JOTA manuscript No.
(will be inserted by the editor)

Theoretical and Practical Convergence of a Self-adaptive Penalty Algorithm for Constrained Global Optimization

M. Fernanda P. Costa · Rogério B. Francisco · Ana Maria A.C. Rocha · Edite M.G.P. Fernandes

Received: date / Accepted: date

Abstract This paper proposes a self-adaptive penalty function and presents a penalty-based algorithm for solving nonsmooth and nonconvex constrained

Communicated by Dario Izzo

M. Fernanda P. Costa, Corresponding author

Centre of Mathematics, University of Minho, 4710-057 Braga, Portugal

E-mail: mfc@math.uminho.pt

Rogério B. Francisco

Center for Research and Innovation in Business Sciences and Information Systems, Polytechnic of Porto, Porto, Portugal

E-mail: rbf@estgf.ipp.pt

Ana Maria A.C. Rocha

Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal

E-mail: arocha@dps.uminho.pt

Edite M.G.P. Fernandes

Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal

E-mail: emgpf@dps.uminho.pt

optimization problems. We prove that the general constrained optimization problem is equivalent to a bound constrained problem in the sense that they have the same global solutions. The global minimizer of the penalty function subject to a set of bound constraints may be obtained by a population-based meta-heuristic. Further, a hybrid self-adaptive penalty firefly algorithm, with a local intensification search, is designed and its convergence analysis is established. The numerical experiments and a comparison with other penalty-based approaches show the effectiveness of the new self-adaptive penalty algorithm in solving constrained global optimization problems.

Keywords Global optimization · Self-adaptive penalty · Firefly algorithm

Mathematics Subject Classification (2000) 90C30 · 90C26 · 90C59

1 Introduction

A penalty technique transforms the constrained optimization problem into a sequence of unconstrained subproblems, in a way that the sequence of solutions of the unconstrained subproblems converges to the optimal solution of the original constrained problem [1]. The technique is simple to implement and takes advantage of existing and powerful unconstrained optimization methods. However, defining a strategy to initialize and update the penalty parameter is not an easy task. To address the concerning issue related to setting the penalty parameter values within a penalty-based algorithm, a new self-adaptive penalty function is derived.

This paper illustrates the behavior of a penalty technique, that relies on a self-adaptive penalty function, to solve constrained global optimization (CGO) problems. To promote convergence to a global optimal solution, the resulting bound constrained global optimization (BCGO) problems are solved by well-known population-based meta-heuristics. Although they have been implemented with different constraint handling techniques for solving CGO problems, mainly penalty-based methods [2–6], this study shows that the proposed self-adaptive penalty technique, when combined with the meta-heuristics, is also very effective in solving CGO problems. In particular, we analyze the performance of the firefly algorithm (FA) [7] when combined with the self-adaptive penalty technique. FA is a swarm intelligence based algorithm that became very popular over the last decade. Several variants of the FA [8–11], including hybrid approaches [12, 13], and applications have been recently reported in the literature [14–16]. The effect of the control parameters on the performance of the FA has been studied in [17–19]. The main motivation for using the FA, besides being one of the most recent meta-heuristics, is related to its success when solving practical and complex problems [2, 20–25]. Although other adaptive penalty based functions have been recently combined with stochastic population-based global optimizers [3, 4, 26–28], our proposal is simpler to implement and the convergence of the algorithm is supported by the theoretical results. The authors in [3] construct a parameter-free penalty function. The therein proposed adaptive penalty gives the objective function value alone if the point is feasible, and combines the sum of constraint viola-

tion with either the objective value or an upper bound of the global minimum if the point is infeasible. They prove that the CGO and the BCGO problems, based on their adaptive penalty function, have the same global minimizers, and present further theoretical results based on the structure of the population-based differential evolution (DE) algorithm [29]. In [4], the adaptive penalty method (APM) investigated in [26] is extended and applied with the DE. The authors in [26] use information from the population, such as the average of the objective function values and the level of violation of each constraint, at each iteration, to define the penalty parameter. In [27, 28], the normalized objective function value and a sum of the normalized constraint violations are combined to define a modified fitness value. In both papers, a real coded genetic algorithm (GA) is used in the adaptive penalty algorithm. No theoretical convergence results are supplied in the last mentioned papers [4, 26–28].

Our contribution goes beyond the self-adaptive penalty function proposal. First, we prove that the CGO and the BCGO problems, based on the proposed self-adaptive penalty function, are equivalent in the sense that they have the same global minimizers. A selected set of meta-heuristics, the FA, a DE strategy with self-adaptive control parameters (jDE) [30], the particle swarm optimization (PSO) algorithm [31, 32], an evolution strategy with covariance matrix adaptation (CMA-ES) [33] and the artificial bee colony (ABC) algorithm [34] are used to solve the BCGO problem. The issue related to the adequacy of the computation of the parameters required to construct the self-adaptive penalty function in a population environment is addressed. Second,

in the context of the FA, we provide a hybrid variant by using a local intensification procedure. The convergence analysis of the algorithm, that takes into consideration the structure of the FA and the properties of the proposed self-adaptive penalty function, is established.

The paper is organized as follows. Sect. 2 presents the new self-adaptive penalty function, Sect. 3 elaborates on the computation of the penalty in a population-based environment and Sect. 4 details the new hybrid self-adaptive penalty FA. Then, the numerical experiments are shown in Sect. 5 and we conclude the paper in Sect. 6.

2 Self-adaptive Penalty Function

This study aims to propose a self-adaptive penalty framework for solving a CGO problem in the following form

$$\min_{x \in X \subset \mathbb{R}^n} f(x) \quad \text{subject to} \quad g(x) \leq 0, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^p$ are continuous possibly nonlinear functions in $X := \{x \in \mathbb{R}^n : -\infty < l_s \leq x_s \leq u_s < \infty, s = 1, \ldots, n\}$ (a compact set) and the feasible set is defined by $S := \{x \in X : g_j(x) \leq 0, j = 1, \ldots, p\}$. Let x^* be a global minimizer to the problem (1) and let $f^* = f(x^*)$ be the global minimum. The feasible set $S \subseteq X$ is assumed to be non-empty with a positive measure. Problems with equality constraints $h(x) = 0$ can be reformulated into the above form using $h(x) - \delta \leq 0$ and $-h(x) - \delta \leq 0$, where δ is a small positive tolerance. Since we do not assume that the functions

$f, g_j, j = 1, \ldots, p$ are differentiable, a derivative-free technique that does not assume convexity and differentiability is required for solving the problem (1).

