AN IMPROVED ELECTROMAGNETISM-LIKE ALGORITHM FOR GLOBAL OPTIMIZATION

Javid Jouzdani1*, Farnaz Barzinpour2, Mohammad Fathian3
1,2,3School of Industrial Engineering,
Iran University of Science and Technology, Tehran, Iran
1javidjouzdani@iust.ac.ir, 2barzinpour@iust.ac.ir, 3fathian@iust.ac.ir

ABSTRACT

Global optimization has been an area of interest for researchers for many years and it is gaining more and more attention due to constant increase of complexity and difficulty of real-life optimization problems. In this paper an improved electromagnetism-like algorithm for global optimization which is based on an attraction-repulsion mechanism of different particle types to move the particles towards the optimum solution. The proposed method may be used either as a stand-alone algorithm or as a part of a hybrid algorithm. Some test results on nonlinear problems are presented. The results show that the algorithm yields more satisfactory results than previously proposed methods of this kind.
1 INTRODUCTION

It is proved that conventional methods like gradient techniques are unable to solve most real-life problems of many variables. The emergence of meta-heuristic algorithms was response to the demand for reliable, fast and flexible optimization methods. These algorithms may be classified in two groups: a group originally designed to solve continuous problems and the other group to deal with discrete ones. From a point of view, meta-heuristic methods may be categorized as population-based and single-solution-based.

One of the recently introduced population-based meta-heuristic algorithms for solving continuous optimization problems is Electromagnetism-like algorithm (EM) introduced by Birbil and Fang [2]. The algorithm is based on electromagnetism theory and in its original version each particle is initially placed in a random position of the solution space. The particles are charged according to a normalized value calculated from the values of the objective function. Each particle exerts a force to other particles and in turn is affected by the force that other particles exert on it. The particle is moved in the space toward the optimality.

Although originally designed for continuous optimization with bounded variables, EM is successfully applied to many discrete problems like project scheduling (Debels, De Reyck and Leus [6]), machine scheduling (Chang, Chen and Chin-Yuan [3]), periodic job-shop scheduling (Jamili, Shafia and Tavakkoli-Moghadd [7]), unicost set covering (Naji-Azimi, Toth and Galli [10]), flow-shop scheduling (Davoudpour and Hadji Molana [9], Rahmati, et al. [11], Naderi, et al. [9]) and travelling salesman problem (Wu, Yang and Fang [13], Javadian, Gol Alikhani and Tavakkoli-Moghaddam [8]). EM is also used in continuous problems for example in training artificial neural network for textile retail operations (Wu, Yang and Wei [14], Wu, Yang and Hung [12]).

In this paper an improved version of EM algorithm is presented and some general test functions of general minor and moderate difficulty is provided as a benchmark to show the efficiency of the algorithm. The organization of this paper is as follows. In Section 2, a description of the original EM algorithm and in Section 3, our proposed method is provided. Section 4 included the computational results and comparisons and Section 5 concludes the paper.

2 ORIGINAL EM ALGORITHM

EM was originally designed for solving bounded optimization problems like (1) where \( L \) and \( U \) represent the lower and upper bounds of the decision variable \( x \).

\[
\begin{align*}
\min z &= f(x) \\
\text{subject to} & \quad x \in [L, U] \\
& \quad \quad \quad \quad \quad \quad \quad \quad (1)
\end{align*}
\]

The general procedure for the original EM algorithm is presented in Algorithm 1 where the local search procedure, force calculation and move procedure are repeatedly performed after an initialization phase. These three are the main subroutines of the EM algorithm. In what follows each section is explained briefly.

2.1 Initialization

Initialization is used to place \( m \) particles in the solution space in an \( n \)-dimensional hypercube bounded by the lower bounds and upper bounds of the decision variable. Algorithm 2 shows the algorithm for initialization.

