IMPLEMENTATION OF A FIXING STRATEGY AND PARALLELIZATION IN A RECENT GLOBAL OPTIMIZATION METHOD

Figen Öztoprak[1], Ş.İlker Birbil
Sabancı University
Istanbul, Turkey
figen@su.sabanciuniv.edu, sibirbil@sabanciuniv.edu

Abstract. Electromagnetism-like Mechanism (EM) heuristic is a population-based stochastic global optimization method inspired by the attraction-repulsion mechanism of the electromagnetism theory. EM was originally proposed for solving continuous global optimization problems with bound constraints and it has been shown that the algorithm performs quite well compared to some other global optimization methods. In this work, we propose two extensions to improve the performance of the original algorithm: First, we introduce a *fixing strategy* that provides a mechanism for not being trapped in local minima, and thus, improves the effectiveness of the search. Second, we use the proposed fixing strategy to parallelize the algorithm and utilize a *cooperative parallel search* on the solution space. We then evaluate the performance of our study under three criteria: the quality of the solutions, the number of function evaluations and the number of local minima obtained. Test problems are generated by an algorithm suggested in the literature that builds test problems with varying degrees of difficulty. Finally, we benchmark our results with that of the Knitro solver with the multistart option set.

Keywords: global optimization, electromagnetism-like mechanism, parallel search

1. Introduction

Electromagnetism-like Mechanism (EM) heuristic was originally proposed for solving continuous global optimization problems with bound constraints (Birbil and Fang, 2003). EM is a population-based stochastic global optimization method inspired by the attraction-repulsion mechanism of the electromagnetism theory. It is shown that after a modification, the algorithm converges to the global optimum with probability one, when the number of iterations is large enough (Birbil, 2004).

In this paper, we present our work on improving the performance of the original EM. In this context, we extend the heuristic in two ways: (i) adding a mechanism for controlling local optimality and applying a *fixing strategy* not to be trapped in the attractive local minima, (ii) parallelizing the search. We try three cooperation scenarios in the parallelization process and use the fixing strategy by means of cooperation between the parallel search threads. We test the performance of the extensions we propose in terms of solution quality, number of function evaluations, and number of local minima obtained. Test problems are generated by the GKLS software (Gaviano et al., 2003). To provide a more comprehensive performance evaluation, we also benchmark our results with that of the *Knitro*[1] solver.

2.Electromagnetism-like Mechanism

EM starts with an initial solution set (particles) and then, an attraction-repulsion mechanism is used iteratively to move those particles towards optimality. The general scheme of the algorithm is given in Figure 1 (except the shaded lines). Here, m is the number of particles, `MAXITER` is the maximum number of iterations, LSITER is the maximum number of local search iterations, and δ is the local search parameter.

[1] Corresponding author

In the Initialize procedure, m particles are randomly generated from the feasible region. The procedure Local is then applied, where the particle with the best objective function value (best particle) takes random (but feasible) steps in each coordinate direction in order to find locations with better function values, i.e., a coordinate search is applied to the best particle. Then, the total force vectors exerted on the particles are calculated. The force exerted on a particle via other particles is inversely proportional to the distance between the particles and directly proportional to the product of their charges. The best particle then attains the maximum charge. The signs of the force vectors are assigned by comparing the objective function values of the related particles. That is, a particle with a better objective function value attracts the other particles with a positive sign force vector. Finally, each particle (except the best one) moves in the direction of the total force vector by a random step length along each coordinate. The algorithm terminates when MAXITER iterations are completed. The complete algorithm and the details of procedures are given in (Birbil and Fang, 2003).

ALGORITHM EM *with the fixing strategy* (m, MAXITER, LSITER, FXITER, δ, ε)

```
1: Initialize()§
2: iteration ← 1
3: while iteration < MAXITER do
4:    Local(LSITER, δ) §
5:    F ← CalcF()§   //modify CalcF(): fixed particles always repel
6:    Move(F) §      //modify Move(F): fixed particles do not move
7:    for i=1 to m do
8:        if TotalMove(Particle[i], FXITER) < ε then  // check the total move in the last FXITER iterations
9:            label Particle[i] as "fixed"
10: iteration ← iteration + 1
```

(§ see Birbil and Fang, 2003)