The CGO problem (1) can be formulated as a BCGO problem with an objective penalty function, that is related to both f and the constraint violation. Thus, the problem (1) is equivalent to

$$\min_{x \in X \subset \mathbb{R}^n} \quad \phi(x) \tag{2}$$

in the sense that they have the same solutions, provided that the objective penalty function ϕ satisfies some properties [3].

In this study, the main goal is to derive a penalty function, that is self-adaptive, in the sense that the constraint violation weights, also considered as penalty parameter values, are not provided by the user but rather they are computed using information gathered from the violated constraints at the current point. Furthermore, the objective function and the constraint violation values are normalized taking into consideration reference values of the objective function and constraints achieved in the search space of the problem. The description of the self-adaptive penalty function follows. The objective function value f at each point x is normalized making use of the two parameters $f^{\min} := \min_{x \in X} f(x)$ and $f^{\max} := \max_{x \in X} f(x)$ in a way that the new fitness F is computed by:

$$F(x) = \frac{f(x) - f^{\min}}{f^{\max} - f^{\min}}. \tag{3}$$

The violation of each constraint j, at each point x of the search space X, is given by $\max\{g_j(x), 0\}$ and the total violation is the sum of the p violations:

$$\Sigma(x) = \sum_{j=1}^{p} \max\{g_j(x), 0\}, \tag{4}$$

which is zero if $x \in S$ (a feasible point) and positive if $x \notin S$. However, to scale the constraint violation to the same order of magnitude as the new fitness F, each constraint violation is normalized using the following expression:

$$V_j(x) = \frac{\max\{g_j(x), 0\}}{g_j^{\max}}, \quad \text{where} \quad g_j^{\max} := \max_{x \in X \setminus S} \{\max\{g_j(x), 0\}\} \tag{5}$$

is the largest value for the violation of the constraint j for all $x \in X \setminus S$, being the subset $X \setminus S$ the relative complement of S in X. Finally, the penalty function to be minimized is as follows:

$$\phi(x) = \begin{cases} F(x), & \text{if } x \in S, \\ F(z) + \dfrac{1}{p} \displaystyle\sum_{j=1}^{p} V_j(x) r_j, & \text{if } x \in X \setminus S \text{ and } f(x) \leq f(z), \\ F(x) + \dfrac{1}{p} \displaystyle\sum_{j=1}^{p} V_j(x) r_j, & \text{if } x \in X \setminus S \text{ and } f(x) > f(z), \end{cases} \tag{6}$$

where $z \in S$ is a fixed point such that $f(z) \geq f^*$, and each weight r_j is defined by the proportion of the search space X that violates the constraint g_j:

$$r_j := \frac{|x \in X : g_j(x) > 0|}{|X|}, \ j = 1, \ldots, p. \tag{7}$$

The next results show that problems (1) and (2) are equivalent, i.e., they have the same global minimizers.

Theorem 2.1 *Let $x^* \in S$ be a global solution to the problem* (1) *and let $z \in S$ be such that $f(z) \geq f(x^*)$. Then, x^* is a global solution to the problem* (2)*, where ϕ is the penalty function defined in* (6)*.*

Proof Let $x^* \in S$ be a global solution to the problem (1). By definition, we have $f(x^*) \leq f(x)$ for all $x \in S$. Hence, for all $x \in S$ we get:

$$\phi(x^*) = \frac{f(x^*) - f^{\min}}{f^{\max} - f^{\min}} \leq \frac{f(x) - f^{\min}}{f^{\max} - f^{\min}} = \phi(x).$$

We now consider the case when $x \in X \setminus S$. Assuming that (a) $f(x) \leq f(z)$, we have $\phi(x^*) = F(x^*) \leq F(z) < F(z) + \frac{1}{p}\sum_{j=1}^{p} V_j(x) r_j = \phi(x)$, since V_j and r_j are positive, $f(x^*) \leq f(z)$ and using the definition (6). Now, assuming that (b) $f(x) > f(z)$, we get

$$\phi(x^*) = F(x^*) \leq F(z) < F(x) < F(x) + \frac{1}{p}\sum_{j=1}^{p} V_j(x) r_j = \phi(x),$$

and therefore $\phi(x^*) \leq \phi(x)$ for all $x \in X$, i.e., x^* is a global solution to the problem (2). □

Lemma 2.1 *If x^* is a global solution to the problem* (2)*, where ϕ is the penalty function defined in* (6)*, then x^* is a feasible point for the problem* (1)*.*

Proof By contradiction, we assume that $x^* \in X \setminus S$. When $f(x^*) \leq f(z)$ and $z \in S$ we get, from (6), $\phi(x^*) = F(z) + \frac{1}{p}\sum_{j=1}^{p} V_j(x^*) r_j > F(z) = \phi(z)$; on the other hand, when $f(x^*) > f(z)$ we obtain the relation (using (6)) $\phi(x^*) = F(x^*) + \frac{1}{p}\sum_{j=1}^{p} V_j(x^*) r_j > F(x^*) > F(z) = \phi(z)$, which contradict the definition of a global solution to the problem (2). Therefore, $x^* \in S$. □

We are now able to establish the reciprocal of Theorem 2.1.

Theorem 2.2 *Let $x^* \in X$ be a global solution to the problem* (2)*, where ϕ is the penalty function defined by* (6)*. Then, x^* is a global solution to the problem* (1)*.*

Proof By Lemma 2.1 $x^* \in S \subset X$. We have $F(x^*) = \phi(x^*) \leq \phi(x)$ for all $x \in X$, and in particular for all $x \in S$ we have $F(x^*) \leq F(x)$, which implies $f(x^*) \leq f(x)$. Therefore x^* is a global solution to the problem (1). □

3 Solving the BCGO Problem

The present penalty method aims to penalize the inequality constraints violation of the problem (1) while the bound constraints are always satisfied when solving (2). According to the Theorems 2.1 and 2.2 it is sufficient to find a global solution to the problem (2), that is, a global minimizer of $\phi(x)$ in X. To solve the BCGO problem, the meta-heuristics FA [7,9,15], jDE [30], PSO [31,32], CMA-ES [33] and ABC [34] have been selected. Since they are population-based algorithms, we now show how to adequate the computation of parameters $f^{\min}$, $f^{\max}$, $f(z)$, $g_j^{\max}$ and r_j, $j = 1, \ldots, p$, shown in (3), (5), (6) and (7), to a technique that handles a population of solutions at each iteration.

