Optimization Methods and Software
Vol. 00, No. 00, January 2008, 1–17

RESEARCH ARTICLE

Modified movement force vector in a electromagnetism-like mechanism for global optimization

Ana Maria A. C. Rocha* and Edite M. G. P. Fernandes

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

(*v1.2 released September 2008*)

This paper presents an algorithm for solving global optimization problems with bounded variables. The algorithm is a modification of the electromagnetism-like mechanism proposed by Birbil and Fang [J. of Global Optimization 25 (2003), pp. 263-282]. The differences are mainly on the local search procedure and on the force vector used to move each point in the population. Several widely used benchmark problems were solved in a performance evaluation of the new algorithm when compared with the original one. A comparison with other stochastic methods is also included. The algorithm seems appropriate for large dimension problems.

Keywords: Global optimization; Electromagnetism-like algorithm; Pattern search method; Performance profile

AMS Subject Classification: 90C30; 90C56

1. Introduction

In this paper, we consider the problem of finding a global solution of a nonlinear optimization problem with box constraints in the following form:

$$\begin{array}{l} \text{minimize } f(x) \\ \text{subject to } x \in \Omega, \end{array} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function and $\Omega = \{x \in \mathbb{R}^n : -\infty < l_k \leq x_k \leq u_k < \infty, k = 1, \ldots, n\}$ is a bounded feasible region. We assume that the objective function f is not convex and may possess many local minima in the set Ω. This class of global optimization problems is very important and frequently encountered in engineering applications.

In the last decades, many algorithms have been proposed to solve problem (1). Probably the most extensively used in practice for moderate and large-dimensional problems are stochastic-type algorithms. For low-dimensional problems, the deterministic procedures may have the advantage of guaranteed convergence. The stochastic methods can be classified in two main categories, namely, the point-to-point search strategies and the population-based search techniques. From the population-based techniques, we would like to emphasize two particular algorithms, the electromagnetism-like mechanism [3] and the particle swarm optimization

*Corresponding author. Email: arocha@dps.uminho.pt

ISSN: 1055-6788 print/ISSN 1029-4937 online

DOI: 10.1080/1055678xxxxxxxxxxxxx
http://www.informaworld.com

(PSO) [12], since they are easy to implement and computationally inexpensive in terms of memory requirement.

In this paper, we are interested in the electromagnetism-like (EM) algorithm proposed in [3]. This algorithm 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 original algorithm also incorporates a simple random local search procedure that is applied coordinate by coordinate to a selected point in the population [2, 3]. Each point is moved according to a vector that reflects the objective function value of the corresponding point relative to the other points in the population, therein denoted by the total force exerted on that particular point. Without imposing any type of smoothness on $f(x)$, asymptotic convergence is proved for a modified version of the original EM [4].

In order to improve its search ability and efficiency and to extend to larger dimension size problems, two modifications are introduced in the original EM algorithm. First, to gather information about a particular point, the random local search is replaced by a pattern search method [13, 17] with guaranteed convergence. Previous work, related with this issue [15], tested this combination, herein denoted as EM - PS algorithm, on eighteen selected problems and compared with the original EM algorithm. The results were promising. Pursuing this research, we now propose a new and simple idea to define a new vector to move each point in the population. In particular, a linear combination of the total force exerted on a point, computed at the current iteration, with that of the previous iteration is used to define the vector to move the corresponding point in the population.

The proposed algorithm is herein extensively experimented on a well-known benchmark problems set, with a total of 64 problems. Performance profile plots, as outline in [6], have been made to assess the average and best behavior of the new algorithm when compared with both EM - PS and original EM algorithms. Experiments on problems with various dimension sizes, ranging from 2 to 100, and a variety of inherent difficulties [18] have been carried out to demonstrate the efficiency of the proposed algorithm in solving large dimension problems. Comparisons with other stochastic algorithms are also included.

The paper is organized as follows. In Section 2 we describe the three EM algorithms. The original EM is briefly introduced in Subsection 2.1, the main ideas concerning the local pattern search method are described in Subsection 2.2, and the proposed modification to the movement force vector is then presented in Subsection 2.3. Section 3 contains the results of all the numerical experiments and we conclude the paper in Section 4.

2. Electromagnetism-like algorithms

In this section, we briefly describe the original EM algorithm and the local pattern search method, and discuss the modified movement force vector.

2.1. *Original EM algorithm*

The EM algorithm starts with a population of randomly generated points from the feasible region. Analogous to electromagnetism, each point is a charged particle that is released to the space. The charge of each point is related to the objective function value and determines the magnitude of attraction of the point over the population. The better the objective function value, the higher the magnitude of attraction.

The charges are used to find a direction for each point to move in subsequent iterations. The regions that have higher attraction will signal other points to move towards them. In addition, a repulsion mechanism is also introduced to explore new regions for even better solutions.

The following notation is used: $x^i \in \mathbb{R}^n$ denotes the ith point of a population; x^{best} is the point that has the least objective function value; $x^i_k \in \mathbb{R}$ is the kth $(k = 1, \ldots, n)$ coordinate of the point x^i of the population; m is the number of points in the population; nf_{max} is the maximum number of function evaluations allowed; nit_{max} is the maximum number of iterations allowed; $lsit_{max}$ denotes the maximum number of local search iterations; and δ is a local search parameter, $\delta \in [0, 1]$.

The EM algorithm contains four main procedures: *Initialize*, *CalcF*, *Move* and *Local*. The general scheme is as follows.

Algorithm 1 (original EM algorithm)
Input: m, nf_{max} or nit_{max}, δ, $lsit_{max}$
  Initialize()
  iteration $\leftarrow$ 1
  while termination criterion is not satisfied **do**
    $F \leftarrow$ *CalcF*()
    Move(F)
    Local - random line search(f, δ, $lsit_{max}$)
    iteration $\leftarrow$ iteration + 1
  end while

The Algorithm *1* is terminated after nf_{max} function evaluations or after nit_{max} iterations. Another criterion based on the relative error of the best found solution falling below some tolerance has also been adopted [3, 9, 15]. Below, details of each procedure are presented.

Initialize is a procedure that aims to randomly generate m points from the feasible region. Each coordinate of a point (x^i_k) $(k = 1, \ldots, n)$ is assumed to be uniformly distributed between the corresponding upper and lower bounds, i.e., $x^i_k = l_k + \lambda(u_k - l_k)$ where $\lambda \sim U(0, 1)$. After computing the objective function value for all the points in the population, the procedure identifies the best point, x^{best}, which is the point with the best function value.

The *CalcF* procedure aims to compute the total force exerted on a point *via* other points. According to the electromagnetism theory this force is inversely proportional to the square of the distance between the points and directly proportional to the product of their charges. The charges of the points are computed according to their objective function values.

For each point x^i, the charge q^i determines the power of attraction or repulsion for that point. In [2, 3] the charge of a point is computed as

$$q^i = \exp\left(-n\frac{f(x^i) - f(x^{best})}{\sum_{k=1}^{m}(f(x^k) - f(x^{best}))}\right), \quad i = 1, \ldots, m. \tag{2}$$

In this way the points that have better objective function values possess higher charges. A different approach to evaluate the charges is adopted in [5, 11].

