**8th World Congress on Structural and Multidisciplinary Optimization**
June 1-5, 2009, Lisbon, Portugal

# Self-Adaptive Penalties in the Electromagnetism-like Algorithm for Constrained Global Optimization Problems

**Ana Maria A.C. Rocha**[1] **& Edite M.G.P. Fernandes**[2]

Department of Production and Systems, School of Engineering, University of Minho, 4710-057 Braga, Portugal

email: [1]arocha@dps.uminho.pt; [2]emgpf@dps.uminho.pt

**1. Abstract**
A well-known approach for solving constrained optimization problems is based on penalty functions. A penalty technique transforms the constrained problem into an unconstrained problem by penalizing the objective function when constraints are violated and then minimizing the penalty function using methods for unconstrained problems. In this paper, we analyze the implementation of a self-adaptive penalty approach, within the electromagnetism-like population-based algorithm, in which the constraints that are more difficult to be satisfied will have relatively higher penalty values. The penalties depend upon the level of constraint violation scaled by the average of the objective function values. Numerical results obtained with a collection of well-known global optimization problems are presented and a comparison with other stochastic methods is also reported.
**2. Keywords:** Global optimization, electromagnetism-like algorithm, penalty technique, adaptive penalty

## 3. Introduction

This paper aims to illustrate the behavior of self-adaptive penalty techniques combined with the electromagnetism-like algorithm for solving nonlinear global optimization problems of the form:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0,\ i = 1, \ldots, p \\ & h_j(x) = 0,\ j = 1, \ldots, m \\ & x \in \Omega, \end{array} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^p$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are nonlinear continuous functions and $\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\}$. We do not assume that the objective function $f$ is convex. There may be many local minima in the feasible region. This class of global optimization problems arises frequently in engineering applications. Specially for large scale problems of type Eq.(1), derivative-free and stochastic methods are the most well-known and used methods. The two main categories of methods to handle constraints in these algorithms are listed below.

1. Methods based on penalty functions. The constraints violation is combined with the objective function to define a penalty function. This function aims to penalize infeasible solutions by increasing its fitness value proportionally to their level of constraints violation. The reader is referred to the papers [1, 2, 4, 10, 11] and the references therein reported.

2. Methods based on biasing feasible over infeasible solutions. They seem to be nowadays interesting alternatives to penalty methods for handling constraints. In this type of methods, constraints violation and the objective function are used separately and optimized by some sort of order, being the constraints violation the most important. This approach is possible in population-based methods but cannot be used with classical point-to-point search methods. See, for example, [5, 12, 13, 14].

Here, we are interested in the electromagnetism-like (EM) algorithm proposed in [2, 3]. This is a stochastic population-based algorithm that simulates the electromagnetism theory of physics by considering each point in the population as an electrical charge. The method uses an attraction-repulsion mechanism to move a population of points towards optimality. The EM algorithm was specifically designed for solving optimization problems with bound constraints. In [2], classical penalty and barrier methods are proposed and tested. We propose the introduction of a self-adaptive penalty technique to handle the equality and inequality constraints of the problem described in Eq.(1).

The remainder of this paper is organized as follows. Section 4 briefly introduces the adaptive penalty framework and Section 5 describes the modifications that are introduced in the EM algorithm to handle the constraints by a penalty framework. Section 6 contains the results of all the numerical experiments, including comparisons with other methods, and we conclude the paper with the Section 7.

## 4. Penalty functions in stochastic algorithms

Most stochastic methods for global optimization are developed primarily for unconstrained or simple bound constrained problems. Then they are extended to constrained optimization problems using, for example, a penalty technique. This technique transforms the constrained problem into an unconstrained problem by penalizing the objective function when constraints are violated. The objective penalty function, in the unconstrained problem, depends on a positive penalty parameter that must be updated throughout the iterative process. The parameter updating rule that gives the best performance of a penalty technique is still nowadays an open issue.

Here, we introduce self-adaptive penalties in the original procedures of the EM algorithm, proposed in [3], to easily handle equality and inequality constraints. The original EM method was developed for bound constrained optimization problems. The equality constraints of the problem are converted into inequality constraints, $|h_j| \leq \epsilon$, $j = 1, \ldots, m$ for $\epsilon > 0$. For simplicity, the problem in Eq.(1) is rewritten as

