Optimization, 2014
Vol. 63, No. 3, 403–419, http://dx.doi.org/10.1080/02331934.2012.655691

A computational study on different penalty approaches for solving constrained global optimization problems with the electromagnetism-like method

M.M. Ali[a†], Mohsen Golalikhani[b*†] and Jun Zhuang[b†]

[a]*School of Computational and Applied Mathematics, Witwatersrand University, Wits – 2050, Johannesburg, South Africa;* [b]*Department of Industrial and Systems Engineering, University at Buffalo, The State University of New York, Buffalo, NY 14260, USA*

(*Received 13 January 2009; final version received 2 January 2012*)

In this article, the application of the electromagnetism-like method (EM) for solving constrained optimization problems is investigated. A number of penalty functions have been tested with EM in this investigation, and their merits and demerits have been discussed. We have also provided motivations for such an investigation. Finally, we have compared EM with two recent global optimization algorithms from the literature. We have shown that EM is a suitable alternative to these methods and that it has a role to play in solving constrained global optimization problems.

Keywords: electromagnetism-like method; constrained optimization problems; penalty functions

1. Introduction

Constrained optimization (CO) problems can be mathematically formulated as the following: given a real objective function f defined on a feasible set $\Omega \subset \mathbb{R}^n$, find a point $x^* \in \Omega$ and the corresponding value f^* such that

$$f^* = f(x^*) = \min\{f(x) | \forall x \in \Omega\}. \tag{1}$$

The set Ω is bounded by the box X and $p+q$ constraints, i.e.

$$\Omega = \{x \in \mathbb{R}^n | x \in X \text{ and } g_i(x) \leq 0 \text{ for } i = 1, 2, \ldots p;\ h_i(x) = 0,\ i = 1, 2, \ldots q\}. \tag{2}$$

We define the following problem

$$(P) \quad \begin{cases} \min & f(x), \\ \text{such that} & g_i(x) \leq 0, i = 1, 2, \ldots, m,\ m = p + 2q, \\ & x \in X, \end{cases}$$

*Corresponding author. Email: mohsengo@buffalo.edu
†The authors' names are in the alphabetical order.

where all equality constraints, $h_i(x)$, have been converted into inequality constraints using

$$|h_i(x)| - \epsilon \leq 0. \tag{3}$$

This means that the point x will be treated as a feasible point for the i-th inequality constraint if (3) holds. The value $\epsilon = 10^{-3}$ and 10^{-2} were used for the numerical experiment. We therefore redefine the feasible set, $\hat{\Omega}$, as

$$\hat{\Omega} = X \cap \left\{ x \in \mathbb{R}^n \middle| \, g_i(x) \leq 0, \; i = 1, 2, \ldots, m \right\}. \tag{4}$$

Furthermore, we assume the full dimensionality of $\hat{\Omega}$, i.e. its Lebesgue measure $m(\hat{\Omega}) > 0$.

In principle, the CO problem (P) can be solved using deterministic or stochastic methods. However, deterministic methods, such as feasible direction and generalized gradient descent, usually make strong assumptions on the continuity and differentiability of the objective function $f(x)$ [11,12,17]. Thus, there is an ongoing interest for stochastic algorithms that can tackle more general CO problems.

The most common approach to solve CO problems by using a stochastic algorithm is based on penalty functions [18,19,26]. In this approach, penalty terms are added to the objective function to penalize the objective function value of any infeasible solution that violates the constraints in (4). Thus, the penalty techniques transform a constrained problem into an unconstrained problem by penalizing the objective function when constraints are violated, and then minimize this penalized objective function using methods for unconstrained problems. The penalty approach has been applied to various stochastic algorithms, such as particle swarm optimization (PSO) [7,21], differential evolution (DE) [15] and genetic algorithm [8,9,10,13,28]. Besides penalty function approaches, the multi-objective approach has been applied to solve CO problems. We have also applied this approach to the electromagnetism-like method (EM) [1]. The multi-objective approach gives rise to an additional parameter which we found highly sensitive (see Ali and Golalikhani [1]). Indeed, Runarsson and Yao [24] demonstrated that the penalty function approach is better than the multi-criteria approach in constraint handling.

The EM [5] has been tested using a simple static penalty approach only for solving problem (P) [1,4,22]. In this penalty approach a penalty term with its parameter is added to the objective function $f(x)$. The author [4] used an interval for the parameter without significant investigations. Results reported in the above studies are less than satisfactory. It is therefore imperative to test EM using other penalty approaches and compare the results with those of other global optimization methods. This is what we aim to achieve with this article. A further motivation of this study is given in Section 3. We use four different types of penalty functions with EM for solving problem (P). Furthermore, we compare the computational results obtained through different types of penalty functions, and we explain their merits and demerits. We also compare our results with those results reported in the literature in order to show the performance of EM.

The rest of this article is organized as follows: Section 2 introduces the EM method and presents statements of it convergence. Section 3 presents various penalty functions and the numerical results of EM obtained with them. Section 4 provides sensitivity analysis for some important parameters of the penalty functions when

they are used in EM. Section 5 compares the performance of EM with the PSO [2] and DE algorithm [21]. Finally, Section 6 provides the conclusions of this article.

2. The EM

Initially designed for bound CO problems, EM [4] utilizes N, n-dimensional points $x_{i,k}$, $i = 1, 2, \ldots, N$, as a population to search the feasible set

$$X = \left\{x \in R^n \middle| l_i \leq x \leq u_i,\ i = 1, 2, \ldots, n\right\}.$$

The index k denotes the iteration (or generation) number of the algorithm. The initial population,

$$S_k = \left\{x_{1,k}, x_{2,k}, \ldots, x_{N,k}\right\}, \tag{5}$$

where $k = 1$, is taken to be uniformly distributed in the search region, X. We denote the population set at the k-th iteration by S_k as the members of the set S_k change with k. After the initialization of S_k, EM continues its iterative process until a stopping condition (e.g. maximum number of iterations) is met. An iteration of EM consists of two steps. In the first step, each point in S_k moves to a different location by using the attraction–repulsion mechanism of the electromagnetism theory. In the second step, points moved by the electromagnetism theory are further moved locally by a local search which then become the members of S_{k+1} in the $(k+1)$-th iteration. Both the attraction–repulsion mechanism and the local search in EM are responsible for driving the members, $x_{i,k}$, of S_k to the close proximity of the global minimizer.