The total force vector F^i exerted on each point x^i is then calculated by adding the individual component forces, F^i_j, between any pair of points x^i and x^j. As the charges (2) are all positive, the direction of a force F^i_j depends on the objective function values at x^i and x^j. Thus, if $f(x^j) < f(x^i)$ (x^j attracts x^i) the direction of the force should be $\overrightarrow{x^i x^j}$, whereas if $f(x^j) \geq f(x^i)$ (x^j repels x^i) the direction of

the force is $\overrightarrow{x^j x^i}$. Thus,

$$F_j^i = \begin{cases} (x^j - x^i)\frac{q^i q^j}{\|x^j - x^i\|^2} \text{ if } f(x^j) < f(x^i) \text{ (attraction)} \\ (x^i - x^j)\frac{q^i q^j}{\|x^j - x^i\|^2} \text{ if } f(x^j) \geq f(x^i) \text{ (repulsion)} \end{cases},$$

and the total force exerted on point x^i of the population is evaluated as

$$F^i = \sum_{j \neq i}^{m} F_j^i, \quad i = 1, 2, \ldots, m. \tag{3}$$

The *Move* procedure uses the total force vector, F^i, to move the point x^i in the direction of the force by a random step length λ. The best point, x^{best}, is not moved and is carried out to the subsequent iteration. To maintain feasibility, the force exerted on each point is normalized and scaled by the allowed range of movement towards the lower bound l_k, or the upper bound u_k, of the set Ω, for each coordinate k. Thus, for $i = 1, 2, \ldots, m$ and $i \neq best$

$$x_k^i = \begin{cases} x_k^i + \lambda \frac{F_k^i}{\|F^i\|}(u_k - x_k^i) \text{ if } F_k^i > 0 \\ x_k^i + \lambda \frac{F_k^i}{\|F^i\|}(x_k^i - l_k) \text{ otherwise} \end{cases}, \quad k = 1, 2, \ldots, n. \tag{4}$$

The random step length λ is assumed to be uniformly distributed between 0 and 1.

Finally, The *Local* procedure performs a local refinement and can be applied to one point or to all points in the population. The local search presented in [3] is a random line search algorithm that is applied coordinate by coordinate to a point x^i in the population.

First, based on the parameter δ, the procedure computes the maximum feasible step length, $s_{max} = \delta \max_k(u_k - l_k)$. This quantity is used to guarantee that the local search generates feasible points. Second, the point x^i is assigned to a temporary point y to store the initial information. Next, for each coordinate k, a random number λ between 0 and 1 is selected as a step length and the point y_k is moved along that direction, $y_k = y_k + \lambda\, s_{max}$. If an improvement is observed, within $lsit_{max}$ iterations, the point x^i is replaced by y and the search along that coordinate k ends.

2.2. *Local pattern search method*

Here, we describe one of the modifications that were incorporated into the original EM algorithm so that better accuracy solutions and faster convergence are attained. Instead of a random line search, the Hooke and Jeeves pattern search algorithm is used to define the *Local* procedure [8, 13]. This is a derivative-free method that searches in the neighborhood of a point x^i for a better approximation using two types of moves: the exploratory move and the pattern move.

This algorithm is a variant of the well-known coordinate search method (a search along the coordinate axes). It incorporates a pattern move to accelerate the progress of the algorithm, by exploiting information obtained from the search in previous successful iterations. To reduce the number of function evaluations, this pattern search algorithm is applied to the current best point only. Thus, at each iteration the exploratory move carries out a coordinate search about the best point, with a step length δ. If a new trial point, y, with a better function value than x^{best} is encountered, the iteration is successful and δ is maintained. Otherwise, the iteration

is unsuccessful and δ should be reduced. If the previous iteration was successful, the vector $y - x^{best}$ defines a promising direction and a pattern move is then implemented, which means that the exploratory move is carried out about the trial point $y + (y - x^{best})$, rather than about the current point y. Then, if the coordinate search is successful, the returned point is accepted as the new point; otherwise, the pattern move is rejected and the method reduces to a coordinate search about y. We refer to [8] for details.

To ensure feasibility in this local pattern search procedure an exact penalty strategy is used. This means that the pattern search is applied to the problem

$$\min g(x) \equiv \begin{cases} f(x) & \text{if } x \in \Omega, \\ \infty & \text{otherwise} \end{cases}$$

rather than to (1). Thus, any trial point that is infeasible would be rejected, since the objective function value is ∞. The EM algorithm that incorporates this local pattern search procedure is presented below. In Algorithm *2*, ε_δ is the factor to reduce the step length δ and δ_{min} is the minimum step length allowed.

Algorithm 2 (EM pattern search algorithm)
Input: m, nf_{max} or nit_{max}, δ, δ_{min}, ε_δ
  Initialize()
  iteration $\leftarrow$ 1
  while termination criterion is not satisfied **do**
    $F \leftarrow$ *CalcF*()
    Move(F)
    Local - pattern search(g, δ, δ_{min}, ε_δ)
    iteration $\leftarrow$ iteration + 1
  end while

When compared with Algorithm *1*, the main difference is on the *Local* procedure. Recent numerical tests with this EM pattern search method on eighteen selected problems show that the computed solutions are in general better than those of the original EM algorithm although at function evaluation costs in some cases. The reader is referred to [15].

2.3. *Modified movement force vector*

The total force exerted on a point reflects the behavior of the objective function value of the corresponding point relative to the other points in the population. Thus, it is appropriate to move the point according to this force. However, past information on the relation between the function values at points in the population may be used to adjust the force during movement and accelerate the convergence of the algorithm.

This section proposes a new modified EM algorithm for solving global optimization problems with bounded variables using a linear combination of the total force exerted on a point, computed at the current iteration, with that of the previous iteration to define the force vector to move that point in the population.

This scheme aims to incorporate in the movement force vector past information of the force exerted on a particular point, i.e., the force used to move the point x^i, F^i, as presented in (4), is a linear combination of the force exerted on that point at the current iteration, $F^i_{\text{iteration}}$, with the total force of the previous iteration, $F^i_{\text{iteration}-1}$,

$$F^i = F^i_{\text{iteration}} + \beta F^i_{\text{iteration}-1} \tag{5}$$

Table 1. Number of problems ($Nprob$) by dimension

Dimension n	$Nprob$	Dimension n	$Nprob$
2	21	10	15
3	4	15	1
4	9	17	1
5	5	20	3
6	1	25	1
9	2	30	1

where β is a positive number termed the memory constant which adjusts the change in the movement force vector. The point can memorize the previous force and adjust the current force to move the point.

This scheme was incorporated into the algorithm of Subsection 2.2. Thus, unlike the previous algorithms that use the resultant force (3) of $m-1$ points to move x^i, this modified algorithm uses a combination of forces, as described in (5), in moving x^i from one iteration to another. Below, we present the corresponding algorithm.