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & c(x) \leq 0 \\ & x \in \Omega, \end{array} \tag{2}$$

where the vector of the inequality constraints is defined by

$$c(x) = (g_1(x), \ldots, g_p(x), |h_1(x)| - \epsilon, \ldots, |h_m(x)| - \epsilon)\,,$$

and the level of constraints violation is measured by the vector

$$v_j(x) = \max\{0, c_j(x)\},\ j = 1, \ldots, p + m\,. \tag{3}$$

In the sequel, a point $x$ with $\sum_j v_j(x) = 0$ is feasible, whereas if $\sum_j v_j(x) > 0$ then the point is infeasible. The basic penalty approach defines a fitness for each point $x$, herein denoted by $\Phi(x)$, by adding to the objective function value a penalty term, $P(x)$, that aims to penalize infeasible solutions. Making use of a suggestion presented in [9], we adopt a more elaborated definition of fitness that sets an average function value to infeasible points that have rather small objective function values,

$$\Phi(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible} \\ \mathcal{F}(x) + P(x), & \text{otherwise} \end{cases} \tag{4}$$

where

$$\mathcal{F}(x) = \begin{cases} f(x), & \text{if } f(x) > \langle f(x) \rangle \\ \langle f(x) \rangle, & \text{otherwise} \end{cases}$$

and

$$\langle f(x) \rangle = \frac{\sum_{i=1}^{p_{size}} f(x^i)}{p_{size}}$$

is the average of the function values over the population of size $p_{size}$. Different penalties may emerge depending on the way penalties vary throughout the minimization process. A brief summary of the most used penalties follows.

### 4.1. Static penalties

In static penalties, a typical penalty term $P(x)$ depends on user defined positive parameters and has the following form

$$P(x) = \sum_{j=1}^{p+m} \mu_{j,l} (v_j(x))^2$$

where $\mu_{j,l}$ are the penalty parameters and $l = 1, \ldots, l_v$, being $l_v$ the number of levels of violation defined by the user. This strategy uses deterministic rules to define different parameter values for each constraint in order to balance constraints separately. When the parameter values are too large, the method mostly rejects infeasible solutions. If the penalty parameters are too small, the method might converge to an

infeasible solution. The reader is referred to [4] and the references therein reported.

4.2. Dynamic penalties

Penalties that change over time, denoted as dynamic penalties, have been used for some time and are summarized in [8] as

$$P(x) = \mu(\text{it}) \sum_{j=1}^{p+m} (v_j(x))^{\gamma(v_j(x))}$$

where $\mu(\text{it})$ is a dynamically modified penalty parameter and "it" represents the iteration counter. Interesting and quite efficient rules found in the literature are: $\mu(\text{it}) = (c\,\text{it})^{\alpha}$, where $c$ and $\alpha$ are user defined constants [10], and $\mu(\text{it}) = \text{it}\sqrt{\text{it}}$ [11]. The penalty parameter does not depend on the number of constraints although the pressure on infeasible solutions increases as "it" increases. The power of the constraint violation, $\gamma(.)$, is a violation dependent constant: $\gamma(z) = 1$ if $z \leq 1$, and $\gamma(z) = 2$, otherwise. See, for example, in [10, 11].

4.3. Adaptive penalties

In the adaptive penalty methods, the user does not need to specify any parameter values to define the penalty updating. The penalties are updated for every iteration according to information gathered from the population, in particular the average of the objective function and the level of constraint violation during the iterative process. In this paper, the penalty term in this self-adaptive penalty context is written as below

$$P(x) = \sum_{j=1}^{p+m} \mu_j (v_j(x))^{\gamma(v_j(x))}. \tag{5}$$

The idea of penalizing the most difficult constraints using the $\gamma(.)$ function in the penalty term, as described above, is also used. Further, constraints which are more difficult to be satisfied will have a relatively higher penalty value. Adapting the ideas presented in [9], we propose the following formula for $\mu_j$:

$$\mu_j = K \frac{\sum_{i=1}^{p_{size}} v_j(x^i)}{\sum_{k=1}^{m+p} \sum_{i=1}^{p_{size}} v_k(x^i)}\,, \quad j = 1, \ldots, p+m \tag{6}$$

where

$$K = \left| \sum_{i=1}^{p_{size}} f(x^i) \right|$$

is a problem dependent factor based on the function values of the population, and $x^i$ represents the $i$th point of the population. The numerator in Eq.(6) $\sum_{i=1}^{p_{size}} v_j(x^i)$ measures the violation of the $j$th constraint and the penalty for each constraint reflects the fraction of its violation over the total violation of the current population. The proposal in [9, 1] uses the square of each violation $v_k$ in the denominator of Eq.(6). In both cases, the penalty values for each iteration depend on the level of violation of each constraint and are scaled by a factor that depends on objective function values.