As with the electromagnetism theory for charged particles, each point $x_{i,k} \in S_k$ in the search space X is assumed as a charged particle where the charge of a point relates to its the objective function value. Points with better objective function value have more charges than other points, and the attraction–repulsion mechanism is a process in EM by which points with more charge attract other points in S_k, and points with less charge repulse other points. Finally, a total force vector, F_i^k, exerted to a point, e.g. the i-th point $x_{i,k}$, is calculated by adding these attraction–repulsion forces and each $x_{i,k} \in S_k$ is moved to the direction of its total force to the location $y_{i,k}$. A local search is used to explore the vicinity of each $y_{i,k}$ by shifting $y_{i,k}$ to $z_{i,k}$. The members, $x_{i,k+1} \in S_{k+1}$, of the $(k+1)$-th iteration are then found by:

$$x_{i,k+1} = \begin{cases} y_{i,k} & \text{if } f(y_{i,k}) \leq f(z_{i,k}), \\ z_{i,k} & \text{otherwise.} \end{cases} \tag{6}$$

Algorithm 1 shows the general scheme of EM. We also provide the description of each step following the algorithm.

Algorithm 1 EM(N, MAXITER, LSITER, δ)

1. Input parameters: Input the maximum number of iteration *MAXITER*, the values for the local search parameters such as *LSITER* and δ, and the size N of the population.
2. Initialize: Set the iteration counter k=1, initialize the members of S_k uniformly in X, and identify the best point in S_k.
3. while $k < MAXITER$ do

4. $F_i^k \leftarrow \text{CalcF}(S_k)$
5. $\text{Move}(x_{i,k}, F_i^k)$
6. $\text{Local}(LSITER, \delta, y_{i,k})$
7. $\text{Select}(S_{k+1}, y_{i,k}, z_{i,k})$
8. $k = k + 1$
9. end while

Input parameter values (Line 1): the EM algorithm is run for $MAXITER$ iterations. In the local search phase, $n \times LSITER$ is the maximum number of locations $z_{i,k}$, within δ distance of $y_{i,k}$, for each i.

Initialize (Line 2): The points $x_{i,k}$, k=1, are selected uniformly in X, i.e. $x_{i,1} \sim Unif(X)$, $i = 1, 2, \ldots, N$, where $Unif$ represents the uniform distribution. The objective function values $f(x_{i,k})$, $i = 1, 2, \ldots, N$, are computed, and the best point

$$x_k^b = \arg \min_{x_{ik} \in S_k} \{f(x_{i,k})\} \tag{7}$$

is identified.

Calculate force (Line 4): In this step, a charged-like value ($q_{i,k}$) is assigned to each point ($x_{i,k}$). The charge $q_{i,k}$ of $x_{i,k}$ is dependent on $f(x_{i,k})$, and points with better objective function have more charge than others. The charges are computed as follows:

$$q_{i,k} = \exp\left(-n \frac{f(x_{i,k}) - f(x_k^b)}{\sum_{j=1}^{N} (f(x_{j,k}) - f(x_k^b))}\right). \tag{8}$$

Then, the force, $F_{i,j}^k$, between two points, $x_{i,k}$ and $x_{j,k}$, is calculated by

$$F_{i,j}^k = \begin{cases} (x_{j,k} - x_{i,k}) \dfrac{q_{i,k} q_{j,k}}{\|x_{j,k} - x_{i,k}\|^2} & \text{if } f(x_{i,k}) > f(x_{j,k}), \\ (x_{i,k} - x_{j,k}) \dfrac{q_{i,k} q_{j,k}}{\|x_{j,k} - x_{i,k}\|^2} & \text{if } f(x_{i,k}) \le f(x_{j,k}). \end{cases} \tag{9}$$

The total force, F_i^k, corresponding to $x_{i,k}$ is now calculated as

$$F_i^k = \sum_{j=1, j \neq i}^{N} F_{i,j}^k. \tag{10}$$

In the revised EM proposed in [6] there were some modifications in this step of the algorithm to preclude the premature convergence. In particular, Birbil et al. [6] selected one of the points in the population other than the current best point as the 'perturbed point', and they modified the method for calculating the force that was exerted on the perturbed point. After applying these modifications they proved that the new revised algorithm exhibits global convergence with probability one. Following [6] we implement these modifications in our numerical study. In this modification a perturbed point, say x_k^p, is selected to be the farthest point from the current best, x_k^b. The component force vector, calculated via (9), $F_{i,p}^k$ corresponding to x_k^p is modified as $\lambda F_{i,p}^k$, $\lambda \sim Unif(0, 1)$. In addition, the direction of the total force vector F_k^p, corresponding to x_k^p, is reversed with a small probability, say $\nu = 0.1$.

Move point $x_{i,k}$ along F_i^k (Line 5): In this step, each point $x_{i,k}$, except for x_k^b, are moved along the total force vector F_i^k using:

$$x_{i,k} = x_{i,k} + \lambda \frac{F_i^k}{\|F_i^k\|}(RNG), \quad i = 1, 2, \ldots, N; i \neq b, \tag{11}$$

where $\lambda \sim Unif(0, 1)$ for each coordinate of $x_{i,k}$, and RNG denotes the allowed range of movement towards the lower or upper bound for the corresponding dimension.

Local search (Line 6): For each $y_{i,k}$ a maximum of $LSITER$ points are generated in each coordinate direction in the δ neighbourhood of $y_{i,k}$. This means that the process of generating local points is continued for each $y_{i,k}$ until either a better $z_{i,k}$ is found or $n \times LSITER$ trial is reached.