Algorithm 3 (modified movement force EM pattern search algorithm)
Input: m, nf_{max} or nit_{max}, δ, δ_{min}, ε_δ, β
Initialize()
iteration $\leftarrow 1$
$F_{\text{iteration}-1} \leftarrow 0$
while termination criterion is not satisfied **do**
    $F_{\text{iteration}} \leftarrow$ *CalcF*()
    Move($F_{\text{iteration}} + \beta F_{\text{iteration}-1}$)
    Local - pattern search(g, δ, δ_{min}, ε_δ)
    iteration $\leftarrow$ iteration + 1
end while

3. Numerical experiments

In this section, we report the numerical results obtained by running the modified movement force EM pattern search algorithm on a set of global optimization problems with bounded variables. The computational tests were performed on a PC with a 3GHz Pentium IV microprocessor and 1Gb of memory. We compare the herein proposed algorithm, as outline in Algorithm *3*, with the original EM algorithm, described in Subsection 2.1, and the EM pattern search, described in Subsection 2.2.

We used a collection of 50 benchmark global optimization test problems, produced in full detail in the Appendix B of [1]. Ten of them were tested for two values of the dimension (n) and one (denoted by $NF3$ in [1]) was tested for five values of n giving a total of 64 problems. Table 1 lists the number of problems distributed by dimension.

We also include a comparison with three benchmark stochastic-type methods. Two of them are based on a population of points and the other is a well-known point-to-point search algorithm. Further tests including a comparison with two variants of the particle swarm optimization algorithm are shown later on in this section. To simplify the reader's task the following notations are used to describe the algorithms under comparison:

- *original EM*: original electromagnetism-like algorithm (Algorithm *1*)
- *EM - PS*: electromagnetism-like algorithm with the local pattern search procedure (Algorithm *2*)

- *modEM - PS*: modified movement force electromagnetism-like algorithm with the local pattern search procedure (Algorithm *3*)
- *CMA - ES*: a evolution strategy with a covariance matrix adaptation [7]
- *PSwarm*: particle swarm in a pattern search algorithm [19]
- *ASA*: adaptive simulated annealing method [10]
- *PSO(Shi+Eberhart)*: particle swarm optimization algorithm based on dynamic inertia weights [16]
- *momentumPSO*: momentum-type particle swarm optimization algorithm [14].

3.1. *Setting parameters*

The values for the constants are: $lsit_{max} = 10$, $\delta = 0.001$ (as it is used in [3]), $\delta_{min} = 1 \times 10^{-8}$ and $\varepsilon_\delta = 0.1$. The values chosen for the two latter constants are typical in pattern search algorithms. Since problem dimensions in the test set vary from 2 to 30, we decided to use the number of points in the population dependent on n. Thus, we set $m = \min\{200, 10n\}$.

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. Denote this parameter by $nruns$. For the remaining part of this paper, we denote the known optimal solution by f_{opt}, the average best function values, i.e., the average value of the best function values obtained over the $nruns$ by f_{avg}, and the best function value by $f_{best} = \min(f^i_{best}),\ i = 1, \ldots, nruns$.

To analyze the sensitivity of the parameter β to the final results, we used the mean absolute error (MAE)

$$MAE = \frac{|f_{opt} - f_{avg}|}{n}, \tag{6}$$

a scaled distance between the average performance and the optimal value [14], that aims to measure the accuracy of the solutions found by the algorithm, and tested three values of β: $0.1, 0.5$ and 0.9. Figure 1 is a 100% stacked column chart for the values of MAE obtained by the selected β values. For each problem, we can compare the percentage that each β value contributes to the total. Hence, the smaller the percentage the better. Overall, the area of the bars corresponding to $\beta = 0.1$ is smaller than the others - 29.7% in contrast with 32% (for $\beta = 0.5$) and 38.3% (for $\beta = 0.9$). This indicates that 0.1 is a better choice for β.

3.2. *Performance profiles*

To evaluate and compare the performance of the three electromagnetism-like algorithms, we use a performance profile initially proposed in [6]. The performance profiles give, for every $\tau \geq 1$, the proportion $\rho(\tau)$ of test problems on which each algorithm under comparison has a performance within a factor τ of the best. The performance profile plot represents the cumulative distribution function of a performance ratio based on an appropriate metric. Dolan and Moré in [6] proposed the use of the computing time required to solve a problem, but other metrics could be used. A brief discussion of our implementation of this performance assessment follows.

Let $\mathcal{P}$ be the set of all problems and $\mathcal{S}$ the set of solvers used in the comparative study. Let $m_{(p,s)}$ be the performance metric found by solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ after a fixed number of function evaluations. Here, a metric that measures

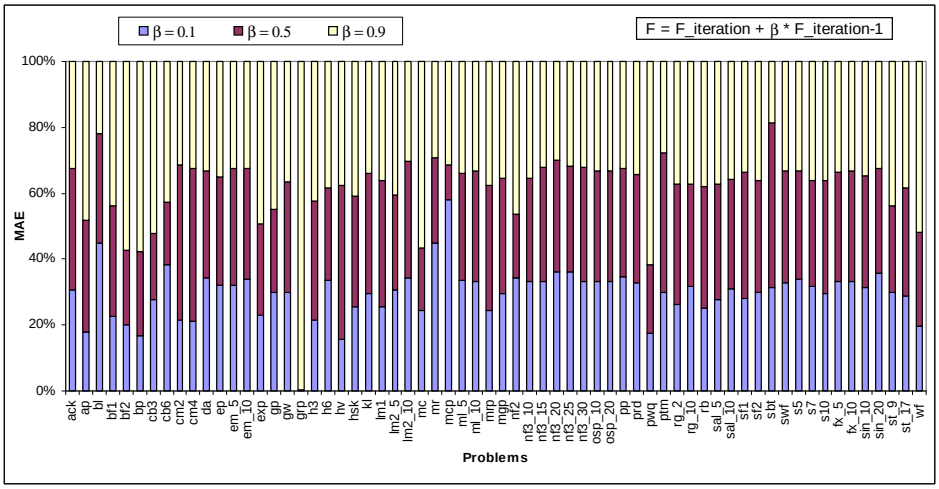

Figure 1. MAE comparison for three values of β.

the relative improvement of the function values, a scaled distance to the optimal function value f_{opt}, defined in [1] by

$$m_{(p,s)} = \frac{f_{avg\,(p,s)} - f_{opt}}{f_{worst} - f_{opt}} \tag{7}$$

is used, where f_{worst} denotes the worst function value found among all solvers on the problem p and $f_{avg\,(p,s)}$ is the average of the best function values found by solver s on problem p, after nf_{max} function evaluations over a certain number of runs. In this context we set $nruns = 30$. Metric (7) is to be used when the average assessment of the solvers is required.

If one is interested in the best assessment of the solvers then $f_{best\,(p,s)}$, the best function value found by solver s on problem p over 30 runs should be used in (7) instead of $f_{avg\,(p,s)}$.