A different formula to penalize differently constraint violations may be used. Defining a zero penalty for constraints that are not violated and increasing penalties as violation increases is the property of the herein proposed formula:

$$\mu_j = K \left( \exp\left( \frac{\sum_{i=1}^{p_{size}} v_j(x^i)}{\sum_{k=1}^{m+p} \sum_{i=1}^{p_{size}} v_k(x^i)} \right)^{l} - 1 \right)\,, \quad j = 1, \ldots, p+m \tag{7}$$

where $l \in \{1, 2\}$. Figure 1 illustrates the behavior of the penalty values $\mu_j$ obtained by Eq.(6) and Eq.(7) ($l = 1, 2$), as a function of the argument "frac" $= \sum_{i=1}^{p_{size}} v_j(x^i) / \sum_{k=1}^{m+p} \sum_{i=1}^{p_{size}} v_k(x^i)$. We remark that in the two cases defined in Eq.(7) the penalty increases faster with the argument "frac" than that of case Eq.(6).

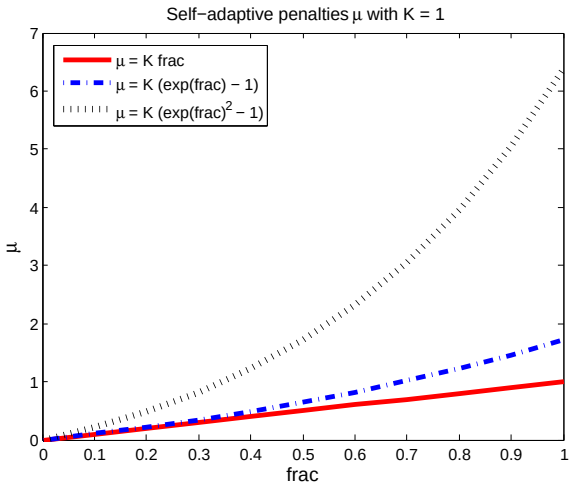


Figure 1: Behavior of self-adaptive penalties $\mu_j$.

## 5. The self-adaptive penalty electromagnetism-like algorithm

In this section, we briefly describe how the self-adaptive penalty approach is incorporated into the main procedures of the original EM algorithm, so that equality and inequality constraints are easily handled. In the first step of the EM algorithm [3], a population of $p_{size}$ points is randomly generated in the set $\Omega$. The best point of the population, $x^{best}$, is then identified. We use the following notation: $x^i$ denotes the $i$th point in the population, $x^i_k$ represents the $k$th coordinate of point $x^i$, where $k = 1, \ldots, n$. In the second step, the algorithm computes the total force exerted on each point by the other points of the population, using the Coulomb's law. The third step is used to move the points according to the corresponding total forces. Finally, a local search procedure to accelerate the convergence speed in the final stage of each iteration is implemented. The main procedures of our self-adaptive penalty electromagnetism-like algorithm are described below.

### 5.1. Initialization of the population

The initial population of $p_{size}$ points is randomly generated. Each point $x^i$ in the population is componentwise computed by

$$x^i_k = l_k + \lambda(u_k - l_k), \text{ for } k = 1, \ldots, n$$

where $\lambda \sim U(0, 1)$. The fitness of each point is evaluated, $\Phi(x^i)$, $i = 1, \ldots, p_{size}$, using Eq.(4), and the best point, $x^{best}$ is identified, where $x^{best} = \arg\min_i \Phi(x^i)$.

### 5.2. Charge and force vector calculation

According to Coulomb's law, the force exerted on the point $x^i$, by other point $x^j$, is directly proportional to the product of their charge and is inversely proportional to the square of the distance between the points.

The charge $q^i$ of point $x^i$ aims to determine the power of attraction or repulsion for that point. Points that have better fitness should possess higher charges and attract points with worst fitness values. Here, each charge relies on a scaled distance of the fitness value at $x^i$ to the fitness of the best point in the population:

$$q^i = \exp\left(\frac{-n(\Phi(x^i) - \Phi(x^{best}))}{\Phi(x^{worst}) - \Phi(x^{best})}\right) \tag{8}$$

for $i = 1, \ldots, p_{size}$. The distance is scaled by the range of fitness of all points of the population, where $x^{worst} = \arg\max_i \Phi(x^i)$. Since the charges $q^i$ and $q^j$ are positive, the direction of each individual component force $F^i_j$ is defined using the fitness at $x^i$ and $x^j$. Thus, if $\Phi(x^j) < \Phi(x^i)$ then the point $x^j$ attracts $x^i$, whereas if $\Phi(x^j) \geq \Phi(x^i)$ then $x^j$ repels $x^i$. Finally, the total force vector $F^i$ exerted on each point $x^i$ $(i = 1, 2, \ldots, p_{size})$ by the other $p_{size} - 1$ points is calculated by adding the individual