Selection for the next iteration (Line 7): In this step, members $x_{i,k+1} \in S_{k+1}$ are selected from $y_{i,k}$ and $z_{i,k}$ using (6), and the best point is identified using (7).

3. Penalty functions and their numerical assessment

The penalized objective function $F(x)$ to be minimized is defined as:

$$F(x) = f(x) + \phi(x), \tag{12}$$

where $\phi(x)$ is the penalty term. The success of an underlying global optimization algorithm depends on the choice of $\phi(x)$. This is due to the fact that a choice of $\phi(x)$ and parameters therein may results in under penalization (may produce solution outside $\hat{\Omega}$) and over penalization (optimal solution in the boundary, $\partial\hat{\Omega}$, of $\hat{\Omega}$ may never be found). Various functional forms of $\phi(x)$ have been studied with a number of population-based search methods, see [15,20]. The algorithm we study here fundamentally differs from the ones studied in [15,20]. For example, EM generates trial points using force vectors while the other population-based methods [15,20] generates trial points by ordinary vector operations [16,27], by geometry-based operations, e.g. using simplex geometry [3], and by model-based operations [14]. In the model-based approach a model of the objective function is optimized. It is therefore necessary to see how EM reacts to various penalty functions. First, we have studied the effect of a switching strategy suggested in [14] using static penalty with four different parameter values. However, the results obtained in each case were inferior to those obtained using simple static penalty [15,29]. Hence, these results are not reported here. Next, we report on the performances of EM on various penalty functions. We consider below four penalty functions for this study [15,18,19,29]. These penalty functions, among other functions, are easy to implement and thus would be preferred by many practitioners.

Death penalty: This simple approach rejects infeasible solutions from the population by setting $\phi(x)$ to be $+\infty$ for any infeasible x. In this case, there will never be any infeasible solutions in the population. Our numerical experiments have shown that EM and other methods work generally well with this penalty if $\hat{\Omega}$ or a reasonable part of it is convex. A similar finding is also reported in [18,29].

Static penalty: In this approach, the penalty function is defined as

$$\phi(x) = d\sum_{i=1}^{m}[\max\{0, g_i(x)\}], \tag{13}$$

where d is the penalty parameter which does not depend on the current generation (iteration) of the algorithm that employs the function. For other penalty functions, we denote the penalty parameter by $d(k)$ as it depends on the iteration counter k of the underlying algorithm. For the test problems considered in our numerical study, we have observed that EM performs poorly with under penalization i.e. when d is small.

Dynamic and adaptive penalties: The general functional form of these two penalty functions is as follows:

$$\phi(x, k) = d(k)H(x), \tag{14}$$

where

$$H(x) = \sum_{i=1}^{m} \theta(\xi_i(x))\xi_i(x)^{\gamma(\xi_i(x))}, \tag{15}$$

and $d(k)$ is a continuous function of the iteration counter of the algorithm. The function $\xi_i(x)$ is given by $\xi_i(x) = \max\{0, g_i(x)\}$, $\theta(\xi_i(x))$ is the assignment function and $\gamma(\xi_i(x))$ is the power of penalty function. The functional forms of $\theta(\xi_i(x))$ and $\gamma(\xi_i(x))$ for both the penalties are presented in the following section.

Although both penalties have the same functional form, $\phi(x, k)$, what makes them different is the way they choose the function $d(k)$. For example, the dynamic penalty uses continuous $d(k)$ such as

$$d(k) = k^{0.1}, \tag{16}$$

while the adaptive penalty implements the following (updating) strategy [27]:

$$d(k+1) = \begin{cases} \beta_1 d(k) & \text{if Case 1,} \\ \beta_2 d(k) & \text{if Case 2,} \\ d(k) & \text{otherwise,} \end{cases} \tag{17}$$

where $\beta_2 > \beta_1$. The Case 1 is when all of the best individuals in the last k generations are feasible, and the Case 2 is when they are not feasible. In other words, if all best individuals of last k generations are feasible, penalty term $d(k+1)$ decreases. If all of them are infeasible, penalty term is increased. Otherwise, if the best individuals in the last k generations consist of both feasible and infeasible solutions, the penalty term does not change.

3.1. *A computational study of penalty functions with EM*

To evaluate the performance of EM with various penalty methods, we have used 16 test problems. The first 13 problems, denoted by G1–G13, are from [23]. The last three problems are denoted by P_1, P_2 and P_3, respectively. The problems P_1 and P_2

are taken from [15], while the problem P_3 is collected from [25]. The details of these three problems are given in the appendix. Here, we have conducted our numerical experiments of EM with the penalty functions and compared the results due to various penalty functions.

For each test problem, we have run EM, with each penalty function, 20 times to get average results. We have recorded the best achieved result, the average of 20 results, the sum of constraints violations and the number of best-achieved results with violated constraints. A constraint $g_i(x)$ (including the converted constraints in (3)) is assumed to be violated if and only if

$$g_i(x) > \hat{\varepsilon}, \tag{18}$$

where $\hat{\varepsilon} = 2.00\text{E}{-5}$. Algorithms are coded in Visual Basic, the size, N, of population is set to be 10 and EM algorithm stops after 350,000 objective function evaluations (fevals) for all test problems except for the low-dimensional problems G8, G11 and G12. For these problems, we use 100,000 fevals to stop EM. For the second set of problems, P_1, P_2 and P_3, EM stops after 300,000 function evaluations.

A simple local search suggested in [5] has been implemented in Algorithm 1 of the EM. The local search iteration, see LSITER in Algorithm 1, has been set to be 30. We have also used the local search parameter $\delta = 0.1$ for G1, G2, G4, G7, G8, G10, P_1 and P_2. For the other test problems we have used the local search parameter to be $\delta = 0.01$.