Let $X_k := \{x_k^1, \ldots, x_k^m\}$ represent the population of the $m < +\infty$ current points at iteration k, where $x_k^i \in \mathbb{R}^n, i = 1, \ldots, m$. To compute the normalized fitness F, as defined in (3), at each point x of the population, the parameters $f^{\min} := \min_{x \in X_k} f(x)$ and $f^{\max} := \max_{x \in X_k} f(x)$ are required, where we note that the point with the lowest function value will have $F(x) = 0$ and the point with largest objective function value will have $F(x) = 1$. To compute the normalized violation of the constraint j, the parameter $g_j^{\max} := \max_{x \in X_k} \{\max\{g_j(x), 0\}\}$ is defined as the largest value for the violation of the constraint j attained at

all points in X_k. The reference point z is the feasible point with the lowest objective function value found so far. If the population has no feasible points, $f(z)$ is initially and temporarily set to $f^{\max}$, so that $f(x) \leq f(z)$ for all $x \in X_k$ and $F(z) = 1$. The value of $f(z)$ is updated only when the first feasible point is encountered. Noting that, at each iteration k, the set of m generated trial points is represented by $T_k := \{t_k^1, \ldots, t_k^m\}$, if the generated T_k contains feasible points, the one with least function value, say $f(t_k^l)$, is compared with $f(z)$ and we set $f(z) = f(t_k^l)$ if $f(t_k^l) < f(z)$; otherwise $f(z)$ is not updated. Similarly, $f(z)$ is maintained to the next iteration if there is no feasible points in the trial population. Finally each weight/penalty parameter r_j is iteratively computed as $r_j := (|x \in X_k : g_j(x) > 0|)/m$ $(j = 1, \ldots, p)$ and represents the proportion of points in the population that violate the constraint g_j. Thus, a constraint that is violated by a larger set of points of the population than any other will have a larger weight.

4 Hybrid Self-adaptive Penalty FA for CGO

This section details the algorithm, that implements the self-adaptive penalty concept, while using the meta-heuristic FA to compute the solution of the BCGO problem (2) (see Algorithm 1). This is a hybrid FA in the sense that a local intensification procedure based on a typical DE mutation operator [29] is implemented aiming to exploit the region around the points of the population. The intensification procedure starts by applying a mutation strategy to the position of the best firefly, x^1, where $\phi(x^1) < \phi(x^i), i = 2, \ldots, m$, componentwise

with probability p_m, to create the mutant best point, $v^1 = x^1 + F_b\left(x^{i_1} - x^{i_2}\right)$, where i_1 and i_2 are two different indices randomly selected from the set $\{2, \ldots, m\}$ and $F_b > 0$ is a real parameter. A projection onto X is carried out if necessary, v^1 and x^1 are compared and the preferred point is selected as new x^1. Here, the preferred point is the one that has the smallest f value if both are feasible; otherwise is the point that has the smallest violation. The DE/best/1 mutation is then applied to the remaining points of the population, $v^i = x^1 + F_o\left(x^{i_1} - x^{i_2}\right)$, $i = 2, \ldots, m$, componentwise with probability p_m, where $F_o > 0$ is a real parameter, and i_1 and i_2 are two different indices randomly chosen from the set $\{1, \ldots, i-1, i+1, \ldots, m\}$. The mutant v^i and x^i are compared and the preferred point is maintained to the next iteration.

For the convergence analysis of the Algorithm 1, we follow the methodology presented in [3]. Attending to the properties of the FA, and the way the penalty function ϕ is defined we can establish the following results.

Theorem 4.1 *Let X_k be the current population of m points at iteration k, T_k be the set of trial points at iteration k, and X_{k+1} be the population with the points selected for the next iteration $k+1$. Then $f(z_k) \geq f(z_{k+1})$, where z_k is the feasible point with the lowest function value in the set X_k and z_{k+1} is the feasible point with the lowest function value found in T_k. Furthermore, $\phi(z_k) \leq \phi(t_k^i)$, for all infeasible $t_k^i \in T_k$.*

Proof Let z_k be the best feasible solution of X_k. Obviously z_k will never be replaced by any infeasible point of T_k. We assume now that there exists a feasible point $t_k^i \in T_k$ such that $\phi(t_k^i) < \phi(z_k)$. Then,

Algorithm 1: Hybrid self-adaptive penalty FA

Data: $k_{\max}$, ϵ, η, m, f^*

Set $k = 1$. Randomly generate $x_k^i \in X$, $i = 1, \ldots, m$, evaluate ϕ and rank fireflies (from lowest to largest ϕ);

while *(*$|f^* - f(x_k^1)| > \epsilon$ *or* $\Sigma(x_k^1) > \eta$*) and* $k \leq k_{\max}$ **do**

  forall the x_k^i *such that* $i = 2, \ldots, m$ **do**

    forall the x_k^j *such that* $j = 1, \ldots, i-1$ **do**

      Move firefly i towards firefly j;

    Set t_k^i = Project x_k^i onto X;

  Based on $X_k \cup T_k$ evaluate ϕ;

  forall the $i = 1, \ldots, m$ **do**

    if $\phi(t_k^i) < \phi(x_k^i)$ **then**

      Set $x_{k+1}^i = t_k^i$;

    else

      Set $x_{k+1}^i = x_k^i$;

  Based on X_{k+1} evaluate ϕ and rank fireflies;

  Invoke the local intensification procedure, evaluate ϕ and rank fireflies;

  Set $k = k + 1$;

$\phi(t_k^i) = F(t_k^i) < F(z_k) = \phi(z_k)$ implies $f(t_k^i) < f(z_k)$, where $f^{\min}$ and $f^{\max}$ (for the definition of fitness F) are selected from the set $X_k \cup T_k$. We conclude that $f(z_k) > f(t_k^i) \geq f(z_{k+1})$. However, if the feasible point $t_k^i \in T_k$ does not satisfy $\phi(t_k^i) < \phi(z_k)$, then $\phi(t_k^i) = F(t_k^i) \geq F(z_k) = \phi(z_k)$ which implies $f(t_k^i) \geq f(z_k)$ and $f(z_{k+1}) = f(z_k)$. In both cases $f(z_k) \geq f(z_{k+1})$. We consider now the case where $t_k^i \in T_k$ is infeasible. We analyze both cases: (a) $f(t_k^i) \leq f(z_k)$ and (b) $f(t_k^i) > f(z_k)$. In case (a), assume that $\phi(t_k^i) < \phi(z_k)$

which implies

$$F(z_k) + \frac{1}{p}\sum_{j=1}^{p} V_j(t_k^i) r_j < F(z_k) \tag{8}$$

since t_k^i is infeasible and $f(t_k^i) \leq f(z_k)$ (see (6)). However, the last condition in (8) is a contradiction because the second term on the left hand side of the equation is positive. When in case (b) we assume that $\phi(t_k^i) < \phi(z_k)$, we get $F(t_k^i) + \frac{1}{p}\sum_{j=1}^{p} V_j(t_k^i) r_j < F(z_k)$ and therefore

$$\frac{1}{p}\sum_{j=1}^{p} V_j(t_k^i) r_j < F(z_k) - F(t_k^i) = \frac{f(z_k) - f(t_k^i)}{f^{\max} - f^{\min}} < 0$$

which is a contradiction. Hence, we must have $\phi(z_k) \leq \phi(t_k^i)$ for all infeasible points $t_k^i \in T_k$. □

