bounds } -10.0 \leqslant x_i \leqslant 10.0, \quad i = 1\ldots,7.
\end{aligned}$$

(g10) The problem (Hock and Schittkowski, 1981) is

$$\begin{aligned}
\text{minimize} \quad & g10(\vec{x}) = x_1 + x_2 + x_3,\\
\text{subject to} \quad & 1 - 0.0025(x_4 + x_6) \geqslant 0, \quad 1 - 0.0025(x_5 + x_7 - x_4) \geqslant 0,\\
& 1 - 0.01(x_8 - x_5) \geqslant 0, \quad x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 \geqslant 0,\\
& x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geqslant 0, \quad x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \geqslant 0,\\
& \textit{And bounds } 100 \leqslant x_1 \leqslant 10000, \quad 1000 \leqslant x_i \leqslant 10,000, \quad i = 2,3, \quad 10 \leqslant x_i \leqslant 1000, \quad i = 4,\ldots,8.
\end{aligned}$$

(g11) The problem (Michalewicz and Schoenauer, 1996) is

$$\begin{aligned}
\text{minimize} \quad & g11(\vec{x}) = x_1^2 + (x_2-1)^2,\\
\text{subject to} \quad & x_2 - x_1^2 = 0,\\
& \textit{And bounds } -1 \leqslant x_i \leqslant 1, \quad i = 1,2.
\end{aligned}$$

References

[1] H.J.C. Barbosa, A.C.C. Lemonge, A new adaptive penalty scheme for genetic algorithms, Information Sciences 156 (2003) 215–251.
[2] N. Belavendram, Quality by Design, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[3] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer Methods in Applied Mechanics and Engineering 191 (11–12) (2002) 1245–1287.
[4] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2000) 311–338.
[5] E.Z. Elfeky, R.A. Sarker, D.L. Essam, Analyzing the simple ranking and selection process for constrained evolutionary optimization, Journal of Computer Science and Technology 23 (1) (2008) 19–34.
[6] R. Farmani, J.A. Wright, Self-adaptive fitness formulation for constrained optimization, IEEE Transactions on Evolutionary Computation 7 (5) (2003) 445–455.
[7] M. Gen, R. Cheng, A survey of penalty techniques in genetic algorithms, Evolutionary Computation (1996) 804–809.
[8] A.B. Hadj-Alouane, J.C. Bean, A genetic algorithm for the multiple-choice integer program, Operations Research 45 (1) (1997) 92–101.
[9] A.G. Hernández-Díaz, L.V. Santana-Quintero, C.A.C. Coello, R. Caballero, J. Molina, A new proposal for multi-objective optimization using differential evolution and rough sets theory, in: Proc. Genetic and Evolutionary Computation Conference, 2006, pp. 675–682.
[10] R. Hinterding, Gaussian mutation and self-adaptation for numeric genetic algorithms, in: Proc. IEEE Int. Conf. Evolutionary Computation, 1997, pp. 87–91.
[11] P.Y. Ho, K. Shimizu, Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme, Information Sciences 177 (14) (2007) 2985–3004.
[12] W. Hock, K. Schittkowski, Test examples for nonlinear programming codes, Lecture Notes in Economics and Mathematical Systems, vol. 187, Springer-Verlag, New York, 1981.
[13] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, 1975.
[14] A. Homaifar, C. Qi, S. Lai, Constrained optimization via genetic algorithms, Simulation 62 (4) (1994) 242–254.
[15] J.A. Joines, C.R. Houck, On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GA's, in: Proc. 1st IEEE Conf. Evolutionary Computation, 1994, pp. 579–584.
[16] S.O. Kimbrough, G.J. Koehler, M. Lu, D.H. Wood, On a feasible–infeasible two-population (FI-2Pop) genetic algorithm for constrained optimization: distance tracing and no free lunch, European Journal of Operational Research 190 (2008) 310–327.
[17] J. Komorowski, L. Polkowski, A. Skowron, Rough sets: a tutorial, in: S.K. Pal, A. Skowron (Eds.), Rough Fuzzy Hybridization: A New Trend in Decision-Making, Springer-Verlag, Singapore, 1999, pp. 3–98.
[18] S. Koziel, Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evolutionary Computation 7 (1) (1999) 19–44.
[19] A.C.C. Lemonge, H.J.C. Barbosa, An adaptive penalty scheme for genetic algorithms in structural optimization, International Journal for Numerical Methods in Engineering 59 (5) (2004) 703–736.
[20] C-H. Lin, C-C. Chuang, A rough set penalty function for marriage selection in multiple-evaluation genetic algorithms, Lecture Notes in Computer Science 4481 (2007) 500–507.
[21] C-H. Lin, J-D. He, A multiple-evaluation genetic algorithm for numerical optimization problems, in: Proc. Computability in Europe: Computation and Logic in the Real World, 2007, pp. 239–246.
[22] Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, in: Proc. 4th Annual Conf. Evolutionary Programming, 1995, pp. 135–155.
[23] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation 4 (1996) 1–32.