3.2. *EM with death penalty*

Results of EM with death penalty are presented in Table 1, where Prob. represents problem, n denotes problem dimension, T denotes average number of trial points to generate initial N feasible solutions, $f(x_k^b)$ denotes the optimal function value obtained after the execution of EM, $mf(x_k^b)$ denotes the average of $f(x_k^b)$, $\#x_k^b \notin \hat{\Omega}$

Table 1. Results for the death penalty approach.

Prob.	n	$f(x^*)$	T	$f(x_k^b)$	$mf(x_k^b)$	$\#x_k^b \notin \hat{\Omega}$	CV
G1	13	−15	4,806,354	−14.9899	−14.3998	0	0
G2	20	−0.803619	10	−0.738893	−0.689201	0	0
G3	5	−1	12,933	−1.0023	−1.0008	0	0
G4	5	−30665.539	34	−30654.882	−30619.700	0	0
G5	4	5126.4981	–				
G6	2	−6961.81388	136,892	−6961.0406	−6960.7572	0	0
G7	10	24.3062091	10,852,815	27.56396	30.73161	0	0
G8	2	−0.095825	1246	−0.095825	−0.095825	0	0
G9	7	680.630057	1849	680.9928	682.9098	0	0
G10	8	7049.3307	1,762,744	7126.1215	7254.9930	0	0
G11	2	0.75	10,348	0.749	0.749	0	0
G12	3	−1	221	−1	−1	0	0
G13	5	0.0539498	–				
P_1	6	−310.0	92	−309.78	−303.79	0	0
P_2	10	−47.7648	–				
P_3	2	−5.5080	22	−5.5079	−5.5072	0	0

denotes the number of infeasible optima out of 20 runs and CV is the amount of total constraint violation over 20 runs. In the death penalty approach the initial $N=10$ solutions have to be feasible. If the algorithm is unable to produce initial N feasible solutions after $1E+8$ random trials then this has been denoted by '−' in Table 1. In this case the algorithm is terminated immediately. One positive aspect of the death penalty approach is that it assures feasibility of results so that no infeasible solution is generated during the execution of EM. Another advantage is that the death penalty does not have any parameter and is easy to implement. However, in some problems (e.g. G1, G7 and G10) this penalty approach needs tedious computations for generating the first random population of the problem. It makes this penalty approach not suitable for problems in which the constraints are difficult to be satisfied.

3.3. *EM with static penalty*

Results of this implementation are summarized in Table 2, where the values of the parameter, d, are also given. We have performed numerical experiments to identify the appropriate values for d, and these values are larger than those used in [4]. Table 2 shows that in spite of larger d, in many test problems (i.e. G3, G4, G5, G6 and G10) there are still constraint violations. We could use even larger values for d to avoid infeasible solutions for these problems. However, in that case the static penalty approach will be similar to the death penalty approach. A comparison between Tables 1 and 2 shows that the static penalty returns better optimal values than the death penalty.

3.4. *EM with dynamic penalty*

We have studied a number of functional forms for the functions $\theta(\cdot)$, $\gamma(\cdot)$ and $d(k)$. In particular, we have found that the functional form of $\theta(\cdot)$ is highly sensitive

Table 2. Results for the static penalty approach.

Prob.	n	$f(x^*)$	d	$f(x_k^b)$	$mf(x_k^b)$	$\#x_k^b \notin \hat{\Omega}$	CV
G1	13	−15	1.00E+05	−14.9920	−14.0660	1	1.70E−05
G2	20	−0.803619	1.00E+05	−0.739588	−0.691339	0	0
G3	5	−1	1.00E+05	−1.0022	−1.0009	0	0
G4	5	−30665.539	1.00E+06	−30659.105	−30641.834	8	2.66E−03
G5	4	5126.4981	1.00E+06	5126.382	5266.908	3	3.19E−02
G6	2	−6961.81388	1.00E+06	−6961.7663	−6961.7387	6	1.23E−04
G7	10	24.3062091	1.00E+06	26.4876	29.6904	1	2.10E−05
G8	2	−0.095825	1.00E+05	−0.095825	−0.095825	0	0
G9	7	680.630057	1.00E+05	680.7103	682.4962	0	0
G10	8	7049.3307	1.00E+07	7089.2883	7240.2423	3	7.32E−03
G11	2	0.75	1.00E+06	0.749	0.749	0	0
G12	3	−1	1.00E+06	−1	−1	0	0
G13	5	0.0539498	1.00E+06	0.122379	2.059540	1	5.30E−05
P_1	6	−310.0	1.00E+06	−310.0	−309.91	9	1.49E−03
P_2	10	−47.7648	1.00E+05	−46.8835	−45.1767	11	1.11E−03
P_3	2	−5.5080	1.00E+05	−5.5080	−5.5069	0	0

to problem. We have numerically tested $\theta(\cdot)$ using various forms such as linear, quadratic and other non-linear form such as $\theta(x) = \sqrt{x}$. Results obtained are inferior in all cases even for $\gamma(x) = \text{constant}$. However, introduction of an additional parameter, α, in the following functional form of $\theta(x)$ has produced much superior results which we report in this section. Our numerical investigations have suggested that the following scheme is a suitable choice. We have used

$$\gamma(\xi_i(x)) = \begin{cases} 1 & \text{if } \xi_i(x) < 1, \\ 2 & \text{otherwise,} \end{cases} \tag{19}$$

$$\theta(\xi_i(x)) = \begin{cases} \alpha & \text{if } \xi_i(x) < 10^{-5}, \\ 10\alpha & \text{if } 10^{-5} \le \xi_i(x) < 10^{-3}, \\ 100\alpha & \text{if } 10^{-3} \le \xi_i(x) < 1, \\ 1000\alpha & \text{if } \xi_i(x) > 1, \end{cases} \tag{20}$$

and $d(k) = k^{0.1}$. The values of α are given in Table 3 where we have summarized the results. Results obtained by the dynamic penalty approach are very encouraging as this penalty approach produces the best results among different penalty functions for a majority of the test problems.