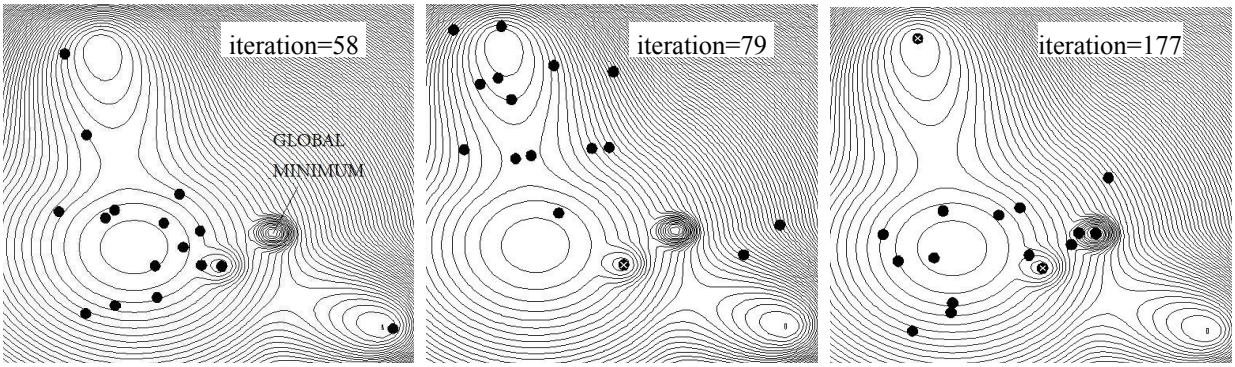
Fig.1. The outline of EM with the fixing strategy

3.The Fixing Strategy

The attraction-repulsion mechanism of EM and the randomness in its procedures may cause visits to the same local minimum several times. Consequently, an early completion due to exhaustion of the maximum number of iterations may result with missing the global minimum. The fixing strategy suggests placing some fixed particles to the locations that are local minimum candidates. Unlike the standard particles of the original method, the fixed particles do not search; they just repel mobile particles providing them to search for new local optima existing in the yet undiscovered part of the solution space. The required changes in the original algorithm are shown by shaded lines in Figure 1. In this work, we do not use any derivative information to decide if a location is a local minimum. We accept that a particle is a local minimum if it does not leave the ε - neighbourhood for a certain number of iterations (FXITER).

Figure 2 illustrates the strategy on an example GKLS problem. The two dimensional problem has 5 local minima. In Figure 2-a, the algorithm finds out one of the local minima that is quite close to the global minimum. Since a better solution cannot be found for several iterations, the particle at this location does not move during a specified number of iterations. Thus, we suspect that it is on a local minimum and label it as fixed (marked with a cross sign in Figure 2-b). So, it starts to repel other particles. In Figure 2-b, the nonfixed particles discover another local minimum and another particle becomes fixed at this

location. The nonfixed particles are repelled by the first two minima, and hence, they are encouraged to find the global minimum (Figure 2-c).



(a) (b) (c)

Fig. 2. An illustration of the fixing strategy

4.The Parallel Implementation

There is a considerable amount of work on parallelizing some well-known meta-heuristics like simulated annealing, tabu search, and genetic algorithms that apply various parallelization strategies (Migdalas, 2003). Crainic et al. (2004) state that multiple search parallel algorithms outperform their sequential counterparts and cooperative multiple search strategies generally perform better than the independent multiple search methods.

In this study, we design several parallel versions of EM not only for increasing the computational power but also for improving the performance of the algorithm. Our experience shows that the algorithm performs better as the number of particles is increased to a certain extent, but beyond that the increase in the number of particles adversely affects the algorithm. This is due to the noise caused by too many interactions between the particles. Therefore, the parallel implementation with smaller groups seems like a viable and, as our results show, a promising approach.

In our parallel design, each parallel processor runs the complete EM algorithm in interaction with other processors. The cooperation is provided by sharing the coordinates of the fixed particles so that the copies of these particles can be produced in the receiver processes to support the application of the fixing strategy. Information exchange among parallel processors is via messaging, i.e., no common central memory is used. The communication is quite restricted, so messaging does not cost significant increase in the computation time. We illustrate the idea of the cooperation in Figure 3 by solving the same example problem as in Section 3 with two processors; one half of the 16 particles in Section 3 is now used by Process 1 and the other half by Process 2: At early iterations, both processes are looking around the same region. This is the relatively large attraction region of the most attractive local minimum (the paraboloid vertex). (a) – (d) Process 2 then obtains the local minimum, fixes a particle at that location and then, sends it to Process 1. Consequently, the particles in both processes are now directed to other regions of the solution space. Furthermore, Process 1 no more has to spend its resources. (b) - (e) At later iterations, Process 1 is the first one that finds another local minimum with a better objective function value. It then fixes a particle there and shares it with Process 2 so that particles of Process 2 soon leave the related region which has already been explored by Process 1. (c) - (f) Finally, both processes find the global minimum; Process 1 finds it first. Since both processors share information, we call this scenario as the *full cooperation*. The complete set of proposed scenarios is explained next.