component forces, $F^i_j$, between any pair of points $x^i$ and $x^j$ :

$$F^i = \sum_{j \neq i}^{p_{size}} F^i_j \equiv \begin{cases} (x^j - x^i)\frac{q^i q^j}{\|x^j - x^i\|^3} & \text{if } \Phi(x^j) < \Phi(x^i) \quad (x^j \text{ attracts } x^i) \\ \\ (x^i - x^j)\frac{q^i q^j}{\|x^j - x^i\|^3} & \text{if } \Phi(x^i) \geq \Phi(x^j) \quad (x^j \text{ repels } x^i) \end{cases} . \tag{9}$$

Figure 2 illustrates an example with a population of 3 points. The total force $F^1$ exerted on $x^1$, by $x^2$ and $x^3$, is given by $F^1 = F^1_2 + F^1_3$. Since $\Phi(x^3)$ is better than $\Phi(x^1)$, $F^1_3$ is an attractive force exerted on $x^1$ by $x^3$. On the other hand, $\Phi(x^2)$ is worse than $\Phi(x^1)$, and $F^1_2$ is a repulsive force exerted on $x^1$ by $x^2$.

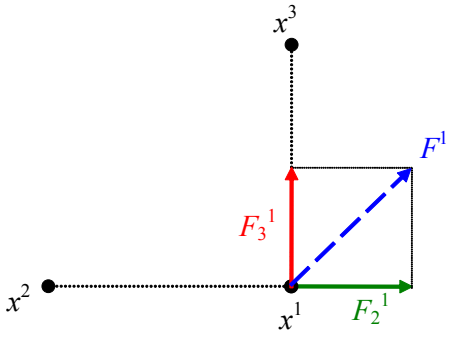


Figure 2: The total force $F^1$ exerted on $x^1$ by $x^2$ and $x^3$.

5.3. Movement along the total force

The total force is used to move the point $x^i$ in the direction of the force by a random step length $\lambda$. The best point, $x^{best}$, is not moved and is carried out to the subsequent iteration. There are two alternative and common ways of moving a point along a search direction while satisfying the bound constraints. One uses the normalized direction and a scaling factor that defines the allowed range movement towards the bounds of the set $\Omega$. The other takes the step along the direction and projects the new point onto the feasible set $\Omega$. Both are described below and tested with a benchmark set of constrained problems.

5.3.1. Movement scaled by the allowed range

Here, we describe the equations that are used to move the points according to the normalized total force, scaling by the allowed range movement towards the lower bounds $l_k$ and upper bounds $u_k$:

$$x^i_k = \begin{cases} x^i_k + \lambda \frac{F^i_k}{\|F^i\|}(u_k - x^i_k) & \text{if } F^i_k > 0 \\ \\ x^i_k + \lambda \frac{F^i_k}{\|F^i\|}(x^i_k - l_k) & \text{otherwise} \end{cases} \tag{10}$$

for $k = 1, 2, \ldots, n$, $i = 1, 2, \ldots, p_{size}$ and $i \neq best$, and $\lambda \sim U(0, 1)$. At the end of this step, the fitness of all points in the population are required so that the best point could be identified. We remark that the above defined movement may not allow $x^i$ to reach the bounds.

5.3.2. Movement followed by projection

Another well-known strategy relies on the movement along the total force followed by a projection of the new point onto the set $\Omega$:

$$\begin{aligned} x^i &= x^i + \lambda F^i \\ \\ x^i_k &= \begin{cases} l_k & \text{if } x^i_k < l_k \\ u_k & \text{if } x^i_k > u_k \\ x^i_k & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

for $k = 1, 2, \ldots, n$, $i = 1, 2, \ldots, p_{size}$ and $i \neq best$ ($\lambda \sim U(0, 1)$).

5.4. Elitist descent search

As a local search, we use an elitist approach by refining a predefined region of $x^{best}$, in order to improve accuracy of the best solution found so far. The search relies on an approximate descent direction for the fitness function.