As the $\min\{m_{(p,s)} : s \in \mathcal{S}\}$ can be zero for a particular problem, the performance ratios used in our comparative study are defined by

$$r_{(p,s)} = \begin{cases} 1 + m_{(p,s)} - \min\{m_{(p,s)} : s \in \mathcal{S}\}, \text{ if } \min\{m_{(p,s)} : s \in \mathcal{S}\} < \epsilon \\ \frac{m_{(p,s)}}{\min\{m_{(p,s)} : s \in \mathcal{S}\}}, \text{ otherwise} \end{cases},$$

for $p \in \mathcal{P}, s \in \mathcal{S}$ and $\epsilon = 0.00001$ (see [19] for a more complete discussion). Then, the overall assessment of the performance of a particular solver s is given by

$$\rho_s(\tau) = \frac{1}{n_P} \text{size}\{p \in \mathcal{P} : r_{(p,s)} \leq \tau\}$$

where n_P is the number of problems in the set $\mathcal{P}$. The "size" is the number of problems in the set such that the performance ratio $r_{(p,s)}$ is less than or equal to τ for solver s. $\rho_s(\tau)$ is the probability (for solver $s \in \mathcal{S}$) that the performance ratio $r_{(p,s)}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio. The function ρ_s is the cumulative distribution function for the performance ratio.

Using the performance profile plot one can compare how well a solver can estimate the optimum relative to the others.

The value of $\rho_s(1)$ gives the probability that the solver s will win over the others in the set. However, for large values of τ, the $\rho_s(\tau)$ measures the solver robustness.

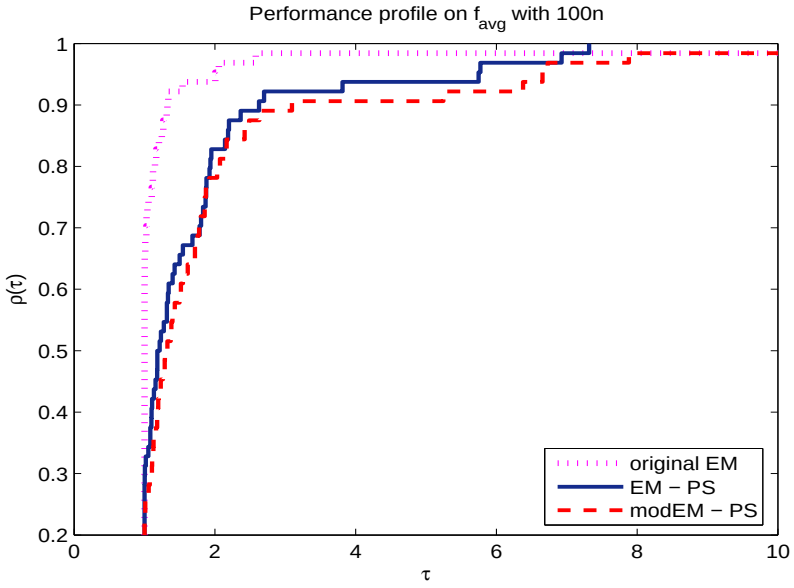

Figure 2. Performance profile on f_{avg} with $100n$ function evaluations.

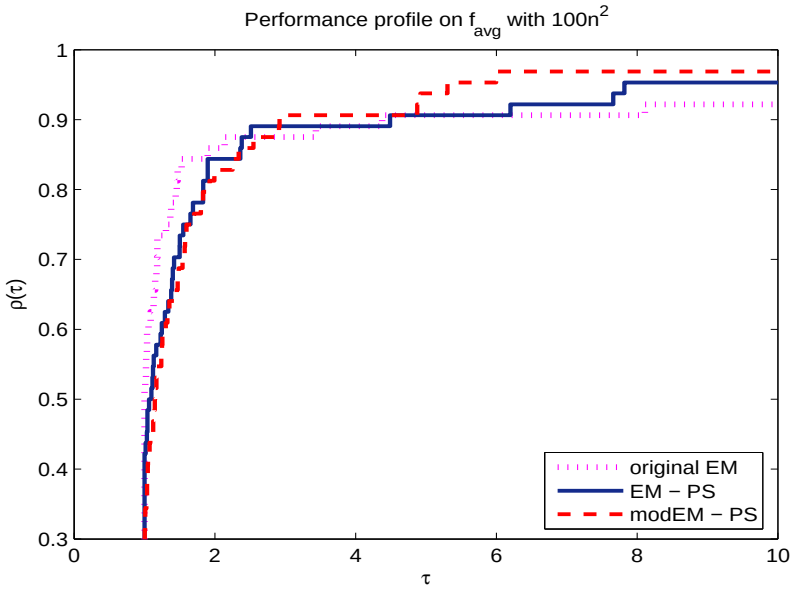

Figure 3. Performance profile on f_{avg} with $100n^2$ function evaluations.

The solver with largest $\rho_s(\tau)$ is the one that solves more problems in the set $\mathcal{P}$.

Two experiments were made with different maximum number of allowed function evaluations: $nf_{max} = 100n$ and $nf_{max} = 100n^2$. The focus here is to compare computational requirement and solution accuracy. The factors n and n^2 aim to show the effect of dimensionality on the algorithm performance, as higher dimension problems are in general more difficult to solve than lower dimension ones.

The performance profile plots for the average solutions found when $nf_{max} = 100n$ and $nf_{max} = 100n^2$ are presented in Figure 2 and Figure 3 respectively. From Figure 2 one may conclude that, as far as the average assessment is concerned, *original EM* dominates the other two over most of the τ values, although it loses in performance for values of τ greater than 7. The *EM - PS* algorithm wins against

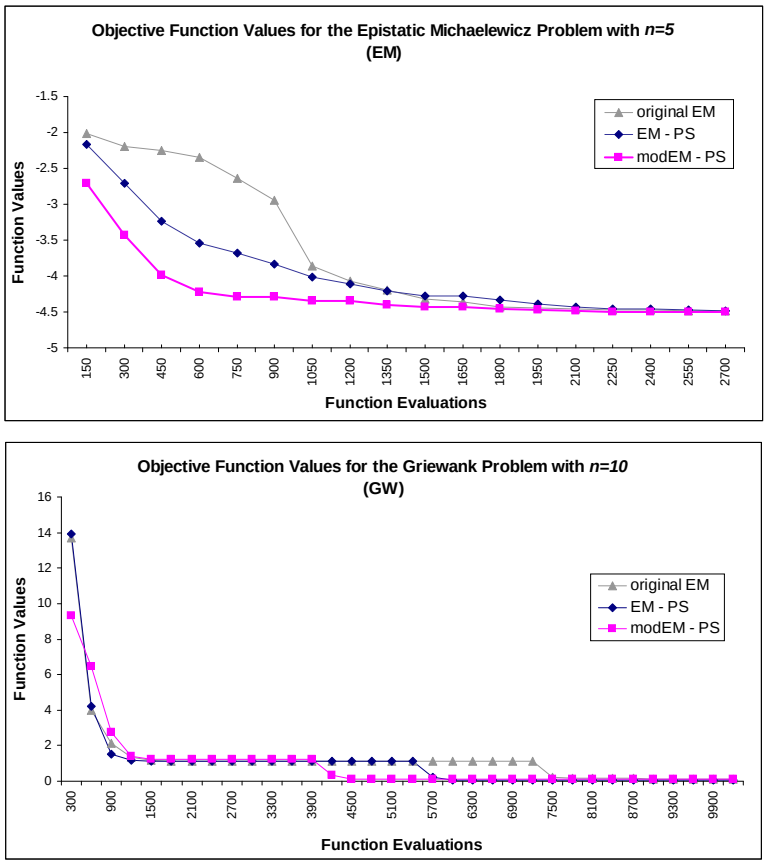

Figure 4. Objective function values for problems EM ($n = 5$) and GW ($n = 10$)

the others with respect to robustness. However, when a maximum of $100n^2$ function evaluations is allowed, and for τ greater than approximately 2.7, the *modEM - PS* algorithm wins against the others and dominates with respect to robustness (see Figure 3). At the beginning of the plot, *original EM* is best.