In the next theorem, we prove that the sequence $\{f(z_k)\}$ converges and the limit is the greatest lower bound, or infimum, f^*.

Theorem 4.2 *Let z_k be the feasible point with the lowest objective function value obtained at iteration k. Then, $\lim_{k\to\infty} f(z_k) = f^*$.*

Proof By Theorem 4.1, $\{f(z_k)\}$ is a monotonically decreasing sequence. Since f^* is the infimum of the sequence, then for all $\delta > 0$, $f^* + \delta$ is not an infimum of the sequence. Hence, there exists $K = K(\delta) \in \mathbb{N}$, such that

$$f^* - \delta < f^* \leq f(z_k) \leq f(z_K) < f^* + \delta$$

for all $k \geq K$, meaning that $f(z_k) \to f^*$ as $k \to \infty$. □

In the Algorithm 1, to select between the current and the trial positions, both penalty function values $\phi(x_k^i)$ and $\phi(t_k^i)$ are compared. When both x_k^i and t_k^i are feasible, the point with the lowest f wins (recall (6) and that parameters

$f^{\min}$ and $f^{\max}$ are computed based on the set $X_k \cup T_k$). On the other hand, when x_k^i and t_k^i are infeasible, the selection is determined by their constraint violation and F values combined in the penalty ϕ. However, when x_k^i is feasible and t_k^i is infeasible, the probability that the trial t_k^i is selected over x_k^i as the current point for the next iteration $k+1$ could be determined.

Theorem 4.3 *Let $x_k^i \in X_k$, where X_k is the current population at iteration k, and $t_k^i \in T_k$, where T_k is the set of trial points at iteration k, be such that x_k^i is feasible and t_k^i is infeasible. Assume that there exists $0 < \bar{r} \leq 1$ such that eventually $r_j \geq \bar{r}$ for $j = 1, \ldots, p$. Then, the probability of selecting t_k^i over x_k^i is zero, i.e., $Pr\left[\frac{1}{p}\sum_{j=1}^{p} V_j(t_k^i) r_j < F(x_k^i) - F(z_k)\right] = 0$ for $r_j, j = 1, \ldots, p$, that satisfy $r_j \geq \bar{r}$.*

Proof Assume that $t_k^i \in T_k$ is almost always selected when compared with a feasible $x_k^i \in X_k$, i.e., $\phi(t_k^i) < \phi(x_k^i)$. Hence, (a) if $f(t_k^i) \leq f(z_k)$, we have $\phi(t_k^i) = F(z_k) + \Sigma^n(t_k^i) < \phi(x_k^i) = F(x_k^i)$ which implies

$$0 < \Sigma^n(t_k^i) < F(x_k^i) - F(z_k) \leq 1, \tag{9}$$

where for simplicity $\Sigma^n(t_k^i) = \frac{1}{p}\sum_{j=1}^{p} V_j(t_k^i) r_j > 0$. On the other hand, (b) if $f(t_k^i) > f(z_k)$, we get $\phi(t_k^i) = F(t_k^i) + \Sigma^n(t_k^i) < \phi(x_k^i) = F(x_k^i)$ yielding

$$0 < \Sigma^n(t_k^i) < F(x_k^i) - F(t_k^i) < F(x_k^i) - F(z_k) \leq 1. \tag{10}$$

We note that in both (9) and (10), $f(x_k^i) - f(z_k) > 0$, provided that $x_k^i \neq z_k$. We now study the probability of $\Sigma^n(t_k^i) < F(x_k^i) - F(z_k)$ being held. We assume that the trial point T_k^i is a random variable with realizations t_k^i and that

$\Sigma^n(T_k^i)$ increases uniformly away from the feasibility. Since $F(x_k^i) - F(z_k)$ is a fixed number in the range $(0, 1]$, $Pr\left[\Sigma^n(T_k^i) < F(x_k^i) - F(z_k)\right] > 0$ holds for (9) and (10). The larger $F(x_k^i) - F(z_k)$, the larger the probability is. However, this probability also depends on $\Sigma^n(T_k^i)$. By contradiction, we assume that there exists $0 < \bar{r} \leq 1$ such that $Pr\left[\frac{1}{p}\sum_{j=1}^{p} V_j(T_k^i) r_j < F(x_k^i) - F(z_k)\right] > 0$, when r_j, $j = 1, \ldots, p$ satisfy $r_j \geq \bar{r}$. This means that

$$\frac{1}{p}\sum_{j=1}^{p} V_j(t_k^i) r_j < F(x_k^i) - F(z_k) \text{ for } r_j \geq \bar{r}, j = 1, \ldots, p. \tag{11}$$

However, there certainly exists a value $T_k^i = t_k^i$, such that for r_j, $j = 1, \ldots, p$ satisfying $r_j \geq \bar{r}$, $\frac{1}{p}\sum_{j=1}^{p} V_j(t_k^i) r_j \geq F(x_k^i) - F(z_k)$, which contradicts (11). □

We now consider the situation when x_k^i is infeasible and the trial t_k^i is feasible and analyze the probability that the current x_k^i is selected over t_k^i as the current point for the next iteration $k + 1$.