ALGORITHM 1. EM( \( m \), MAXITER, LSITER, \( \delta \) )
Algorithm 1. General procedure of original EM

\begin{algorithm}
\begin{algorithmic}
\State Initialize()
\State $i \leftarrow 1$
\While{$i < \text{maxIter}$}
\State Local($\text{LSITER}, \delta$)
\State CalculateForces()
\State Move()
\State $i \leftarrow i + 1$
\EndWhile
\end{algorithmic}
\end{algorithm}

Algorithm 2. Initialization

\begin{algorithm}
\begin{algorithmic}
\For{$i = 1$ to $m$}
\For{$k = 1$ to $n$}
\State $\lambda \leftarrow U(0,1)$
\State $x_k^i \leftarrow l_k + \lambda(u_k - l_k)$
\EndFor
\State CalculateForces()
\EndFor
\State $x^{\text{best}} \leftarrow \arg \min \{ f(x^i), \forall i \}$
\end{algorithmic}
\end{algorithm}

\section{Local Search Procedure}

In Algorithm 3, the local search procedure is depicted where $\delta$ is the search step size and $\text{LSITER}$ is the maximum number of iterations in this subroutine. Each coordinate of each particle, $x$, is perturb by a random proportion, $\lambda$, of $\delta$. The sign of the perturbation is randomly selected by using a uniformly distributed random variable, $\lambda$. If the operation yields a better solution the coordinates of the original particle, $x$, is updated. In the end the best known solution is updated.

\section{Force Calculation}

The superposition principle, illustrated in Algorithm 4, states that the force exerted on an electromagnetic particle is directly proportional to the product of their charges and inversely proportional to the distance between them (Cowan 1968).

\begin{equation}
q^i = \exp \left( -n \frac{f(x^i) - f(x^{\text{best}})}{\sum_{k=1}^{m} f(x^k) - f(x^{\text{best}})} \right), \forall i
\end{equation}
In EM, each particle is charged according to its objective function value. The charge for each particle is denoted by $q^j$ and is calculated by (2). It can be seen that the higher the objective function value for a particle the more charge it has and the amount. The direction of the force exerted on each particle is given by (3). Algorithm 4 illustrates the force calculation procedure.

**Algorithm 3. Local search procedure**

1. $\text{counter} \leftarrow 0$
2. $\text{length} \leftarrow \delta \left( \max_k \{u_k - l_k\} \right)$
3. for $i = 1$ to $m$ do
4.     for $k = 1$ to $n$ do
5.         $\lambda^i_k \leftarrow U(0,1)$
6.     while $\text{counter} < \text{LSITER}$ do
7.         $y \leftarrow x^i$
8.         $\lambda_2 \leftarrow U(0,1)$
9.         if $\lambda_1 > 0.5$ then
10.            $y_k \leftarrow y_k + \lambda_2 \cdot \text{Length}$
11.        else
12.            $y_k \leftarrow y_k - \lambda_2 \cdot \text{Length}$
13.        end if
14.        if $f(y) < f(x^i)$ then
15.            $x^i \leftarrow y$
16.            $\text{counter} \leftarrow \text{LSUTER} - 1$
17.        end if
18.        $\text{counter} \leftarrow \text{counter} + 1$
19.     end while
20. end for
21. end for
22. $x_{\text{best}} \leftarrow \arg \min \{f(x^i), \forall i\}$

**Algorithm 3. Local search procedure**

2.4 Move mechanism

Algorithm 5 shows the move mechanism in which the position of each particle is updated according to the total normalized force exerted on it. $u$ and $l$ are the upper and lower bounds respectively. Line 7 and 9 of the algorithm guarantees the feasibility of the newly reached position. The noticeable point is that the best point is not moved during the procedure.
ALGORITHM 4. CalculateForce()

1: for \( i = 1 \) to \( m \) do
2: \[ q^i \leftarrow \exp \left( -n \sum_{k=1}^{n} f(x_k) - f(x_{\text{best}}) \over \sum_{k=1}^{n} f(x_k) - f(x_{\text{best}}) \right) \]
3: \( F^i \leftarrow 0 \)
4: end for
5: for \( i = 1 \) to \( m \) do
6: for \( j = 1 \) to \( m \) do
7: if \( f(x_i) < f(x^j) \) then
8: \( F^i \leftarrow F^i + (x^j - x^i) \over \|x^j - x^i\|^2 \)
9: else
10: \( F^i \leftarrow F^i - (x^j - x^i) \over \|x^j - x^i\|^2 \)
11: end if
12: end for
13: end for