To further analyze the effect of nf_{max} on the algorithm performance, we plot the objective function values *versus* the number of function evaluations of four particular problems of the test set: EM, GW, $NF3$ and OSP (see [1]). Figures 4 and 5 show that as the number of function evaluations increases the best algorithm for the problem may change. For example, the top plot of Figure 4 (for problem EM with $n = 5$) shows that *modEM - PS* outperforms the other two for any number of function evaluations, whereas for problem GW ($n = 10$) - in the bottom plot - the *modEM - PS* is the best at the very beginning, then *original EM* and *EM - PS* algorithms perform slightly better than *modEM - PS* from the 300th until the 1200th function evaluation. At the 3800th function evaluation, *modEM - PS* starts to be the best algorithm at least until the 5500th function evaluation. The top plot of Figure 5 corresponds to the problem $NF3$ with $n = 10$ and illustrates the best performance of *modEM - PS* except between the 400th and the 1500th function evaluation. From the bottom plot (for problem OSP with $n = 10$) we can conclude that *modEM - PS* is the best algorithm from the 4200th function evaluation on. Until this point, *EM - PS* seems to be slightly better than the other two.

Finally, we also include the performance profile plots concerning the best solutions found. See Figure 6 when $nf_{max} = 100n$ and Figure 7 when $nf_{max} = 100n^2$. In the best assessment one may conclude that *original EM* should be implemented if only $100n$ function evaluations were allowed. However, if one is willing to al-

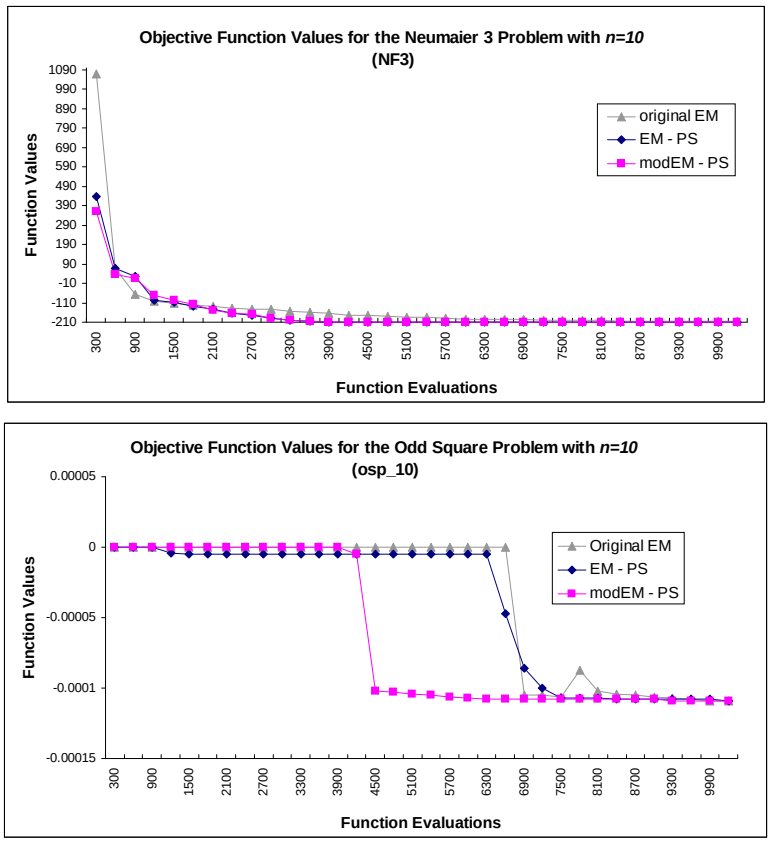

Figure 5. Objective function values for problems $NF3$ ($n = 10$) and OSP ($n = 10$).

low $100n^2$ function evaluations then *modEM - PS* should be preferred, although *original EM* is the best for $\tau \leq 1.5$.

3.3. *Comparison with other stochastic methods*

Three benchmark stochastic-type methods were selected to compare and assess the average and the best effectiveness of the new *modEM - PS* algorithm: (i) *CMA - ES*, a population-based evolution strategy with a covariance matrix adaptation [7]; (ii) *PSwarm*, a population-based particle swarm in a pattern search algorithm [19]; and (iii) *ASA*, a point-to-point search based on adaptive simulated annealing [10]. The tests were done considering 30 independent runs, and a population of $\min\{200, 10n\}$ points is used with *CMA - ES*, *PSwarm* and *modEM - PS*. Our comparison is based on the performance profiles, as described in Subsection 3.2. First we plot the performance profiles on the average assessment of the solvers. See Figure 8. Then, the performance assessment is based on the best objective function value, as illustrated in Figure 9. In both cases, the solvers were allowed to run for $10n$ iterations.

From Figure 8, we can conclude that *CMA - ES* wins over the other methods for all values of τ, and is closely followed by our *modEM - PS* algorithm. The solver *ASA* seems to have the worst performance. However, when the assessment is based on the best function value, the plots in Figure 9 illustrate that the herein proposed *modEM - PS* algorithm wins over the others.

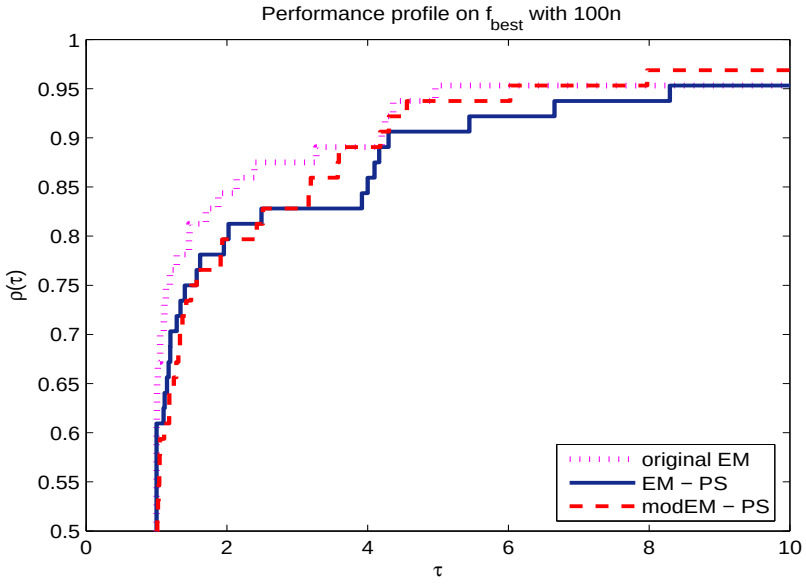

Figure 6. Performance profile on f_{best} with $100n$ function evaluations.