Theorem 4.4 *Let $x_k^i \in X_k$, where X_k is the current population of m points at iteration k, and $t_k^i \in T_k$, where T_k is the set of trial points at iteration k, be such that x_k^i is infeasible and t_k^i is feasible. Then, there exists $0 < \bar{r} \leq 1$ such that the probability of selecting x_k^i over t_k^i is zero, i.e., $Pr\left[\frac{1}{p}\sum_{j=1}^{p} V_j(x_k^i) r_j < F(t_k^i) - F(z_k)\right] = 0$ when $r_j, j = 1, \ldots, p$ satisfy $r_j \geq \bar{r}$.*

Proof Assume that $x_k^i \in X_k$ is almost always selected when compared with a feasible $t_k^i \in T_k$, which means that $\phi(x_k^i) < \phi(t_k^i)$. When (a) $f(x_k^i) \leq f(z_k)$, $\phi(x_k^i) = F(z_k) + \Sigma^n(x_k^i) < \phi(t_k^i) = F(t_k^i)$ and $\Sigma^n(x_k^i) < F(t_k^i) - F(z_k)$ is obtained. When (b) $f(x_k^i) > f(z_k)$, $\phi(x_k^i) = F(x_k^i) + \Sigma^n(x_k^i) < \phi(t_k^i) = F(t_k^i)$ implies $\Sigma^n(x_k^i) < F(t_k^i) - F(x_k^i)$ or $\Sigma^n(x_k^i) < F(t_k^i) - F(z_k)$.

Assuming that the trial point T_k^i and $f(T_k^i)$ are random variables with realizations t_k^i and $f(t_k^i)$ respectively, we have that $F(T_k^i) - F(z_k)$ is bounded (since t_k^i is feasible, $f(t_k^i)$ is bounded and $f(z)$ is fixed). Thus, there exists a set of values $T_k^i = t_k^i$ such that $\Sigma^n(x_k^i) < F(t_k^i) - F(z_k)$ holds, which means that $Pr\left[\Sigma^n(x_k^i) < F(T_k^i) - F(z_k)\right] > 0$. However, there certainly exists $0 < \bar{r} \leq 1$ such that $\frac{1}{p}\sum_{j=1}^{p} V_j(x_k^i) r_j > F(t_k^i) - F(z_k)$ holds for r_j, $j = 1, \ldots, p$, that satisfy $r_j \geq \bar{r}$, implying that $Pr\left[\frac{1}{p}\sum_{j=1}^{p} V_j(x_k^i) r_j < F(T_k^i) - F(z_k)\right] = 0$ for $r_j \geq \bar{r}, j = 1, \ldots, p$. □

5 Numerical Experiments

In this section, the performance of the self-adaptive penalty technique when solving a benchmark set of CGO problems is investigated. Unless otherwise stated, we set $m = 50$. In the context of defining the reference point z, and the value of the penalty in (6), a point x is considered feasible if $\Sigma(x) \leq$ 1e-8.

First, we aim to analyze the effectiveness of the technique when using a meta-heuristic to compute a global minimizer of the penalty $\phi(x)$ in X, as defined by the BCGO problem (2). The FA, jDE, PSO, CMA-ES and ABC meta-heuristics are tested, using the parameter values as suggested in the papers [21,30,32–34]. For this experiment, the set g01–g13 of the g-collection[1] is used, noting that problems g03, g05, g11 and g13 have equality constraints

[1] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello Coello, C. Deb, Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. TR, Nanyang T.U., Sept. 18, 2006.

Table 1 'Average' and 'St.dev.' produced by the self-adaptive penalty algorithm.

P	FA		jDE		PSO		CMA-ES		ABC	
	average	St.dev.	average	St.dev.	average	St.dev.	average	St.dev.	average	St.dev.
g01	-14.99994	5.6e-05	-14.99700	5.6e-03	-14.88459	4.5e-02	-14.99999	1.1e-05	-14.99971	3.1e-04
g02	-0.50753	2.6e-02	-0.16931	1.5e-02	-0.40400	3.1e-02	-0.47232	1.4e-01	-0.17057	1.3e-02
g03	-0.99685	4.7e-03	-0.12897	2.9e-01	-0.31185	1.3e-01	-0.88107	1.9e-01	-0.18967	1.6e-01
g04	-30623.06	3.6e+01	-30662.87	3.8e+00	-30665.27	*2.0e-01*	-30665.39	*2.0e-01*	-30614.97	1.1e+02
g05	5176.680	7.6e+01	5198.139	1.3e+02	5323.849	2.7e+02	5564.546	4.5e+02	5144.948	1.8e+01
g06	-6961.46	1.5e-01	-6444.56	3.4e+02	-6483.57	3.3e+02	-6422.51	4.9e+02	-6871.53	1.6e+02
g07	32.10136	3.1e+00	37.37962	6.7e+00	36.54181	4.4e+00	24.81349	3.7e-01	29.76146	1.2e+01
g08	*-0.09583*	*2.8e-17*	-0.09500	1.4e-03	*-0.09583*	*2.8e-17*	*-0.09583*	*2.8e-17*	-0.09568	3.2e-04
g09	680.694	1.9e-02	798.558	7.5e+01	686.498	4.8e+00	681.031	1.0e-01	848.862	9.9e+01
g10	7119.26	1.7e+01	7184.60	2.2e+02	8542.70	3.6e+02	7670.40	5.3e+02	7594.84	2.3e+02
g11	0.74990	1.9e-06	0.95614	9.2e-02	0.83089	7.4e-02	0.74995	9.2e-02	0.74991	4.0e-06
g12	*-1.00000*	8.1e-11	-0.99896	2.2e-03	-0.99985	1.5e-04	*-1.00000*	0.0e+00	-0.99975	1.7e-04
g13	0.61042	1.6e-01	0.84229	2.0e-01	0.65304	2.4e-01	0.75379	2.1e-01	0.99988	2.5e-04

and the tolerance δ=1e-4 is used. In these comparisons, we stop the algorithms after 200000 function evaluations. The results are summarized in Table 1, where 'average' and 'St.dev.' represent the average and the standard deviation of the function values obtained by the algorithms after 20 runs. The best known optimal solutions, 'f^*', are displayed in Table 2. Best results (the wins) are 'underlined' and ties are in the 'italic' style. From the table it is possible to see that the FA has a larger number of wins than the others in both criteria. Overall, the self-adaptive penalty technique, with simple and easy to code meta-heuristics for solving the BCGO problem, is effective in finding global optimal solutions to CGO problems.