First, two exploring points $x_i^{rand}$, $i = 1, 2$, in a neighborhood of the best point $x^{best}$, of ray $\varepsilon_r$ (positive), are randomly generated. Based on these 2 points, we use an idea proposed in [7] to generate an approximate descent direction, $d$, for $\Phi$, at the best point $x^{best}$,

$$d = -\frac{1}{\sum_{j=1}^{2} |\Delta\Phi_j|} \sum_{i=1}^{2} \Delta\Phi_i \frac{x^{best} - x_i^{rand}}{\|x^{best} - x_i^{rand}\|}, \tag{12}$$

where $\Delta\Phi_j = \Phi(x^{best}) - \Phi(x_j^{rand})$. A new trial point $y \in \Omega$, is computed using the normalized descent direction with a prescribed step size $\alpha \in (0, 1]$. If $\Phi(x^{best}) < \Phi(y)$ then $y$ is discarded, the step size is halved (*i.e.*, $\alpha \leftarrow \alpha/2$) and a new point is evaluated along that descent direction. This classical backtracking strategy is implemented, for at least $LsIt_{max}$ iterations. Otherwise, an approximate descent direction for $f$, at $y$, is computed, the step size, $\alpha$, is reset to 1 and the process is repeated. We refer to [12, 13] for details.

## 6. Numerical results

In this section, we report the numerical results obtained by running a set of 13 benchmark constrained global problems, described in full detail in the Appendix of [16]. The problems are known as g01, g02, ..., g13. We remark that g02, g03, g08 and g12 are maximization problems while the others are minimization ones. The former were solved as minimization problems. The algorithm is coded in the C programming language and it contains an interface to connect to AMPL so that the problems coded in AMPL could be easily solved [6]. The computational tests were performed on a PC with a 3GHz Pentium IV microprocessor and 1Gb of memory. Details of the selected problems are reported in Table 1, where "Prob" identifies the problem, "Type of $f$" lists the characteristic of the objective function, $n$ represents the number of variables, $p$ and $m$ are the number of inequality and equality constraints respectively, $n_{act}$ is the number of active constraints at the solution and $f_{global}$ is the best solution known in the literature.

Table 1: Details of the selected problems.

| Prob | Type of $f$ | $n$ | $p$ | $m$ | $n_{act}$ | $f_{global}$ |
|---|---|---|---|---|---|---|
| g01 | quadratic | 13 | 9 | 0 | 6 | -15.000000 |
| g02 | nonlinear | 20 | 2 | 0 | 1 | 0.803619 |
| g03 | polynomial | 10 | 0 | 1 | 1 | 1.000500 |
| g04 | quadratic | 5 | 6 | 0 | 2 | -30665.539 |
| g05 | cubic | 4 | 2 | 3 | 3 | 5126.497 |
| g06 | cubic | 2 | 2 | 0 | 2 | -6961.814 |
| g07 | quadratic | 10 | 8 | 0 | 6 | 24.30621 |
| g08 | nonlinear | 2 | 2 | 0 | 0 | 0.095825 |
| g09 | polynomial | 7 | 4 | 0 | 2 | 680.630 |
| g10 | linear | 8 | 6 | 0 | 6 | 7049.248 |
| g11 | quadratic | 2 | 0 | 1 | 1 | 0.749900 |
| g12 | quadratic | 3 | 1 | 0 | 0 | 1.000000 |
| g13 | nonlinear | 5 | 0 | 3 | 3 | 0.053942 |

6.1. Setting general parameters

When stochastic methods are used to solve problems, the impact of the random number seeds has to be taken into consideration, and each algorithm should be run on each problem a certain number of times, herein denoted by $n_{runs}$. The values stated to the general parameters are as below:

- to convert equality constraints into inequalities, $\epsilon = 0.001$ is used;
- the used ray of the neighborhood of $x^{best}$ is $\varepsilon_r = 0.001$;

- the maximum number of iterations in the local search is $LsIt_{max} = 10$.

During the execution of each run, the best feasible solution is registered $f^j_{best}$, $j = 1, \ldots, n_{runs}$. The values reported in the subsequent tables are

$$\text{best} = \min_{j=1,\ldots,n_{runs}} f^j_{best}, \quad \text{average} = \frac{\sum_{j=1}^{n_{runs}} f^j_{best}}{n_{runs}},$$

which are the best solution and the average solution obtained after the specified runs, respectively.

6.2. Comparison between different self-adaptive penalties

In order to analyze the practical performance of both alternatives for moving points along the total force, see Eq.(10) and Eq.(11), in this self-adaptive penalty electromagnetism-like algorithm, we solved the 13 problems using the 3 versions of adaptive penalties Eq.(6) and Eq.(7) with $l = 1, 2$. In these experiments, a population of 50 points is used, each algorithm was executed until it reached 350000 function evaluations, and $n_{runs} = 30$.