Algorithm 4. Force calculation subroutine

ALGORITHM 5. Move()

1: for \( i = 1 \) to \( m \) do
2: if \( i \neq \text{best} \) then
3: \( \lambda \leftarrow U(0,1) \)
4: \( F^i \leftarrow F^i \|F^i\| \)
5: for \( k = 1 \) to \( n \) do
6: if \( F^i_k > 0 \) then
7: \( x_k^i \leftarrow x_k^i + \lambda F^i_k (u_k - x_k^i) \)
8: else
9: \( x_k^i \leftarrow x_k^i + \lambda F^i_k (x_k^i - l_k) \)
10: end if
11: end for
12: end if
13: end for

Algorithm 5. The move mechanism

3 IMPROVED ELECTROMAGNETISM-LIKE ALGORITHM

In this section the improved algorithm is described and its steps are explained. The Improved EM algorithm includes 4 main steps: initialization, move, local search, solution refinement and another local search. Algorithm 6 illustrates the algorithm of the improved EM
algorithm. In addition to addition of a local search step, the contribution of this research lies mainly in the solution refinement phase. In the next section, this phase is described.

**ALGORITHM 6. Improved-EM**

1: Initialize()
2: \( i \leftarrow 1 \)
3: \( \text{while } i < \text{maxIter}, \text{ and } T < T_f \) do
4: \( \text{CalculateForces()} \)
5: \( \text{Move()} \)
6: \( \text{Local}(\text{LSITER}, \delta_i) \)
7: \( \text{RefineSolutions()} \)
8: \( \text{Local}(\text{LSITER}, \delta_2) \)
9: \( T \leftarrow \text{schedule}(i) \)
10: \( i \leftarrow i + 1 \)
11: end while

**Algorithm 6. General algorithm of the improved EM**

It should be mentioned that the local search step sizes are set dynamically based on the standard deviation of the charges of the particles. Following equations show the formula based on which the step sizes are calculated.

\[
\delta_i = \ln(n_i) \left( \sigma \{ q_i | i = 1, 2, K, n \} \right)^{\ln(n_i)} \\
\delta_2 = 0.01\delta_i
\]  

where \( \sigma(g) \) denotes the standard deviation, \( n \) is the dimension of the search space, \( i \) is the iteration counter and \( q_i \) is the charge of the \( i^{th} \) particle. This way of defining the step size makes the search flexible; the step size decreases exponentially when the number of iterations or the dimension of the space increases and increases logarithmically when the number of iterations or the dimension of the space increases. For example when the particles are near the best solution, and the number of iterations is large (i.e. the algorithm is in its final stages), the step size is small and causes more intensification and when the algorithm is in its early stages, and the particles have very different charges, the step size is rather large and therefore more diversification is provided. Another advantage of this method is that the step size is not considered as a parameter.

### 3.1 Solution refinement phase

In this step, the population is refined through a tournament-like refinement of solutions. The whole procedure is an elitism approach that is followed for \( \theta \) percent of the population with the lowest objective function value. The best and the worst solutions in the population are determined in Lines 3 and 4 of the algorithm, respectively. If the best solution of the population is better than the best known solution, the worst individual is replaced by a particle placed on a straight line connecting the worst solution to the best. This makes the worst solution closer to the best solution. However, the replacement occurs even if the best individual of the population is not better than the best known solution with a probability, similar to what happens in Simulated Annealing algorithm. The probability depends on the temperature of the solution space and the objective function value of the worst and the best individual in the population (see Line 11 in Algorithm 7).
Algorithm 7. General algorithm of solution refinement in improved EM

where \( m \), the number of particles, is considered to be equal to 5 times of the search space dimension.

Figure 1. Rastrigin function surface
4 COMPUTATIONAL RESULTS

4.1 Unconstrained Problems

We performed a number of tests using some general test functions (Birbil and Fang [2]). The algorithm is coded in MATLAB 2009a and is run on a PC equipped with Intel® Core™2 Duo CPU T3800 @ 2.40GHz and 2.00 GB of RAM with Microsoft Windows® XP SP2 operating system. The results are presented in the following table. As an example the plot for Rastrigin function is presented in Figure 1 and the behavior of the particles for this function is illustrated in Figure 2.