Table 2 Results from our study and from [3].

		our study					results in [3]		
P	f^*	F_b	F_o	best$_E$	worst$_E$	St.dev.$_E$	best$_E$	worst$_E$	St.dev.$_E$
g01	-15.0000000	1	1	3.000e-11	5.100e-10	1.255e-10	1.358e-06	9.166e-06	2.178e-06
g02	-0.803619	2.5	0.8	1.080e-03	4.285e-02	1.021e-02	3.836e-05	3.909e-02	1.117e-02
g03	-1.000000	0.1	1.5	2.822e-04	1.000e+00	1.970e-01	4.354e-09	7.854e-01	1.537e-01
g04	-30665.539	2.5	0.8	3.285e-04	3.491e-04	5.875e-06	1.035e-08	3.250e-06	7.105e-07
g05	5126.49810	1.5	0.8	1.352e-02	6.832e+01	2.045e+01	1.018e-10	3.468e+02	9.842e+01
g06	-6961.81388	1.2	0.8	5.688e-05	6.493e-04	1.727e-04	1.373e-10	1.291e-10	3.129e-12
g07	24.306209	2.5	0.8	3.260e-02	2.521e-01	6.163e-02	1.846e-05	1.467e-04	3.029e-05
g08	-0.095825	1	1	4.142e-08	4.142e-08	0	5.008e-11	5.008e-11	0
g09	680.630057	0.01	0.8	5.657e-03	4.292e-02	9.889e-03	2.16e-12	2.16e-12	0
g10	7049.33070	2.5	0.8	5.449e+01	6.484e+02	1.770e+02	7.900e+00	3.731e+01	8.385e+00
g11	0.7500000	1.5	0.8	7.642e-05	7.642e-05	4.850e-06	0	0	0
g12	-1.0000000	1	1	0	0	0	0	0	0
g13	0.0539498	0.01	0.8	3.947e-02	8.575e-01	1.796e-01	3.851e-01	9.107e-01	1.433e-01

For this set of problems $p_m = 1$ is used, except with problem g03 where $p_m = 0.4$.

Second, we aim to compare the hybrid self-adaptive penalty FA with other algorithms available in the literature. Three recently proposed adaptive penalty-based stochastic global optimizers [3, 4, 27] are used. When invoking the local intensification search in the FA, some parameters have been chosen to be problem dependent, namely p_m, F_b and F_o, with the goal of giving the best performances. To compare our results with those reported in [3] (an adaptive penalty-based DE algorithm), we stop the algorithm after 50000 function evaluations (as indicated in [3]). The results are summarized in Table 2, where 'best$_E$', 'worst$_E$' and 'St.dev.$_E$' represent the best error value, $f_{\text{best}} - f^*$, the worst error, and the standard deviation of the error values, based on 100 runs,

respectively. Although the results produced by our algorithm are satisfactory, they are not superior to those reported in [3] except for problems g01 and g13, being g12 a tie. A larger number of function evaluations would certainly be required for some problems. While the local search has provided good quality solutions, it has raised the computational effort.

Table 3 Results from our study and from [4].

	our study			results in [4]		
P	best	average	St.dev.	best	average	St.dev.
g01	-15.000000	-15.000000	0.000e+00	-15	-12.5	2.37254e+00
g02	-0.803603	-0.787892	1.379e-02	-0.8036	-0.7688	3.568e-02
g03	-0.980341	-0.962513	8.508e-03	-1.0	-0.2015	3.4508e-01
g04	-30665.538672	-30665.538672	1.866e-11	-30665.5	-30665.5	0
g05	5125.273729	5125.105038	6.083e-02	5126.4981	5126.4981	0
g06	-6961.813876	-6961.813876	9.331e-13	-6961.8	-6961.8	0
g07	24.312256	24.376587	5.044e-02	24.306	30.404	2.156839e+01
g08	-0.095825	-0.095825	2.848e-17	-0.09582	-0.09582	0
g09	680.630123	680.630848	4.258e-04	680.63	680.63	3e-05
g10	7103.509964	7279.735151	1.375e+02	7049.25	7351.17	5.2562430e+02
g11	0.749900	0.749900	8.050e-13	0.75	0.98749	5.590e-02

The results for g03 were obtained with $p_m = 0.1$, $F_b = 0.01$ and $F_o = 0.8$. For the other problems, the values are as previously defined.

When comparing our results with those produced by DUVDE+APM in [4] (the APM with dynamic use of DE variants), the subset g01–g11 is used. The results are summarized in Table 3, where the 'best', the 'average', and the 'St.dev.' of the solutions obtained in 20 independent runs, are shown. The algorithms terminate after 350000 function evaluations. The conclusions are

Table 4 Results from our study and from [27].

	our study			results in [27]		
P	best	average	St.dev.	best	average	St.dev.
g01	-15.000000	-15.000000	0.000e+00	-15.000	-14.552	7.0e-01
g02	-0.803585	-0.797191	4.850e-03	-0.803202	-0.755798	1.33210e-01
g03	-0.976735	-0.940689	1.024e-02	-1.000	-0.964	3.01e-01
g04	-30665.538672	-30665.538672	7.371e-12	-30665.401	-30659.221	2.043e+00
g05	5125.031908	5125.103381	3.639e-02	5126.907	5214.232	2.47476e+02
g06	-6961.813876	-6961.813876	9.214e-13	-6961.046	-6953.061	5.876e+00
g07	24.309466	24.347169	1.478e-02	24.838	27.328	2.172e+00
g08	-0.095825	-0.095825	5.624e-17	-0.095825	-0.095635	1.055e-03
g09	680.630196	680.631087	3.438e-04	680.773	681.246	3.22e-01
g10	7050.095847	7149.949024	4.839e+01	7069.981	7238.964	1.37773e+02
g11	0.749900	0.749900	1.125e-16	0.749	0.751	2e-03
g12	-1.000000	-1.000000	0.000e+00	-1.000000	-0.999940	1.41e-04
g13	0.353983	0.628807	1.136e-01	0.053941	0.286270	2.75463e-01

These results were obtained with the values of p_m, F_b and F_o defined for Table 2.

that our algorithm is able to produce comparative and high quality solutions when a larger number of evaluations is allowed.

Table 4 shows the results obtained after 50 runs, produced by our algorithm when solving the set g01–g13 with $m = 100$ and a maximum of 500000 function evaluations (as in [27], where a self-adaptive penalty-based GA, is used). The results of our study are in general superior to those reported in [27] and we reiterate the previous conclusions.