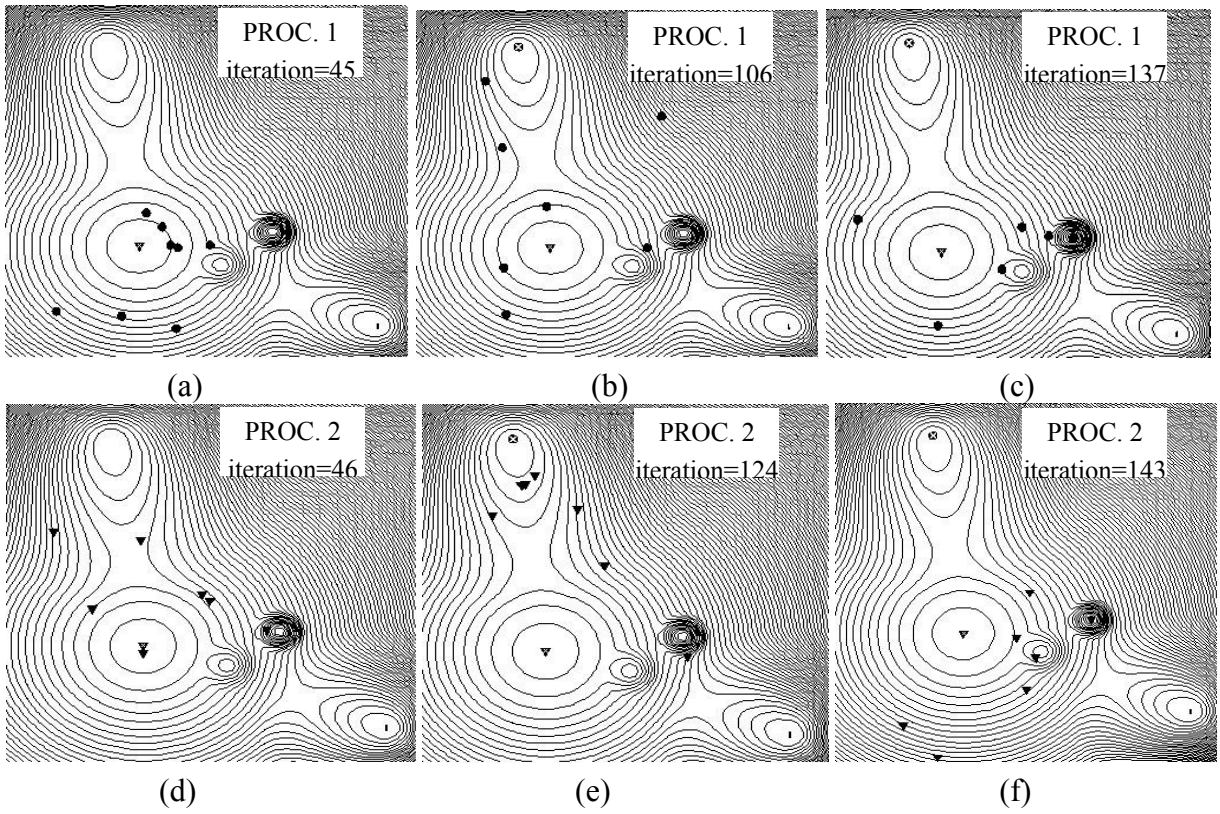


Fig.3. An illustration of the cooperation among the parallel processors

We try three parallelization scenarios: no cooperation (Scenario 0), limited cooperation (Scenario 1) and full cooperation (Scenario 2). In *no cooperation* scenario, parallel processors do not communicate during their search and results are gathered when they terminate. In other words, we divide the *EM particles team* into smaller search teams so that the communications among the teams are not allowed. The *limited cooperation* scenario is similar to a master-slave model. One of the processors is the root and the main process runs on it. The role of other processors is to support the root by informing it when they fix a particle. However, all results are also gathered when parallel search is completed since a non-root process may still find a better solution than the root does. In the *full cooperation* scenario, there is no root process and all processors communicate: When one of the processors has a fixed particle, it sends its information to all other processors. Again, the results are gathered after termination.