Table 2 reports the best and average results obtained after the 30 runs with the point movement based on Eq.(10). We mark with bold typeface the best solutions found in each study. Table 3 contains the obtained results when using the Eq.(11) to move the points. Comparing the results in both tables, we may conclude that Eq.(10) gives better solutions in 9 problems: g01, g02, g03, g06, g08, g10, g11, g12, g13. It seems that Eq.(6) is able to provide solutions with higher accuracy, followed by the Eq.(7) with $l = 1$.

Table 2: Results for the self-adaptive penalties Eq.(6) and Eq.(7), for $l = 1, 2$, considering movement in Eq.(10).

| | $\mu_j$ in Eq.(6) | | $\mu_j$ in Eq.(7) for $l = 1$ | | $\mu_j$ in Eq.(7) for $l = 2$ | |
|---|---|---|---|---|---|---|
| Prob | best | average | best | average | best | average |
| g01 | -14.981720 | -13.364670 | **-14.992800** | -12.188632 | -14.977760 | -12.031448 |
| g02 | **-0.797711** | -0.751747 | -0.783135 | -0.727828 | -0.774127 | -0.712425 |
| g03 | **-0.999807** | -0.964702 | -0.999794 | -0.997730 | -0.999212 | -0.994997 |
| g04 | **-30614.3** | -30567.4 | -30596.9 | -30516.7 | -30611.4 | -30541.6 |
| g05 | 5127.90 | 5320.67 | 5128.22 | 5287.70 | **5127.25** | 5358.54 |
| g06 | **-6961.71** | -6961.14 | -6961.64 | -6960.71 | -6961.60 | -6960.51 |
| g07 | 40.58663 | 51.73630 | **33.09758** | 58.38708 | 38.124170 | 54.523665 |
| g08 | **-0.095825** | -0.095821 | -0.095825 | -0.095816 | -0.095825 | -0.095812 |
| g09 | 680.8360 | 681.2683 | 680.9865 | 682.0359 | **680.8002** | 681.9273 |
| g10 | **7081.32** | 7790.85 | 7176.04 | 7822.19 | 7103.26 | 7691.32 |
| g11 | **0.749000** | 0.749001 | 0.749000 | 0.749004 | 0.749001 | 0.749004 |
| g12 | **-0.999925** | -0.999524 | -0.999916 | -0.998960 | -0.999913 | -0.999020 |
| g13 | 0.082410 | 0.885214 | **0.079990** | 0.930806 | 0.0945563 | 1.168597 |

6.3. Comparison with other penalty stochastic-based methods

In this final part of the paper, we compare the results obtained with the proposed self-adaptive penalty electromagnetism-like algorithm with those reported in the paper [1], where an adaptive penalty is implemented in a genetic algorithm context. The authors test their adaptive penalty algorithm with the problems g01, ..., g11. The authors propose five variants of the algorithm. The results listed in the Table 4 are from the original version which is the most similar to the adaptive penalty version herein proposed. The table contains the best and the average solutions found with the following set of parameters:

- the algorithms were allowed to run for 5000 iterations;
- the population has 100 points;

- $n_{runs} = 25$ ;

that are the same as those used in [1]. The first two columns of results correspond to the movement that uses the normalized force scaled by the allowed range, see Eq.(10). The next two columns correspond to the well-known move and project onto the bounds strategy, see Eq.(11). Our reported results correspond to the best solutions found after running the three adaptive formulae for $\mu$ (see Eq.(6) and Eq.(7) for $l = 1, 2$). It seems that in general the best results are obtained with Eq.(10), contradicting some user's suggestions. For problems g04 and g07 the version based on Eq.(11) gives much better solutions, and for problems g06 and g09 gives slightly better solutions. When comparing our results with Barbosa and Lemonge's results ([1]) we note that our self-adaptive penalty algorithm gives better results in problems g05, g08, g10 and g11, and is competitive in problems g01, g06 and g09.

Table 3: Results for the self-adaptive penalties Eq.(6) and Eq.(7), for $l = 1, 2$, considering movement in Eq.(11).