3.5. *EM with adaptive penalty*

The implementation of this penalty requires the initialization of $d(1)$ which we have presented in Table 4 along with the summarized results. We have used $\beta_1 = 0.95$ and $\beta_2 = 1.1$ in (17). In Table 4, † denotes the case where more than 16 infeasible final solutions were found out of 20 runs. Table 4 shows that the results are quite encouraging although the algorithm returns infeasible results for some problems (i.e. G5, G6 and G10). One advantage of this kind of penalty function is that in this

Table 3. Results for the dynamic penalty approach.

Prob.	n	$f(x^*)$	α	$f(x_k^b)$	$mf(x_k^b)$	$\#x_k^b \notin \hat{\Omega}$	CV
G1	13	−15	100	−14.9974	−14.4074	0	0
G2	20	−0.803619	1000	−0.748726	−0.704819	0	0
G3	5	−1	100	−1.0024	−1.0014	0	0
G4	5	−30665.539	100	−30654.668	−30637.301	0	0
G5	4	5126.4981	2000	5126.765	5258.158	7	1.88E−02
G6	2	−6961.81388	1000	−6961.1305	−6960.800	0	0
G7	10	24.3062091	1000	25.5581	29.4959	0	0
G8	2	−0.095825	100	−0.095825	−0.095825	0	0
G9	7	680.630057	100	680.8541	682.5100	0	0
G10	8	7049.3307	100	7099.7691	7195.4998	2	9.60E−04
G11	2	0.75	100	0.749	0.749	0	0
G12	3	−1	100	−1	−1	0	0
G13	5	0.0539498	1000	0.058757	1.88808	0	0
P_1	6	−310.0	1000	−310.0	−309.25	0	0
P_2	10	−47.7648	100	−46.9232	−45.1583	0	0
P_3	2	−5.5080	100	−5.5080	−5.5068	0	0

Table 4. Results for the adaptive penalty approach.

Prob.	n	$f(x^*)$	$d(1)$	$f(x_k^b)$	$mf(x_k^b)$	$\#x_k^b \notin \hat{\Omega}$	CV
G1	13	−15	100	−14.9949	−14.2491	0	0
G2	20	−0.803619	1000	−0.748726	−0.712199	0	0
G3	5	−1	100	−1.0022	−1.0014	0	0
G4	5	−30665.539	100	−30663.835	−30639.754	10	1.72E−03
G5	4	5126.4981	2000	†			
G6	2	−6961.81388	1000	†			
G7	10	24.3062091	1000	26.6337	29.2485	0	0
G8	2	−0.095825	100	−0.095825	−0.095825	0	0
G9	7	680.630057	100	681.0428	382.3096	0	0
G10	8	7049.3307	100	†			
G11	2	0.75	100	0.749	0.749	0	0
G12	3	−1	100	−1	−1	0	0
G13	5	0.0539498	1000	0.11782	2.1318	0	0
P_1	6	−310.0	1000	−310.0	−309.78	11	4.61E−03
P_2	10	−47.7648	100	−45.1583	−0.712199	1	2.70E−05
P_3	2	−5.5080	100	−5.5080	−5.5076	0	0

approach the starting penalty parameter $d(1)$ is insensitive and the final value of $d(k)$ which usually converges to a suitable value. We have shown this fact in following section.

4. Sensitivity analysis of parameter values

In this section, we have provided a sensitivity analysis for some of the important parameters of the penalty functions investigated. In particular, the penalty parameter d is used in static and adaptive penalties, and the parameter p in $d(k) = k^p$ is used in dynamic penalty. The death penalty does not have any parameter and thus we have excluded this from the analysis in this section. We begin with the parameter d of the static penalty function and show its effects on the average optimal value and the total constraint violation in Figure 1. The average values and the total constraint violations are based on 10 independent runs on both G3 and G10. As expected, Figure 1 shows that the objective value improves with the increase of d. Moreover, it can be seen that the feasibility increases with the increase of d.

Next, we study the initial value of the parameter $d(k)$, namely $d(1)$ in the adaptive penalty. We use four functions, namely G3, G4, G5 and G8. Our study has shown that the final value of $d(k)$ is largely independent of the initial choice of $d(1)$. For example, the final value $d(k)$ for G4 diverges to ∞ (respectively converges to zero for G3) regardless of the initial choice of $d(1)$. In Figure 2, we have shown this trend for all four functions. Results presented in Figure 2 are averages on 10 independent runs. Clearly, the final $d(k)$ is problem dependent and not on the initial choice of $d(1)$. This characteristic can be considered as an advantage of adaptive penalty function.

Finally, we study the parameter p in $d(k) = k^p$ for the dynamic penalty. We use the functions G1 and G10 for this study. The values for the other parameter, α, are fixed to the values presented in Table 3. Again the results are based on 10 independent runs. The function $d(k)$ is a monotonically increasing continuous

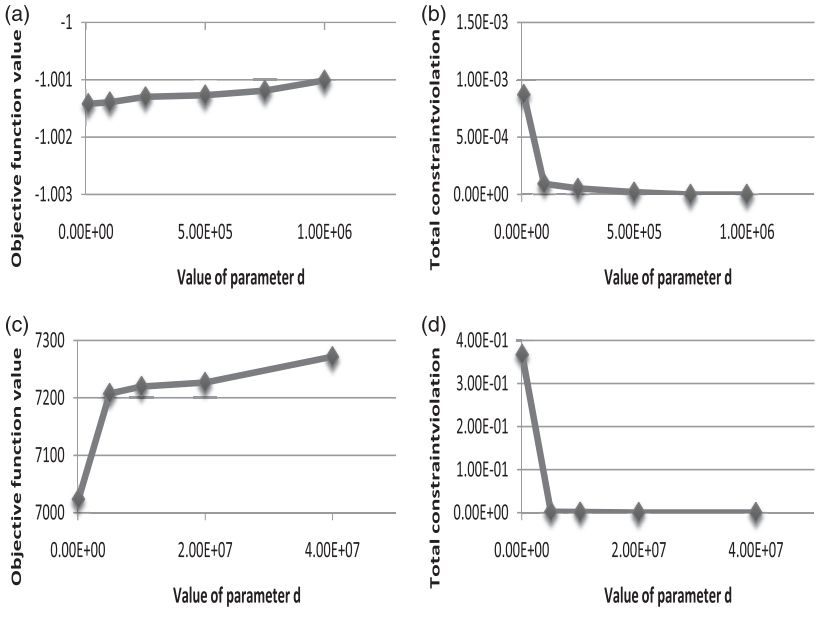

Figure 1. Effects of d in static penalty: (a) optimal objective value for problem G3, (b) total constraint violation for problem G3, (c) optimal objective value for problem G10 and (d) total constraint violation for problem G10.