Now, we compare our algorithm with a modified ABC algorithm, that uses Deb's rules consisting of three simple heuristic rules for constraint handling [35]. The following conditions are considered: $m = 40$, 30 runs and a maxi-

Table 5 Results from our study and from [35].

	our study			results in [35]		
P	best	average	St.dev.	best	average	St.dev.
g01	-15.000000	-15.000000	0.00e+00	-15.000	-15.000	0.000
g02	-0.803470	-0.778942	1.348e-02	-0.803598	-0.792412	1.2e-02
g03	-1.000278	-0.999522	3.645e-04	-1.000	-1.000	0.000
g04	-30665.538673	-30665.538672	2.220e-11	-30665.539	-30665.539	0.000
g05	5153.670975	5451.215691	2.490e+02	5126.484	5185.714	7.5358e+01
g06	-6961.813876	-6961.813876	1.850e-12	-6961.814	-6961.813	2e-03
g07	24.320519	24.757232	5.157e-01	24.330	24.473	1.86e-01
g08	-0.095825	-0.095825	2.823e-17	-0.095825	-0.095825	0.000
g09	680.631787	680.641211	1.142e-02	680.634	680.640	4e-03
g10	7072.574892	7221.442900	9.565e+01	7053.904	7224.407	1.33870e+02
g11	0.749900	0.749900	1.129e-16	0.750	0.750	0.000
g12	-1.000000	-1.000000	0.000e+00	-1.000	-1.000	0.000
g13	0.056841	0.659425	1.764e-01	0.760	0.968	5.5e-02

These results were obtained with the values of p_m, F_b and F_o defined for Table 2.

mum of 240000 function evaluations (like in [35]). From the results in Table 5, it is possible to conclude that the hybrid self-adaptive penalty FA performs similarly to the modified ABC on nine problems, is better on g06 and g13 and is worse on g05 and g10.

Finally, a set of 20 problems available in `http://www.ime.usp.br/~egbirgin/`[2] is used. We aim to compare the herein proposed hybrid self-adaptive penalty FA with other penalty-type approaches. The comparison involves the results presented in [36], where an augmented Lagrangian framework is combined

[2] E.G. Birgin, C.A. Floudas, J.M. Martínez, Global minimization using an augmented Lagrangian method with variable lower-level constraints, TR MCDO121206, Jan. 22, 2007.

Table 6 Comparing our results with those in [36] and [37].

		our study			results in [36]			results in [37]	
P	f^*	best	median	n.f.e.(b)	best	median	n.f.e.(b)	sol.	n.f.e.
1	0.0293	0.0690	12.0945	59405	0.0342	0.1204	9608	0.0625	39575
2a	-400.00	-400.000	-380.7241	59420	-380.674	-369.111	15813	-134.1127	115107
2b	-600.00	-400.000	-366.330	59411	-385.051	-360.786	15808	-768.4569	120057
2c	-750.00	-749.999	-749.770	52238	-743.416	-693.743	15612	-82.9774	102015
2d	-400.00	-400.000	-399.980	26005	-399.910	-399.492	15394	-385.1704	229773
3a	-0.3888	-0.3882	-0.3837	62306	-0.3880	-0.3849	18928	-0.3861	48647
3b	-0.3888	-0.3888	-0.3881	2741	-0.3888	-0.3888	2589	-0.3888	3449
4	-6.6666	-6.6667	-6.6667	20825	-6.6667	-6.6667	2242	-6.6666	3547
5	201.1600	201.1593	201.1593	20824	201.159	201.159	2926	201.1593	14087
6	376.2919	376.2921	376.2939	20874	376.292	376.293	5617	0.4701	1523
7	-2.8284	-2.8284	-2.8283	20836	-2.8284	-2.8284	3434	-2.8058	13187
8	-118.700	-118.7049	-118.7048	20791	-118.705	-118.705	2884	-118.7044	7621
9	-13.4020	-13.4019	-13.4019	31068	-13.4018	-13.4017	5732	-13.4026	68177
10	0.74178	0.74179	0.74181	19551	0.7418	0.7418	6342	0.7420	6739
11	-0.5000	-0.5000	-0.5000	6141	-0.5000	-0.5000	3313	-0.5000	3579
12	-16.739	-16.7393	-16.6103	20765	-16.7389	-16.7389	98	-16.7389	3499
13	189.350	189.347	226.017	23514	189.345	189.347	9230	195.9553	8085
14	-4.5142	-4.5142	-4.5142	27267	-4.5142	-4.5142	6344	-4.3460	19685
15	0.0000	0.0000	0.0000	5696	0.0000	0.0000	2546	0.0000	1645
16	0.70492	0.7049	0.7049	1017	0.7049	0.7049	1850	0.7181	22593

For this set of problems we set $p_m = 0.5$, $F_b = 1$ and $F_o = 1$, except for problem 1 where $p_m = 1$, $F_b = 2.5$ and $F_o = 0.8$ and problem 13 where $p_m = 1$, $F_b = 0.2$ and $F_o = 0.8$.

For this experiment, we use $m = \min\{5n, 50\}$, ϵ=1e-5, η=1e-6 and $k_{\max} = 600$ (similarly to [36]).

with a meta-heuristic, known as artificial fish swarm algorithm, and those reported in [37], where a non-differentiable exact penalty function framework is implemented with the deterministic DIRECT algorithm. The results are summarized in Table 6, where 'best' is the best solution found among the 30

runs, 'median' is the median of the 30 solutions, and 'n.f.e.(b)' is the number of function evaluations to reach the value 'best'. The solution, 'sol.', the number of function evaluations, 'n.f.e.', reported in [37], and the best-known solution available in the literature, 'f^*', are also shown in the table. When we compare our results with those in [36], we conclude that the quality of the obtained solutions is comparable although a larger number of function evaluations are needed to reach those solutions. On the other hand, the quality of our solutions is superior to the one displayed by the penalty-based DIRECT algorithm [37].

6 Conclusions

We present a new self-adaptive penalty function that aims to penalize solutions, that violate the constraints of the problem, and is user-independent in the sense that penalty parameter values are set automatically by the information gathered from the violated constraints at each iteration. We establish the existence of an equivalence between the CGO problem and the BCGO problem with the self-adaptive penalty objective. The paper also shows the practical performance of a set of well-known meta-heuristics when solving the BCGO problem by demonstrating that they are effective in converging to the global solutions. Due to the superior performance of the recent FA meta-heuristic, the paper proposes a hybrid FA aiming to enhance the quality of the solutions. The convergence analysis of the algorithm has also been established. With the numerical experiments carried out with two sets of benchmark problems we demonstrate that the proposed self-adaptive penalty method is effective in

solving CGO problems. Future developments will be focused on solving higher dimensional optimization problems and reducing the computational effort in terms of function evaluations.

Acknowledgements The authors would like to thank the referees, the Associate Editor and the Editor-in-Chief for their valuable comments and suggestions to improve the paper. This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundação para a Ciência e Tecnologia within the projects UID/CEC/00319/2013 and UID/MAT/00013/2013.