| | $\mu_j$ in Eq.(6) | | $\mu_j$ in Eq.(7) for $l = 1$ | | $\mu_j$ in Eq.(7) for $l = 2$ | |
|---|---|---|---|---|---|---|
| Prob | best | average | best | average | best | average |
| g01 | -11.18420 | -8.49607 | -11.701710 | -8.982671 | **-12.070230** | -8.789214 |
| g02 | -0.380851 | -0.309595 | -0.360710 | -0.298674 | **-0.387869** | -0.300699 |
| g03 | **-0.999571** | -0.994274 | -0.999526 | -0.959930 | -0.999155 | -0.975950 |
| g04 | **-30665.39** | -30655.23 | -30665.25 | -30656.93 | -30665.14 | -30657.80 |
| g05 | 5128.90 | 5341.72 | 5126.54 | 5284.06 | **5126.50** | 5222.61 |
| g06 | -6961.557 | -6960.709 | **-6961.633** | -6960.378 | -6961.627 | -6960.799 |
| g07 | 26.986460 | 30.529847 | **25.957410** | 30.486127 | 27.679550 | 30.556134 |
| g08 | -0.095823 | -0.095795 | **-0.095825** | -0.093585 | -0.095824 | -0.091350 |
| g09 | **680.6794** | 681.8607 | 680.7594 | 681.5287 | 680.7005 | 681.6833 |
| g10 | 8287.77 | 10096.51 | **7572.76** | 9769.70 | 7753.54 | 9981.89 |
| g11 | **0.749000** | 0.749004 | 0.749000 | 0.749007 | 0.749001 | 0.749005 |
| g12 | -0.997125 | -0.989470 | **-0.998174** | -0.990618 | -0.997586 | -0.989686 |
| g13 | 0.099175 | 0.914603 | **0.090531** | 1.630516 | 0.155731 | 1.509456 |

To further examine the performance of the proposed self-adaptive penalty EM algorithm, we compare the results of 5 problems reported in [2], where classical penalty and barrier methods are incorporated into the EM algorithm. For a fair comparison we use the therein reported conditions:

- TP1 – $p_{size} = 30$; maximum number of iterations= 75, and $f_{global} = -30665.539$
- TP2 – $p_{size} = 40$; maximum number of iterations= 100, and $f_{global} = -310$
- TP3 – $p_{size} = 20$; maximum number of iterations= 50, and $f_{global} = -5.508013$
- TP4 – $p_{size} = 30$; maximum number of iterations= 50, and $f_{global} = -83.254$
- TP5 – $p_{size} = 20$; maximum number of iterations= 75, and $f_{global} = -5.7398$.

The results are reported in Table 5, where we use "TP#" to identify the problem, in the first column. In the next four columns we list our best results, and register the formulae used to get them. In the table, $nfe_{avg}$ corresponds to the average number of function evaluations computed over the $n_{runs} = 10$ runs. The last four columns of the table include the results reported in [2], where "P" means penalty method and "B" stands for barrier method.

Table 4: Best and average results from our study and [1].

| Prob | our study with Eq.(10) best | our study with Eq.(10) average | our study with Eq.(11) best | our study with Eq.(11) average | in [1] best | in [1] average |
|---|---|---|---|---|---|---|
| g01 | **-14.996420** | -14.459611 | -12.120460 | -8.995533 | -14.999980 | -14.952633 |
| g02 | **-0.788632** | -0.751887 | -0.403694 | -0.315550 | -0.8015556 | -0.772480 |
| g03 | **-0.999471** | -0.998131 | -0.998764 | -0.978430 | -1.000490 | -1.000404 |
| g04 | -30631.430 | -30556.277 | **-30665.480** | -30661.708 | -30665.523 | -30665.095 |
| g05 | **5126.508** | 5290.027 | 5126.654 | 5289.862 | 5127.3606 | 5321.5714 |
| g06 | -6961.563 | -6960.594 | **-6961.726** | -6960.783 | -6961.796 | -6742.343 |
| g07 | 38.813650 | 62.912459 | **26.944410** | 30.732890 | 24.416253 | 26.457358 |
| g08 | **-0.095825** | -0.095819 | -0.095824 | -0.095805 | -0.095825 | -0.087690 |
| g09 | 680.8378 | 681.9545 | **680.8213** | 681.4629 | 680.6334 | 680.7851 |
| g10 | **7068.0150** | 7857.3535 | 7971.2070 | 10254.4451 | 7205.1436 | 8392.4000 |
| g11 | **0.749000** | 0.749002 | 0.749000 | 0.749003 | 0.752354 | 0.855617 |

Table 5: Best and average results from our study and [2].

| Prob | our study best | our study average | our study $nfe_{avg}$ | our study | in [2] best | in [2] average | in [2] $nfe_{avg}$ | |
|---|---|---|---|---|---|---|---|---|
| TP1 | -30389.66 | -30350.36 | 3095 | Eq.(7)+Eq.(10) | -30563.29 | -30374.47 | 2534 | "P" |
| | | | | | -30596.78 | -30447.68 | 2264 | "B" |
| TP2 | -293.1590 | -280.9349 | 5128 | Eq.(7)+Eq.(10) | -297.7095 | -293.9035 | 4311 | "P" |
| | | | | | -307.2018 | -297.8254 | 3969 | "B" |
| TP3 | -5.498888 | -5.498888 | 1571 | Eq.(6)+Eq.(10) | -5.5036 | -5.4256 | 886 | "P" |
| | | | | | -5.4756 | -5.4245 | 885 | "B" |
| TP4 | -82.8625 | -82.8652 | 2067 | Eq.(7)+Eq.(10) | -83.0960 | -82.5450 | 1597 | "P" |
| | | | | | -83.1574 | -81.9877 | 1347 | "B" |
| TP5 | -5.682180 | -5.665284 | 2332 | Eq.(7)+Eq.(10) | -5.6450 | -5.4857 | 1092 | "P" |
| | | | | | -5.6346 | -5.0523 | 1251 | "B" |