![Figure 2. Particle behaviour for Rastrigin function](image)

In the above figure, the blue points are the positions chosen by the particles, the yellow point are the current position of the particles and the green points are the initial positions. The red star shows the best known solution (i.e. the best particle).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Davis</th>
<th>Rastrigin</th>
<th>Sine Envelope</th>
<th>Stenger</th>
<th>Trid5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst ( f(x^*) )</td>
<td>0.6305</td>
<td>-1.9753</td>
<td>0.0000</td>
<td>0.0470</td>
<td>-29.9994</td>
</tr>
<tr>
<td>Best ( f(x^*) )</td>
<td>0.0008</td>
<td>-2.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-30.0000</td>
</tr>
<tr>
<td>Average ( f(x^*) )</td>
<td>0.0815</td>
<td>-1.9957</td>
<td>0.0000</td>
<td>0.0040</td>
<td>-29.9999</td>
</tr>
<tr>
<td>Std. ( f(x^*) )</td>
<td>0.1385</td>
<td>0.0064</td>
<td>0.0000</td>
<td>0.0092</td>
<td>0.0001</td>
</tr>
<tr>
<td>Time for Best ( x^* )</td>
<td>0.42 sec</td>
<td>0.33 sec</td>
<td>0.33 sec</td>
<td>0.31 sec</td>
<td>3.60 sec</td>
</tr>
<tr>
<td>Average time</td>
<td>0.40 sec</td>
<td>0.33 sec</td>
<td>0.34 sec</td>
<td>0.32 sec</td>
<td>3.62 sec</td>
</tr>
</tbody>
</table>

Considering time spent on finding the results it can be claimed that the algorithm is able to
reach a satisfying solution in a reasonable time. In addition, due to improvements in the algorithm, some kinds of the problems can be solved more accurately (as for Sine Envelope Function) than the original EM algorithm (Birbil and Fang [2]).

4.2 Constrained Problems

By following the approach proposed in (Ali and Golalikhani [1]), the constrained problems can be solved easily and efficiently by the proposed algorithm. The following programs are solved by this approach as examples and the results are presented in Table 3. The results show that the improved EM algorithm not only outperforms the original version, it is also capable of solving constrained problems efficiently.

\[
(P1) \min z = x_1^2 + x_2^2 \\
\text{subject to} \\
x_1^2 + x_2^2 - 12.25 \leq 0
\]

\[
(P2) \min z = Rastrigin (x) \\
\text{subject to} \\
x_1^2 + x_2^2 - 12.25 \leq 0
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst ( f(x^*) )</td>
<td>0.2500</td>
<td>-0.5687</td>
</tr>
<tr>
<td>Best ( f(x^*) )</td>
<td>0.2500</td>
<td>-1.6915</td>
</tr>
<tr>
<td>Average ( f(x^*) )</td>
<td>0.2500</td>
<td>-1.6513</td>
</tr>
<tr>
<td>Std. ( f(x^*) )</td>
<td>0.0000</td>
<td>0.1569</td>
</tr>
<tr>
<td>Constraint</td>
<td>0.0000</td>
<td>-0.3838</td>
</tr>
<tr>
<td>Time for Best ( x^* )</td>
<td>0.20 seconds</td>
<td>0.20 seconds</td>
</tr>
<tr>
<td>Average time</td>
<td>0.21 seconds</td>
<td>0.20 seconds</td>
</tr>
</tbody>
</table>

5 CONCLUSION AND FUTURE WORKS

In this paper, an improved electromagnetism-like algorithm for global optimization based on an attraction-repulsion mechanism of different particle types is proposed. In our algorithm, the particles move towards the optimum solution under the influence of the forces calculated forces. The results show that the algorithm yields more promising results than previously proposed methods of this kind. Applying this algorithm in real-world problems may be considered as one of our future works. In addition, different types of hybridizations of this improved EM algorithm with other meta-heuristics may be an interesting area of study from both theoretical and practical points of view.
REFERENCES