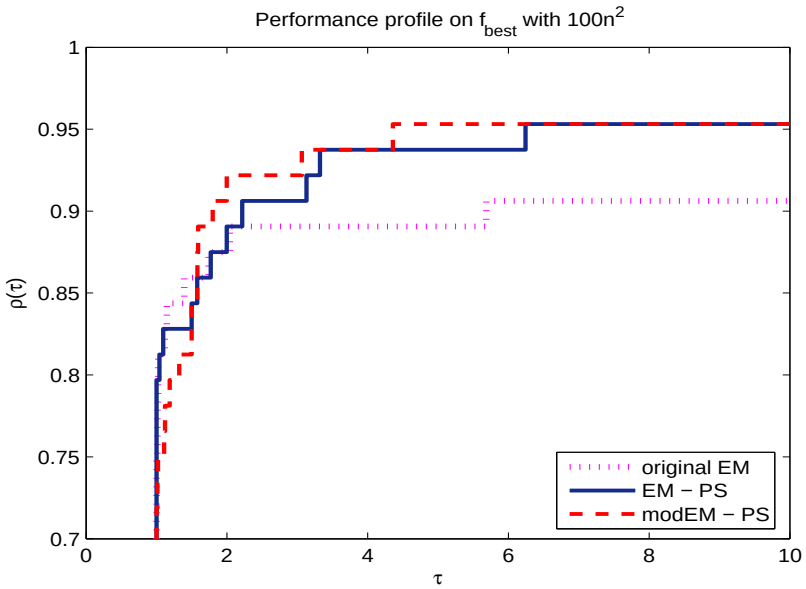

Figure 7. Performance profile on f_{best} with $100n^2$ function evaluations.

3.4. *Experiments with varied dimension problems*

To analyze the performance of the new proposed *modEM - PS* algorithm, when compared with *original EM* and *EM - PS* algorithms, as the problem dimensions size increases, two multi-modal functions with varied dimensions are used. The first function is not in the previously referred test set and is used in [14] with similar purposes,

$$\begin{array}{l} \text{maximize } f(x) \equiv -\sum_{i=1}^{n} \left(\sin(x_i) + \sin\left(\frac{2x_i}{3}\right)\right) \\ \text{subject to } x_i \in [3, 13], i = 1, \ldots, n. \end{array}$$

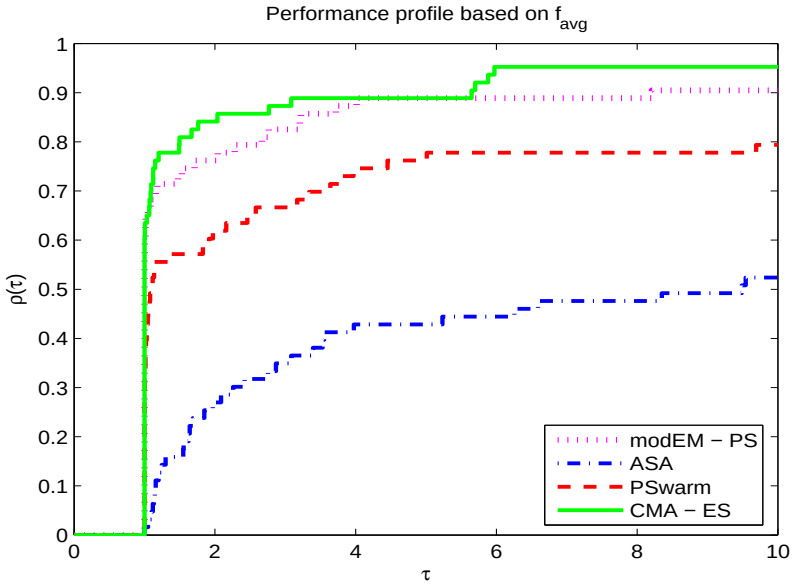

Figure 8. Performance profile based on f_{avg} for $10n$ iterations.

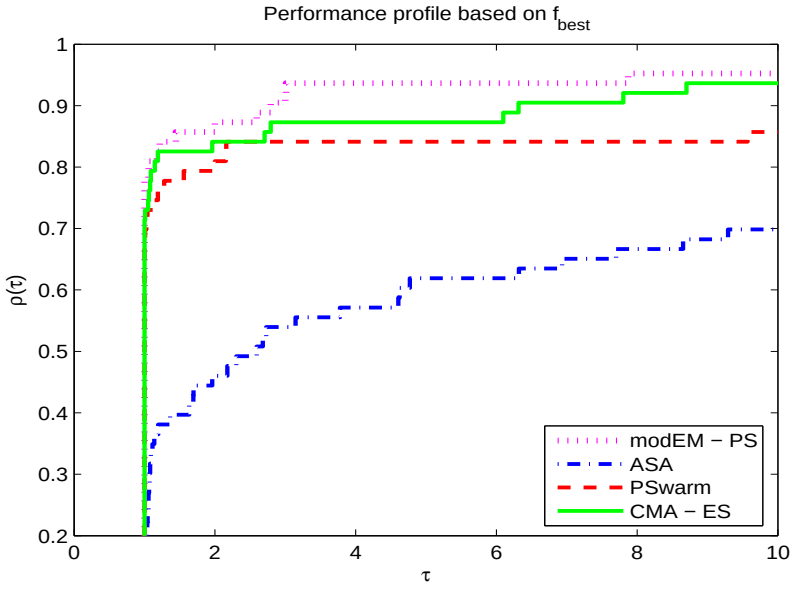

Figure 9. Performance profile based on f_{best} for $10n$ iterations.

We consider five values of n (10, 25, 50, 75, 100). The analytical solution is given by $1.216n$. We include Figures 10 and 11 to show the results for $n = 50$ and $n = 100$ of the three electromagnetism-like algorithms under comparison. Here, we analyze the average of the best objective function values, f_{avg}, over 20 runs, with a population of 50 points. The termination criterion is based on the number of allowed iterations nit_{max}, that is set to 5000.