## 7. Conclusions and future work

This paper presents a new EM type algorithm for solving constrained global optimization problems. The equality and inequality constraints are easily handled by a self-adaptive penalty technique aiming at penalizing constraints that are more difficult to be satisfied. The penalty parameters do not depend on user supplied values, are computed from information gathered from the population and depend on the level of violation of each constraint and on a factor that relies on objective function values.
Computational tests carried out with a set of well-known global optimization problems show that the proposed self-adaptive penalty electromagnetism-like algorithm is able to effectively solve constrained problems till optimality. A comparison with a similar adaptive penalty framework is also included. The preliminary results reported in the paper seem to show that this approach is competitive with other algorithms, although some modifications are required to improve accuracy.

## 8. References

**[1]** H.J.C. Barbosa and A.C.C. Lemonge, An adaptive penalty method for genetic algorithms in constrained optimization problems, *Frontiers in Evolutionary Robotics*, H. Iba (ed.) 34 pp. 2008 (ISBN: 978-3-902613-19-6).

**[2]** S.I. Birbil, *Stochastic Global Optimization Techniques*, PhD thesis, (2002) North Carolina State University.

**[3]** S.I. Birbil and S.-C. Fang, An electromagnetism-like mechanism for global optimization, *Journal of Global Optimization*, 25, 263–282, 2003.

**[4]** C.A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, *Computers in Industry*, 41(2), 113–127, 2000.

**[5]** K. Deb, An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering*, 186, 311–338, 2000.

**[6]** R. Fourer, D.M. Gay, and B.W. Kernighan, A modeling language for mathematical programming, *Management Science*, 36(5), 519–554, 1990.

**[7]** A.-R. Hedar and M. Fukushima, Derivative-free filter simulated annealing method for constrained continuous global optimization, *Journal of Global Optimization*, 35, 521–549 (2006).

**[8]** J. Joines and C. Houck, On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, *Proceedings of the first IEEE Conference on Evolutionary Computation* D. Fogel (ed.), IEEE Press, 579–584, 1994.

**[9]** A.C.C. Lemonge and H.J.C. Barbosa, An adaptive penalty scheme for genetic algorithms in structural optimization, *Internacional Journal for Numerical Methods in Engineering*, 59(5), 703–736, 2004.

**[10]** J.-L. Liu and J.-H. Lin, Evolutionary computation of unconstrained and constrained problems using a novel momentum-type particle swarm optimization, *Engineering Optimization*, 39(3), 287– 305, 2007.

**[11]** Y.G. Petalas, K.E. Parsopoulos and M.N. Vrahatis, Memetic particle swarm optimization, *Annals of Operations Research*, 156, 99–127, 2007.

**[12]** A.M.A.C. Rocha and E.M.G.P. Fernandes, Feasibility and dominance rules in the electromagnetism-like algorithm for constrained global optimization problems, *Lecture Notes in Computer Science*, Computational Science and Its Applications (O. Gervasi et al. (eds.)), 5073, 768–783, 2008.

**[13]** A.M.A.C. Rocha and E.M.G.P. Fernandes, Implementation of the electromagnetism-like algorithm with a constraint-handling technique for engineering optimization problems, *2008 Eighth International Conference on Hybrid Intelligent Systems*, ISBN: 978-0-7695-3326-1, IEEE Computer Society, 690-695, 2008.

**[14]** T.P. Runarsson and X. Yao, Stochastic ranking for constrained evolutionary optimization, *IEEE Transactions on Evolutionary Computation*, 4(3), 284–294, 2000.

**[15]** Y. Shi and R.C. Eberhart, Empirical study of particle swarm optimization, *Proceedings of the 1999 Congress on Evolutionary Computation*, 1945–1950, 1999.

**[16]** A.E.M. Zavala, A.H. Aguirre, and E.R.V. Diharce, Constrained optimization via particle evolutionary swarm optimization algorithm (PESO). *GECCO'05*, 209–216, 2005.