[24] Z. Pawlak, Rough sets, International Journal of Computer and Information Sciences 11 (1982) 341–356.
[25] Z. Pawlak, A. Skowron, Rudiments of rough sets, Information Sciences 177 (2007) 3–27.
[26] Z. Pawlak, A. Skowron, Rough sets: some extensions, Information Sciences 177 (2007) 28–40.
[27] Z. Pawlak, A. Skowron, Rough sets and Boolean reasoning, Information Sciences 177 (2007) 41–73.
[28] D. Powell, M.M. Skolnick, Using genetic algorithms in engineering design optimization with non-linear constraints, in: Proc. 5th Int. Conf. Genetic Algorithms, 1993, pp. 424–431.
[29] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 4 (2000) 344–354.
[30] T.P. Runarsson, X. Yao, Search biases in constrained evolutionary optimization, IEEE Transactions on Systems Man Cybernetics—Part C: Applications and Reviews 35 (2) (2005) 233–243.
[31] L.V. Santana-Quintero, A.G. Hernández-Díaz, J. Molina, C.A.C. Coello, R. Caballero, DEMORS: a hybrid multi-objective optimization algorithm using differential evolution and rough set theory for constrained problems, Computers & Operations Research (2009), http://dx.doi.org/10.1016/j.cor.2009.02.006.
[32] M. Schoenauer, S. Xanthakis, Constrained GA optimization, in: Proc. 5th Int. Conf. Genetic Algorithms, 1993, pp. 573–580.
[33] G. Taguchi, S. Chowdhury, S. Taguchi, Robust Engineering, McGraw-Hill, New York, 2000.
[34] B. Tessema, G.G. Yen, A self-adaptive penalty function based algorithm for constrained optimization, in: Proc. IEEE Congress on Evolutionary Computation, 2006, pp. 950–957.
[35] J.T. Tsai, T.K. Liu, J.H. Chou, Hybrid Taguchi-genetic algorithm for global numerical optimization, IEEE Transactions on Evolutionary Computation 8 (4) (2004) 365–377.
[36] S. Venkatraman, G.G. Yen, A generic framework for constrained optimization using genetic algorithms, IEEE Transactions on Evolutionary Computation 9 (4) (2005) 424–435.
[37] Y. Wang, Z. Cai, G. Guo, Y. Zhou, Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 37 (3) (2007) 560–575.
[38] Y. Wang, Z. Cai, Y. Zhou, Z. Fan, Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique, Structural and Multidisciplinary Optimization 37 (4) (2009) 395–413.
[39] Y. Wang, Z. Cai, Y. Zhou, W. Zeng, An adaptive tradeoff model for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 80–92.
[40] Y. Wu, Taguchi Methods for Robust Design, ASME, New York, 2000.
[41] M. Yuchi, J.H. Kim, Grouping-based evolutionary algorithm: seeking balance between feasible and infeasible individuals of constrained optimization problems, in: Proc. Congress on Evolutionary Computation, 2004, pp. 280–287.
[42] M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences 178 (15) (2008) 3043–3074.