The proposed *modEM - PS* presents the best convergence rate and numerical accuracy. The best solutions found by *modEM - PS* were 60.799109 and 121.598218 for $n = 50$ and $n = 100$ respectively, whereas the other two algorithms did not reach solutions as good as these, in less that 5000 iterations (see also Table 2). Good results were also obtained by the momentum-type particle swarm optimiza-

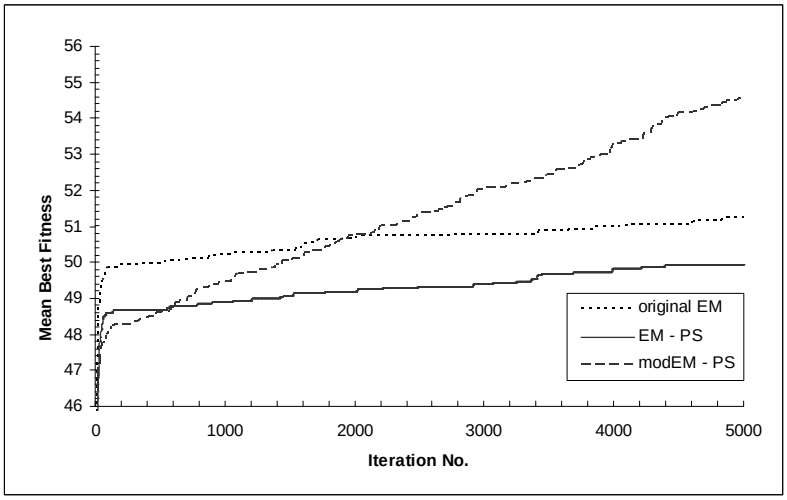

Figure 10. Convergence of *original EM*, *EM - PS* and *modEM - PS* algorithms on a problem with $n = 50$.

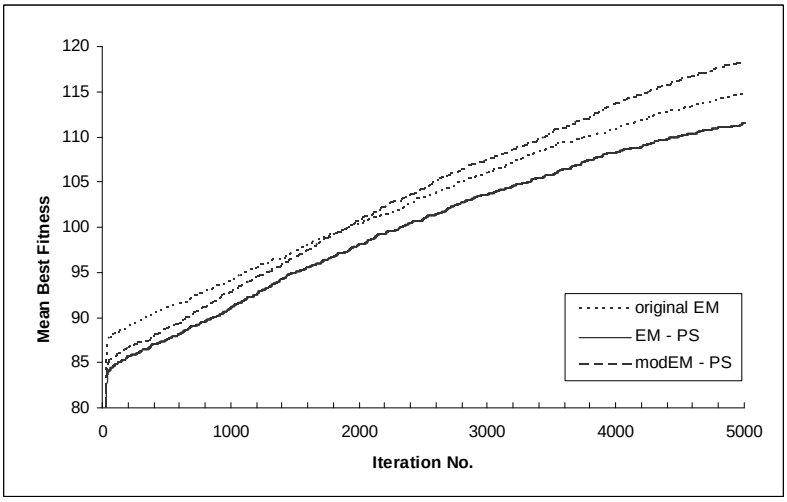

Figure 11. Convergence of *original EM*, *EM - PS* and *modEM - PS* algorithms on a problem with $n = 100$.

tion algorithm (herein denoted by *momentumPSO*) proposed in [14]. The results obtained by *momentumPSO* on this problem are shown in Table 2. This table also lists the results obtained by another variant of the particle swarm algorithm, proposed in [16], which incorporates a dynamically adjustable inertia weight parameter in the particle velocity equation (referred as *PSO(Shi+Eberhart)* in the table).

Besides reporting f_{best} and f_{avg}, Table 2 contains other interesting quantities that behave differently as problem dimension increases. They measure the accuracy of the solutions found by the algorithms. One is MAE (see (6)) and the other is the standard deviation (SD) of the solutions found after all the executed runs:

$$SD = \sqrt{\frac{\sum_{i=1}^{nruns}(f_{best}^{i} - f_{avg})^2}{nruns}}.$$

The proposed algorithm achieves in general the lowest numerical errors (MAE) and lowest standard deviations. Using MAE as a measurement of error, Figure 12 shows how the *modEM - PS* error grows from $n = 10$ to $n = 50$ and decreases

Table 2. Performance of *original EM*, *EM - PS*, *modEM - PS* and two PSO algorithms

n	algorithm	f_{opt}	f_{best}	f_{avg}	MAE	SD
10	*original EM*	12.16	12.160	12.160	0.00002	0.0000
	EM - PS		12.160	12.160	0.00002	0.0000
	modEM - PS		12.160	12.160	0.00002	0.0000
	PSO(Shi+Eberhart)		12.160	12.160	0.00002	0.0000
	momentumPSO		12.160	12.160	0.00002	0.0000
25	*original EM*	30.40	30.400	28.762	0.06553	2.5790
	EM - PS		30.400	28.613	0.07149	2.4315
	modEM - PS		30.400	30.400	0.00002	0.0000
	PSO(Shi+Eberhart)		30.400	29.702	0.02790	1.0649
	momentumPSO		30.400	30.201	0.00796	0.3971
50	*original EM*	60.80	54.843	51.220	0.19160	4.3890
	EM - PS		57.821	49.930	0.21741	8.5362
	modEM - PS		60.799	54.545	0.12509	6.9343
	PSO(Shi+Eberhart)		58.475	51.465	0.18669	3.3673
	momentumPSO		60.799	60.749	0.00101	0.2163
75	*original EM*	91.20	81.272	72.587	0.24818	9.4144
	EM - PS		80.280	73.331	0.23825	8.0091
	modEM - PS		91.199	87.724	0.04634	4.1407
	PSO(Shi+Eberhart)		77.543	69.985	0.28287	5.1261
	momentumPSO		91.199	91.082	0.00157	0.5064
100	*original EM*	121.60	120.606	114.699	0.06901	8.1524
	EM - PS		120.606	111.523	0.10077	12.6402
	modEM - PS		121.598	118.416	0.03184	4.0513
	PSO(Shi+Eberhart)		105.446	91.353	0.30247	6.0855
	momentumPSO		121.598	121.249	0.00351	0.8296

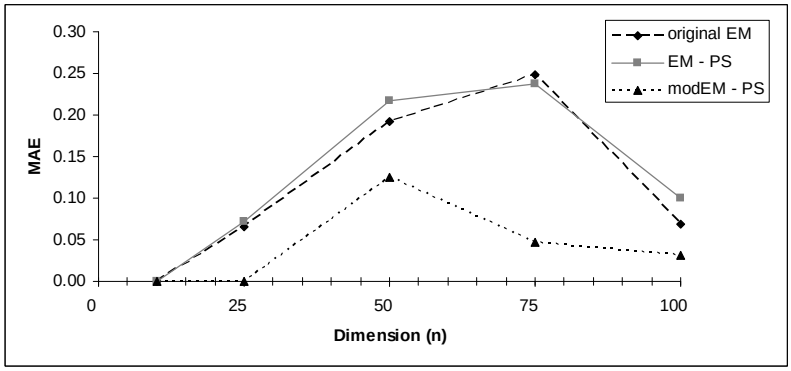

Figure 12. Error behavior for *original EM*, *EM - PS* and *modEM - PS* algorithms.

from $n = 50$ to $n = 100$. The errors of *original EM* and *EM - PS* grow faster with n although also decrease for $n = 100$. Figure 13 allows error comparisons as the problem dimension increases between the EM-like algorithms and the two PSO algorithms.