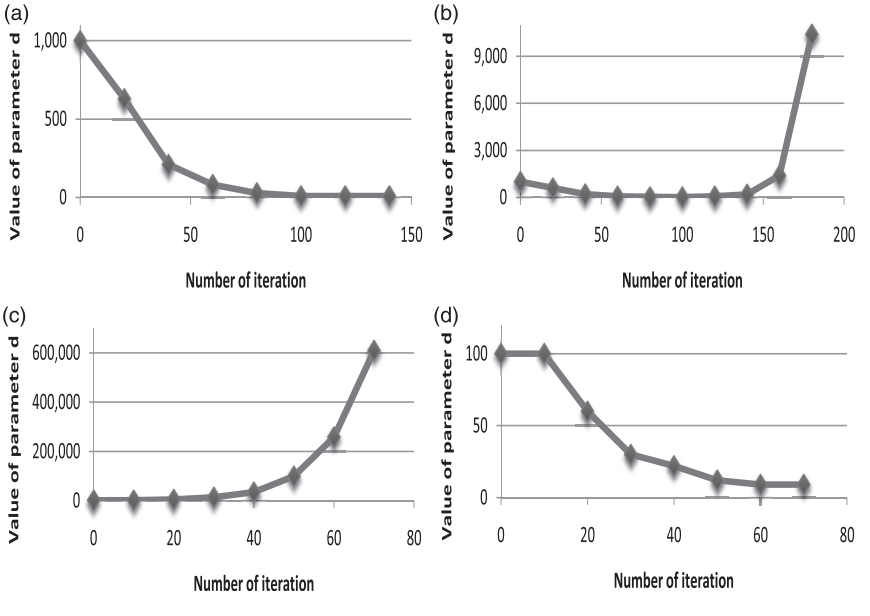

Figure 2. Effects of initial $d(1)$ in adaptive penalty: (a) test problem G3, (b) test problem G4, (c) test problem G5 and (d) test problem G8.

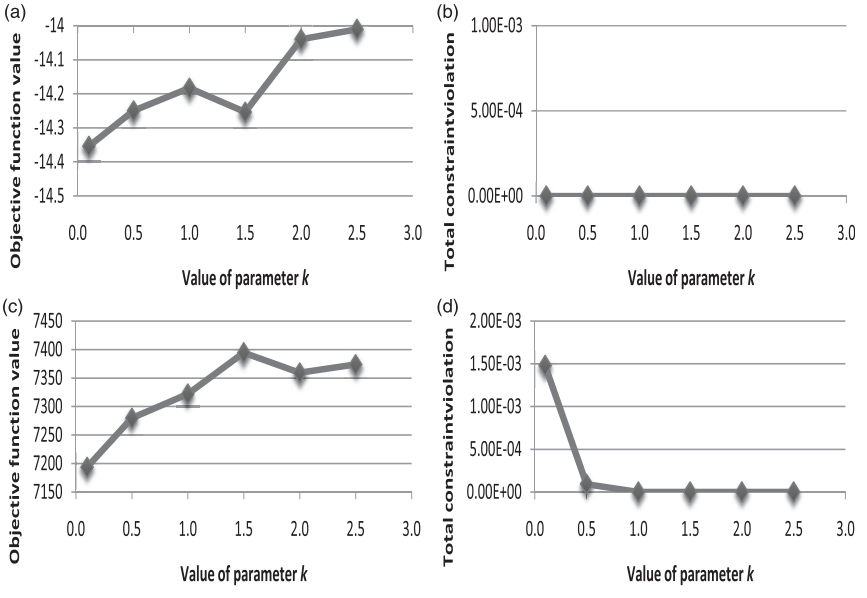

Figure 3. Effects of the parameter p of the continuous $d(k)$: (a) optimal objective value for problem G1, (b) total constraint violation for problem G1, (c) optimal objective value for problem G10 and (d) total constraint violation for problem G10.

function of the iteration counter k. Figure 3 shows that the average of optimal solution increases (improves), and also the feasibility increases with the increase of the iteration counter, as expected.

Therefore, the most critical factor for setting penalty parameters is finding suitable values which are not so small (for a small value the results may be infeasible), and also not so big (for a large value the results may have large deviation from optimal objective function values). We have chosen the parameter values for the parameters d, p and α based on the above observation. Results reported in this article are based on suitable choices for these parameters, but their small variations also produced similar results.

5. Comparison of EM with PSO and DE

In this section, we compare the performance of EM with the PSO [21] and a recent version of DE [2]. To do this, we use the static penalty function (12) and (13) in EM. We have chosen this penalty function due to the fact that it has only one parameter and therefore in our view the comparison will be less biased by the parameter value. Indeed, for a number of problems the comparison would favour EM more if we have used the results of EM based on the dynamic penalty function, but we have avoided this since the dynamic penalty has two parameters. We have used the same stopping condition for all algorithms, Section 3.1. Table 5 compares the results obtained by these methods. Table 5 shows that some average $mf(x_k^b)$ values are better than the known global minimizer, see G3. This is due to the use of $\hat{\varepsilon}$ tolerance in (18).

Table 5. Comparison between EM, PSO and DE.