5.Computational Results

As we have mentioned above, our motivation in parallelizing EM is not simply making the algorithm run faster but improving its overall performance. So, our tests are based on three criteria: the quality of solutions, the number of function evaluations and the number of local minima recovered. To provide a fair benchmark, we use the same algorithm parameters (including the number of particles) for all test configurations. The algorithms are coded in C++ and MPI is used for parallel programming.

The properties of our test problems are summarized in Table 1 (cf. Gaviano et al., 2003). For each combination of parameters, 10 problems are generated, so we have a total of 360 different test problems. Since we implement a stochastic search method, we generated 10 replicates with different seeds for each problem. In total 3600 problems are solved. All problems are continuous global optimization problems with bound constraints. Here, global radius is the radius of the attraction region of the global minimum and

global distance is the distance between the global minimum and the parabolid vertex. The global minimum value for all of our test problems is -1. We generated differentiable functions to be able to use them also to test Knitro[2].

Table 1. The parameters for GKLS test problems

Dimension	2, 5, 10
Number of local minima	5, 10, 50
Global radius	1%, 5% of the distance between lower and upper bounds
Global distance	12.5%, 25% of the distance between lower and upper bounds

We apply EM with $m = 8 \times n/p$, MAXITER = $100 \times n$, where m, n and p denote the number of particles, the problem dimension and the number of processors respectively. The local search is applied only to the best particle, and the maximum number of local search iterations LSITER is 3. We fix a particle if it moves less than 10^{-9} during $0.2 \times$MAXITER iterations. Local search parameter δ is 10^{-3}. We stop the algorithm when it completes maximum number of iterations or when it becomes close enough to the global optimum. Following (Birbil and Fang, 2003), we are satisfied by arriving $10^{-4}|f_{glob}|$ neighbourhood of the global minimum f_{glob}. To have a fair comparison, we use the same set of seeds whenever the number of processors is equal. We select the following option settings for Knitro version 5.1: `hessopt = bfgs`, `ms_enable = yes` (multistart option is set). All other options are set to their default values (number of solutions to compute in case of multistart is $min(200, 10 \times n)$ by default).

First, we test performance of the fixing strategy versus classical EM, and also compare the performance of both versions of EM with Knitro. As Table 2 indicates, in terms of the solution quality, EM with fixing strategy is the best on average.

Table 2. The performance benchmark for the fixing strategy

		Method		
		EM	**EM with fixing strategy**	**Knitro with multistart**
Average final objective function value	n=2	-0.7164	-0.7416	-0.3742
	n=5	-0.0922	-0.1105	-0.0836
	n=10	0.0063	0.0087	-0.0087
	overall	***-0.2674***	***-0.2811***	***-0.1555***
Percentage of results with the global minimum	n=2	53.75%	57.67%	16.67%
	n=5	8.08%	9.17%	3.34%
	n=10	0.08%	0.08%	0.83%
	overall	***20.74%***	***22.31%***	***6.94%***
Average number of function evaluations*	n=2	675.86	766.60	819.66
	n=5	1660.45	1916.36	1559.74
	n=10	2384.34	2507.85	3508.52
	overall	***1573.55***	***1730.27***	***1962.64***

(* number of gradient evaluations for Knitro is added)

Second, we try to see how number of parallel search threads affects the performance of the algorithm. From Table 3 we realize that a certain degree of parallelization may improve the performance but there is not a monotone relationship between the number of processors and the performance criteria.

[2] Knitro does not allow starting with different seeds, thus the experiments with Knitro involve only single replication.

Also, applying the fixing strategy in 4 parallel processors is the least successful probably due to the decrease in the number of particles caused by fixing, and also because of using only $2 \times n$ particles, which could be far too few to carry out the search.

Table 3. The performance benchmark for the parallelization (no communication scenario)