A second example is used to show the algorithms behavior as n increases. We selected the Neumaier 3 problem, denoted by $NF3$ in [1]. The analytical solution depends on n and is given by $\frac{n(n+4)(n-1)}{6}$. Our analysis uses five values of n: $10, 15, 20, 25, 30$. Table 3 lists values of f_{best}, f_{avg}, MAE and SD obtained with the three EM algorithms, after $100n^2$ function evaluations, over 30 runs, with a population of $\min\{200, 10n\}$ points. For all tested values of n, the f_{best} values found by *EM - PS* and *modEM - PS* are very close to the analytical results. The errors do not grow very much with n. The numerical experiments carried out and described in this subsection allow us to conclude that the proposed *modEM - PS* is the most appropriate for solving large global optimization problems with bounded variables.

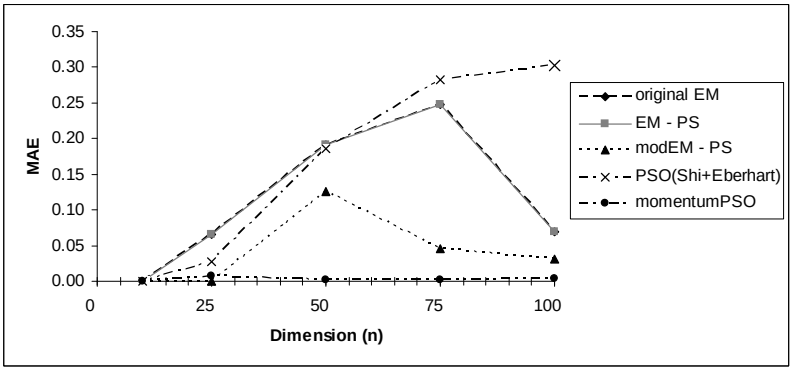

Figure 13. Error behavior for *original EM*, *EM - PS*, *modEM - PS* and two PSO algorithms.

Table 3. Performance of *original EM*, *EM - PS* and *modEM - PS* to solve problem $NF3$

n	algorithm	f_{opt}	f_{best}	f_{avg}	MAE	SD
10	*original EM*	-210	-209.4891	-199.9787	1.0021	10.6693
	EM - PS		-210.0000	-210.0000	0.0000	0.0000
	modEM - PS		-210.0000	-209.9999	0.0000	0.0002
15	*original EM*	-665	-636.1431	-621.7852	2.8810	17.8468
	EM - PS		-664.9996	-664.9903	0.0006	0.0138
	modEM - PS		-664.9999	-664.9935	0.0004	0.0120
20	*original EM*	-1520	-1400.1130	-1363.3129	7.8344	43.0081
	EM - PS		-1519.9895	-1519.7756	0.0112	0.3087
	modEM - PS		-1519.9812	-1519.6476	0.0176	0.5144
25	*original EM*	-2900	-2662.9455	-2609.6381	11.6145	60.7286
	EM - PS		-2899.8501	-2897.6537	0.0939	2.6563
	modEM - PS		-2899.7614	-2897.4835	0.1007	3.0164
30	*original EM*	-4930	-4501.3718	-4403.9782	17.5341	112.0166
	EM - PS		-4927.5608	-4918.9484	0.3684	10.4312
	modEM - PS		-4927.2926	-4922.6403	0.2453	5.6080

4. Conclusions

We have presented modifications to the electromagnetism-like algorithm given in [3] for solving global optimization problems like (1). The crucial modifications are concerned with the local search procedure and the force vector that is used to define the direction of movement of each point in the population.

The local search procedure based on a pattern search algorithm is able to improve accuracy of the found solutions and to perform slightly better than the original EM algorithm especially if a large number of function evaluations is permitted. The effect of the modified force vector to move the points of the population in the EM mechanism has been positive since the corresponding algorithm is proven to be robust, with a fast convergence, and high level of accuracy. From the sensitivity analysis, the value $\beta = 0.1$ for the memory constant seems a good choice.

Further research will consider other approaches to evaluate the charges of the points in the population. A more detailed study concerning the optimal choice for the β parameter, so that the algorithm convergence is improved, is our future challenge.

Acknowledgments

The authors are grateful to the two anonymous referees for their valuable suggestions.

References

[1] M.M. Ali, C. Khompatraporn, and Z.B. Zabinsky, *A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems*, Journal of Global Optimization 31 (2005), pp. 635–672.
[2] S.I. Birbil, *Stochastic global optimization techniques*, PhD Thesis, North Carolina State University, 2002.
[3] S.I. Birbil and S. Fang, *An electromagnetism-like mechanism for global optimization*, Journal of Global Optimization 25 (2003), pp. 263–282.
[4] S.I. Birbil, S. Fang, and R. Sheu, *On the convergence of a population-based global optimization algorithm*, Journal of Global Optimization 30 (2004), pp. 301–318.
[5] D. Debels, B. DeReyck, R. Leus, and M. Vanhoucke, *A hybrid scatter search/electromagnetism metaheuristic for project scheduling*, European Journal of Operational Research 169 (2005), pp. 638–653.
[6] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming 91 (2002), pp. 201–213.
[7] N. Hansen, *The CMA evolution strategy: a comparing review*, in *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, eds., Springer-Verlag, 2006, pp. 75–102.
[8] R. Hooke and T.A. Jeeves, *Direct search solution of numerical and statistical problems*, Journal of the Association for Computing Machinery 8 (1961), pp. 212–229.
[9] W. Huyer and A. Neumaier, *Global optimization by multilevel coordinate search*, Journal of Global Optimization 14 (1999), pp. 331–355.
[10] L. Ingber, *Adaptive simulated annealing (ASA): lessons learned*, Control and Cybernetics 25 (1996), pp. 33–54.
[11] P. Kaelo and M.M. Ali, *Differential evolution algorithms using hybrid mutation*, Computational Optimization and Applications 37 (2007), pp. 231–246.
[12] J. Kennedy and R.C. Eberhart, *Particle swarm optimization*, in *IEEE International Conference on Neural Network*, 1995, pp. 1942–1948.
[13] R.M. Lewis and V. Torczon, *Pattern search algorithms for bound constrained minimization*, SIAM Journal on Optimization 9 (1999), pp. 1082–1099.
[14] J. Liu and J. Lin, *Evolutionary computation of unconstrained and constrained problems using a novel momentum-type particle swarm optimization*, Engineering Optimization 39 (2007), pp. 287–305.
[15] A.M.A.C. Rocha and E.M.G.P. Fernandes, *A modified electromagnetism-like algorithm based on a pattern search method*, in *Lecture Notes in Electrical Engineering, Proceedings of the European Computing Conference*, N. Mastorakis and V. Mladenov, eds., Vol. 2, Part 9, Chapter 12, pp. 1035–1042, Springer-Verlag, (to appear in 2009).
[16] Y. Shi and R.C. Eberhart, *Empirical study of particle swarm optimization*, in *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999, pp. 1945–1950.
[17] V. Torczon, *On the convergence of pattern search algorithms*, SIAM Journal on Optimization 7 (1997), pp. 1–25.
[18] A. Törn, M.M. Ali, and S. Viitanen, *Stochastic global optimization: problem classes and solution techniques*, Journal of Global Optimization 14 (1999), pp. 437–447.
[19] A.I.F. Vaz and L.N. Vicente, *A particle swarm pattern search method for bound constrained global optimization*, Journal of Global Optimization 39 (2007), pp. 197–219.