	EM			PSO			DE		
Prob.	$mf(x_k^b)$	$\#x_k^b \notin \hat{\Omega}$	CV	$mf(x_k^b)$	$\#x_k^b \notin \hat{\Omega}$	CV	$mf(x_k^b)$	$\#x_k^b \notin \hat{\Omega}$	CV
G1	−14.0660	1	1.70E−05	−12.6823	0	0	−14.8761	0	0
G2	−0.691339	0	0	−0.584064	0	0	−0.7604	0	0
G3	−1.0009	0	0	−1.0025	0	0	−1.0001	0	0
G4	−30641.83	8	2.66E−03	−30665.71	20	1.21E−02	−30665.4993	4	3.02E−04
G5	5266.908	3	3.19E−02	5429.288	3	3.40E−05	5217.5601	5	1.40E−3
G6	−6961.738	6	1.23E−04	−6955.472	5	1.44E−03	−6961.7632	5	6.22E−6
G7	29.6904	1	2.10E−05	25.0439	1	1.80E−05	24.5765	2	1.22E−5
G8	−0.095825	0	0	−0.095825	0	0	−0.095825	0	0
G9	682.4962	0	0	680.6628	0	0	682.5501	0	0
G10	7240.2423	3	7.32E−03	7464.0071	5	3.33E−03	7188.7286	4	5.63E−03
G11	0.749	0	0	0.749	0	0	0.749	0	0
G12	−1	0	0	−1	0	0	−1	0	0
G13	2.0595	1	5.30E−05	0.748896	1	1.1E−05	0.278	0	0
P_1	−309.91	9	1.49E−03	−298.80	20	1.43E−03	309.665	6	6.44E−5
P_2	−45.1767	11	1.11E−03	−41.3184	14	2.78E−03	−46.76	7	1.37E−3
P_3	−5.5069	0	0	−5.5080	0	0	−5.5080	0	0

Table 6. CPU times s.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	P_1	P_2	P_3
EM	3	10	2	2	2	1	4	<1	3	3	<1	39	2	2	2	<1
PSO	6	16	3	4	4	2	7	<1	5	5	<1	32	4	5	6	2
DE	4	11	6	4	8	2	6	2	4	3	<1	24	9	3	6	3

Note that the penalty parameter d used in EM is in accordance with Table 2. We have used the parameters of PSO [21] and DE [2] as suggested in the respective references. The summarized results in Table 5 are taken from 20 independent runs on each problem. Table 5 shows that PSO, on average, is the worst performer amongst all three algorithms. For example, PSO has produced only the best $mf(x_k^b)$ for G9 and P_3. It has also produced 69 infeasible final solutions, i.e. $\#x_k^b \notin \hat{\Omega}$, which is the highest number of the three.

A comparison between EM and DE shows that DE is superior to EM with respective to $\#x_k^b \notin \hat{\Omega}$, scoring 33 infeasible final solutions against 43 for EM. In terms of the average results, $mf(x_k^b)$, EM and DE are comparable except for problem G13. The success rate of DE on G13 is 39%. The DE algorithm often failed on this problems due to the fact that it got stuck to the corresponding local minimizer. The local minimum value found for G13 was 0.4388. It is important to note that EM with dynamic penalty has produced a better success rate than DE on G13. The success rate of EM on G13 is 44%. Although Table 3 shows a higher $mf(x_k^b)$ for EM than that of DE in Table 5, it is due to the fact that a few runs of EM ended at higher function values than 0.4388. During our numerical study we observed that all algorithms located two different global minimizers of G11.

Finally, we compare the average computational times for this study using Table 6 which clearly shows that in most problems EM is the best followed by PSO. DE is the worse performer with respect to the computational time except for G12. The comparison clear shows that EM has a role to play in the field of global optimization.

6. Conclusion

We have used different kinds of penalty functions in order to solve CO problems with the EM. We have demonstrated that the convergence property of EM holds for solving constrained problems.

We have studied the performance of the EM with a number of penalty functions and have shown the good performance of the method using the static and dynamic penalties. Finally, we have compared the performance of the EM using the static penalty function with two recent algorithms, namely a PSO and a DE algorithm. With this comparison, we have established that the EM can be used as a suitable alternative to many existing methods for solving constrained global optimization problems.

Acknowledgements

We thank the anonymous associate editor and two anonymous referees for their helpful comments. We also thank Ms Elizabeth Newell from the University at Buffalo for her editorial comments which improved the presentation of this article.

References

[1] M.M. Ali and M. Golalikhani, *An electromagnetism-like method for non-linearly constrained global optimization*, Comput. Math. Appl. 60 (2010), pp. 2279–2285.
[2] M.M. Ali and Z. Kajee-Bagdadi, *Local exploration based differential evolution algorithm for constrained global optimization*, Appl. Math. Comput. 208 (2009), pp. 31–48.
[3] M.M. Ali and A. Törn, *Population set-based global optimization algorithms: some modifications and numerical studies*, Comput. Oper. Res. 31 (2004), pp. 1703–1725.
[4] S.I. Birbil, *Stochastic global optimization techniques*, Ph.D. thesis, North Carolina State University, Raleigh, NC, 2002.
[5] S.I. Birbil and S.-C. Fang, *An electromagnetism-like mechanism for global optimization*, J. Global Optim. 25 (2002), pp. 263–282.
[6] S.I. Birbil, S.-C. Fang, and R.-L. Sheu, *On the convergence of a population-based global optimization algorithm*, J. Global Optim. 30 (2005), pp. 301–318.
[7] G. Coath and S.K. Halgamuge, *A comparison of constraint-handling methods for the application of particle swarm optimization to constrained non-linear optimization problems*, Proceedings of the 2003 IEEE Conference on Evolutionary Computation, Vol. 4, Canberra, Australia, 2003, pp. 2419–2725.
[8] D.W. Coit, A.E. Smith, and D.M. Tate, *Adaptive penalty methods for genetic optimization of constrained combinatorial problems*, INFORMS J. Comput. 8 (1996), pp. 173–182.
[9] M. Gen and R. Cheng, *A survey of penalty techniques in genetic algorithms*, Proceedings of the 1996 International Conference on Evolutionary Computation, Nagoya, Japan, 1996, pp. 804–809.
[10] A.B. Hadj-Alouane and J.C. Bean, *A Genetic algorithm for the multiple-choice integer program*, Oper. Res. 45 (1997), pp. 92–101.
[11] D.M. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.
[12] R. Horst and P.M. Pardalos, *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1995.
[13] J. Joines and C. Houck, *On the use of non-stationary penalty functions to solve non-linear constrained optimization problems with gas*, Proceedings of the first IEEE Conference on Evolutionary Computation, Orlando, Florida, 1994, pp. 579–584.
[14] P. Kaelo and M.M. Ali, *Some variants of the controlled random search algorithm for global optimization*, J. Optim. Theory and Appl. 130 (2006), pp. 253–264.
[15] Z. Kajee-Bagdadi, *Differential evolution algorithms for constrained global optimization*, M.Sc. thesis, School of Computational and Applied Mathematics, University of the Witwatersrand, South Africa, 2007.
[16] H. Lu and W. Chen, *Self adaptive velocity particle swarm optimization for constrained optimization problems*, J. Global Optim. 41 (2008), pp. 427–445.
[17] D.G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison Wesley, Boston, MA, 1973.
[18] Z. Michalewicz, D. Dasgupta, R. Le Riche, and M. Schoenauer, *Evolutionary algorithms for constrained engineering problems*, Comput. Ind. Eng. 30 (1996), pp. 851–870.
[19] Z. Michalewicz and M. Schoenauer, *Evolutionary algorithms for constrained parameter optimization problems*, Evol. Comput. 4 (1996), pp. 1–32.