		Classical EM			EM with fixing strategy		
		1 proc.	**2 proc.**	**4 proc.**	**1 proc.**	**2 proc.**	**4 proc.**
Percentage of results with the global minimum	n=2	53.75%	54.25%	52.00%	57.67%	59.75%	45.33%
	n=5	8.08%	9.67%	11.92%	9.17%	7.08%	6.08%
	n=10	0.08%	0.17%	0.00%	0.08%	0.17%	0.25%
	overall	***20.74%***	***21.36%***	***21.31%***	***22.31%***	***22.33%***	***17.22%***
Average number of local minima obtained	n=2	1.15	1.38	2.14	1.42	1.71	2.43
	n=5	1	1.10	1.56	1.07	1.14	1.54
	n=10	1	1.02	1.90	0.99	1.02	2.09
	overall	***1.05***	***1.17***	***1.87***	***1.16***	***1.29***	***2.02***
Average number of function evaluations	n=2	675.86	848.91	959.02	766.60	1047.22	751.15
	n=5	1660.45	2226.43	2978.80	1916.36	2343.18	2933.65
	n=10	2384.34	3839.18	5568.38	2507.85	3827.73	6887.28
	overall	***1573.55***	***2311.51***	***3168.73***	***1730.27***	***2406.04***	***3524.03***
Average CPU time	n=2	0.106	0.043	0.029	0.103	0.042	0.029
	n=5	2.849	0.876	0.479	2.805	0.864	0.499
	n=10	36.589	10.438	5.537	36.467	10.056	5.527
	overall	***13.182***	***3.786***	***2.014***	***13.125***	***3.654***	***2.018***

Finally, we test the performance of different parallelization scenarios on two processors. We again solve 3600 problems. Table 4 summarizes the results. Our first impression from the figures in this table is that the cooperation scenarios are not relatively successful on the average.

Table 4. The performance benchmark for the parallelization scenarios

		2 processors			
		No comm. without fixing	No comm. with fixing	Limited comm.	Full comm.
Percentage of results with the global minimum	n=2	54.25%	59.75%	57.33%	56.00%
	n=5	9.67%	7.08%	6.33%	5.83%
	n=10	0.17%	0.17%	0.17%	0.17%
	overall	***21.36%***	***22.33%***	***21.28%***	***20.67%***
Average number of local minima obtained	n=2	1.38	1.71	1.58	1.50
	n=5	1.10	1.14	1.13	1.10
	n=10	1.02	1.02	1.02	1.01
	overall	***1.17***	***1.29***	***1.24***	***1.20***
Average number of function evaluations	n=2	848.91	1047.22	1132.10	1049.99
	n=5	2226.43	2343.18	2916.15	2313.12
	n=10	3839.18	3827.73	4867.50	4222.53
	overall	***2311.51***	***2406.04***	***2971.92***	***2528.55***

	n=2	0.043	0.042	0.042	0.042
Average	n=5	0.876	0.864	0.922	0.940
CPU time	n=10	10.438	10.056	10.731	10.711
	overall	***3.786***	***3.654***	***4.024***	***4.022***

We observe that the figures in Table 4 do not necessarily show that the proposed cooperation is always inferior. As a further analysis, we compare the results of scenarios 0 and 2 for 3600 problems. Since we apply both scenarios with the same seeds, we can give a one-to-one comparison: In 134 problems, the cooperation has encouraged the algorithm to find a better solution than it would find without cooperation; in 61 problems, the cooperation actually led to the global minimum.

6.Conclusion

In this paper, we proposed a fixing strategy for EM and also studied the paralel version of the algorithm. The experiments on a large set of problems reflected that the fixing strategy may lead to better results. The proposed fixing strategy can still be improved; for instance, using the derivative information would probably accelerate the algorithm since a local minimum can be identified immediately. We also observed that parallelization had a certain potential. However, it is not easy to arrive at general conclusions about the performance of parallelization and the cooperation scenarios. Clearly, the performance depends on many factors about the problem at hand. Thus, our future work involves performance tests on different sets of problems. In the long run, we also plan to switch from parallel processes to a cooperative multiagent optimization environment.

References

Birbil, Ş.İ., Fang, S.-C.(2003) "An electromagnetism-like mechanism for global optimization", Journal of Global Optimization, Vol. 25 No 3, pp.263-282.

Birbil, Ş.İ., Fang, S.-C., Sheu, R.-L.(2004) "On the convergence of a population-based global optimization algorithm", Journal of Global Optimization, Vol.30 No 2, pp.301-318.

Gaviano, M, Kvasov, D.E., Lera, D., Sergeyev, Y.D.(2003), "Algorithm 829: Software for Generation of Classes of Test Functions with Known Local and Global Minima for Global Optimization", ACM Transactions on Mathematical Software, Vol. 29, No 4, pp. 469–480.

Migdalas, A., Toraldo, G., Kumar, V.(2003) "Nonlinear optimization and parallel computing", Parallel Computing, Vol. 29, pp. 375-391.

Crainic,T.G., Gendreau, M., Hansen, P., Miladenovic, N.(2004) "Cooperative parallel variable neighborhood search for the p-median", Journal of Heuristics, Vol. 10, pp. 293-314.

[1] http://www.ziena.com