References

1. Fiacco, A.V., McCormick, G.P.: Extensions of sumt for nonlinear programming: equality constraints and extrapolation. Manage. Sci. **12**(11), 816–828 (1966)
2. Gandomi, A.H., Yang, X.-S., Alavi, A.H.: Mixed variable structural optimization using firefly algorithm. Comput. Struct. **89**(23–24), 2325–2336 (2011)
3. Ali, M.M., Zhu, W.X.: A penalty function-based differential evolution algorithm for constrained global optimization. Comput. Optim. Appl. **54**(3), 707–739 (2013)
4. Silva, E.K., Barbosa, H.J.C., Lemonge, A.C.C.: An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization. Optimization and Engineering **12**(1–2), 31–54 (2011)
5. Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N.: Memetic particle swarm optimization. Ann. Oper. Res. **156**, 99–127 (2008)
6. Collange, G., Delattre, N., Hansen, N., Quinquis, I., Schoenauer, M.: Multidisciplinary optimization in the design of future space launchers. In: Breitkopf, P., Coelho, R.F. (eds.): Multidisciplinary Design Optimization in Computational Mechanics, Wiley, pp. 487–496 (2010)

7. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: Watanabe, O., Zeugmann, T. (eds.): Stochastic Algorithms: Foundations and Applications, Lecture Notes in Computer Sciences, vol. 5792, pp. 169–178 Springer, Berlin, Heidelberg (2009)
8. Fister, I., Fister, Jr. I., Yang, X.-S., Brest, J.: A comprehensive review of firefly algorithms. Swarm Evolut. Comput. **13**, 34–46 (2013)
9. Yang, X.-S.: Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio Inspir. Com. **2**(2), 78–84 (2010)
10. Yu, S., Yang, S., Su, S.: Self-adaptive step firefly algorithm. J. Appl. Math. **2013**, Article ID 832718, 8 pages (2013)
11. Wang, H., Wang, W., Sun, H., Rahnamayan, S.: Firefly algorithm with random attraction. Int. J. Bio Inspir. Com. **8**(1) 33–41 (2016)
12. Farahani, Sh.M., Abshouri, A.A., Nasiri, B., Meybodi, M.R.: Some hybrid models to improve firefly algorithm performance. International Journal of Artificial Intelligence **8**(S12), 97–117 (2012)
13. Guo, L., Wang, G.-G., Wang, H., Wang, D.: An effective hybrid firefly algorithm with harmony search for global numerical optimization. The Scientific World Journal **2013**, Article ID 125625, 9 pages (2013)
14. Li, H., Ye, C.: Firefly algorithm on multi-objective optimization of production scheduling system. Advances in Mechanical Engineering and its Applications **3**(1), 258–262 (2012)
15. Yang, X.-S., He, X.: Firefly algorithm: recent advances and applications. International Journal of Swarm Intelligence **1**(1), 36–50 (2013)
16. Yang, X.-S., Hosseini, S.S.S., Gandomi, A.H.: Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Appl. Soft Comput. **12**(3), 1180–1186 (2012)
17. Cheung, N. J., Ding, X.-M., Shen, H.-B.: Adaptive firefly algorithm: parameter analysis and its application. PLoS ONE **9**(11) e112634 (2014)
18. Wang, H., Zhou, X., Sun, H., Yu, X., Zhao, J., Zhang, H., Cui, L.: Firefly algorithm with adaptive control parameters. Soft Computing (2016) DOI: 10.1007/s00500-016-2104-3
19. Cheung, N. J., Ding, X.-M., Shen, H.-B.: A non-homogeneous firefly algorithm and its convergence analysis. J. Optim. Theory Appl. **170**(2) 616–628 (2016)

20. Baykasoğlu, A., Ozsoydan, F.B.: An improved firefly algorithm for solving dynamic multidimensional knapsack problems. Expert Syst. Appl. **41**(8), 3712–3725 (2014)
21. Costa, M.F.P., Rocha, A.M.A.C., Francisco, R.B., Fernandes, E.M.G.P.: Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization. Advances in Operations Research **2014**, Article ID 215182, 12 pages (2014)
22. Costa, M.F.P., Rocha, A.M.A.C., Francisco, R.B., Fernandes, E.M.G.P.: Firefly penalty-based algorithm for bound constrained mixed-integer nonlinear programming. Optimization **65**(5), 1085–1104 (2016)
23. Sayadi, M.K., Hafezalkotob, A., Naini, S.G.J.: Firefly-inspired algorithm for discrete optimization problems: an application to manufacturing cell formation. J. Manuf. Syst. **32**(1), 78–84 (2013)
24. Srivatsava, P.R., Mallikarjun, B., Yang, X.-S.: Optimal test sequence generation using firefly algorithm. Swarm Evolut. Comput. **8**, 44–53 (2013)
25. Alb, M., Alotto, P., Magele, C., Renhart, W., Preis, K., Trapp, B.: Firefly algorithm for finding optimal shapes of electromagnetic devices. IEEE Trans. Magn. **52**(3) 1–4 (2016)
26. Barbosa, H.J.C., Lemonge, A.C.C.: An adaptive penalty method for genetic algorithms in constrained optimization problems. In: Iba, H. (ed.), Frontiers in Evolutionary Robotics, pp. 9–34, I-Tech Education and Publishing, Austria (2008)
27. Tessema, B., Yen, G.G.: A self adaptive penalty function based algorithm for constrained optimization. In: IEEE Congress on Evolutionary Computation, pp. 246–253 (2006)
28. Tessema, B., Yen, G.G.: An adaptive penalty formulation for constrained evolutionary optimization. IEEE Trans. Syst. Man Cybern. A Syst. Hum. **39**(3), 565–578 (2009)
29. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
30. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**, 646–657 (2006)
31. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of 1995 IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)

32. Ali, M.M, Kaelo, P.: Improved particle swarm algorithms for global optimization. Appl. Math. Comput. 196(2), 578–593 (2008)

33. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)

34. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. **39**(3), 459–471 (2007)

35. Karaboga, D., Akay, B.: A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems. Appl. Soft Comput. **11**(3), 3021–3031 (2011)

36. Costa, M.F.P., Rocha, A.M.A.C., Fernandes, E.M.G.P.: An artificial fish swarm algorithm based hyperbolic augmented Lagrangian method. J. Comput. Appl. Math. **259**(Part B), 868–876 (2014)

37. Di Pillo, G., Lucidi, S., Rinaldi, F.: An approach to constrained global optimization based on exact penalty functions. J. Glob. Optim. **54**(2), 251–260 (2012)