[20] K. Miettinen, M. Makela, and J. Toivanen, *Numerical comparison of some penalty based constraint handling techniques in genetic algorithm*, J. Global Optim. 27 (2003), pp. 427–446.
[21] K.E. Parsopoulos and M.N. Vrahatis, *Particle swarm optimization method for constrained optimization problems*, Front. Artif. Intell. Appl. Ser. 76 (2002), pp. 214–220.
[22] A.M.A.C. Rocha and E.M.G.P. Fernandes, *Feasibility and Dominance Rules in the Electromagnetism-like Algorithm for Constrained Global Optimization*, Lecture Notes in Computer Science, Vol. 5071, Springer-Verlag, Berlin-Heidelberg, 2008, pp. 768–783.
[23] T.P. Runarsson and X. Yao, *Stochastic ranking for constrained evolutionary optimization*, IEEE Trans. Evol. Comput. 4 (2000), pp. 284–294.
[24] T.P. Runarsson and X. Yao, *Search biases in constrained evolutionary optimization*, IEEE Trans. Syst. Man Cyber. Part C 35 (2005), pp. 223–243.
[25] W.F. Sacco, *Private communication*, Faculdade de Matematica, Universidade Federal do Oeste do Para, Brazil, 2011.
[26] A.E. Smith and D.W. Coit, *Constraint Handling Techniques – Penalty Functions*, *Handbook of Evolutionary Computation*, Oxford University Press and Institute of Physics Publishing, Oxford, 1997.
[27] R. Storn and K. Price, *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces*, J. Global Optim. 11 (1997), pp. 341–359.
[28] J.M. Yang, Y.P. Chen, J.T. Horng, and C.Y. Kao, *Applying Family Competition to Evolution Strategies for Constrained Optimization*, Lecture Notes in Computer Science, Vol. 1213, Springer-Verlag, Berlin-Heidelderg-New York, 1997, pp. 201–211.
[29] O. Yeniay, *Penalty function methods for constrained optimization with genetic algorithms*, Math. Comput. Appl. 10 (2005), pp. 45–56.

Appendix

P_1: $\min f(x) = -25(x_1-2)^2-(x_2-2)^2-(x_3-1)^2-(x_4-4)^2-(x_5-1)^2-(x_6-4)^2$, subject to

$$\begin{aligned}(x_3-3)^2+x_4 &\geq 4,\\ (x_5-3)^2+x_6 &\geq 4,\\ x_1-3x_2 &\leq 2,\\ -x_1+x_2 &\leq 2,\\ x_1+x_2 &\leq 6,\\ x_1+x_2 &\geq 2,\\ 0\leq x_1 &\leq 6,\\ 0\leq x_2 &\leq 6,\\ 1\leq x_3 &\leq 5,\\ 0\leq x_4 &\leq 6,\\ 1\leq x_5 &\leq 5,\\ 0\leq x_6 &\leq 10.\end{aligned}$$

Optimal solution: $f(x^*) = -310$, $x^* = (5, 1, 5, 0, 5, 10)$.

P_2: $\min f(x) = \sum_{i=1}^{10} x_j\left(c_i + ln\frac{x_j}{x_1+\cdots+x_{10}}\right)$, subject to

$$h_1(x) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0,$$
$$h_2(x) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0,$$
$$h_3(x) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0,$$
$$0 \le x_i \le 10,\ i = 1, \ldots, 10.$$

$$c = (-6.089, -17.164, -34.054, -5.914, -24.721, -14.986, -24.1, -10.708, -26.662, -22.179)$$

Optimal solution: $f(x^*) = -47.764888$,

$$x^* = (0.04066, 0.1477212, 0.7832057, 0.001414339, 0.48529363, 0.000693183, 0.074052, 0.017950966, 0.0373268186, 0.09688446).$$

$$P_3: \quad \min f(x) = -x_1 - x_2, \quad \text{subject to}$$

$$g_1(x) = x_2 - 2{x_1}^4 + 8{x_1}^3 - 8{x_1}^2 - 2 \le 0,$$
$$g_2(x) = x_2 - 4{x_1}^4 + 32{x_1}^3 - 88{x_1}^2 + 96x_1 - 36 \le 0,$$
$$0 \le x_1 \le 3,$$
$$0 \le x_2 \le 4.$$

Optimal solution: $f(x^*)$=-5.5080, $x^* = (2.32952, 3.17849)$.