A Memetic Differential Evolution Algorithm for Continuous Optimization

Santiago Muelas, Antonio LaTorre, José-María Peña
CeSViMa, Parque Tecnológico UPM Campus de Montegancedo, 28223, Pozuelo de Alarcón
Facultad de Informática, Universidad Politécnica de Madrid, 28660, Boadilla del Monte
Madrid, Spain
{*smuelas, atorre, jmpena*}*@fi.upm.es*

***Abstract*—Continuous optimization is one of the most active research lines in evolutionary and metaheuristic algorithms. Since CEC 2005 and CEC 2008 competitions, many different algorithms have been proposed to solve continuous problems. Despite there exist very good algorithms reporting high quality results for a given dimension, the scalability of the search methods is still an open issue. Finding an algorithm with competitive results in the range of 50 to 500 dimensions is a difficult achievement. This contribution explores the use of a hybrid memetic algorithm based on the differential evolution algorithm, named MDE-DC. The proposed algorithm combines the explorative/exploitative strength of two heuristic search methods, that separately obtain very competitive results in either low or high dimensional problems. This paper uses the benchmark problems and conditions required for the workshop on "evolutionary algorithms and other metaheuristics for Continuous Optimization Problems – A Scalability Test" chaired by Francisco Herrera and Manuel Lozano.**

***Keywords*-memetic algorithms, de, continuos optimization**

I. Introduction

Continuous optimization is a field of research which is getting more and more attention in the last years. Many real-world problems from very different domains (biology, engineering, data mining, etc.) can be formulated as the optimization of a continuous function. These problems have been tackled using evolutionary algorithms (EA) or similar metaheuristics [1], [2].

Selecting an appropriate algorithm to solve a continuous optimization problem is not a trivial task. Although a particular algorithm can be configured to perform properly in a given scale of problems (considering the number of variables as their dimensionality), the behavior of the algorithm degrades as this dimensionality increases, even if the nature of the problem remains the same.

In this contribution, a new hybrid method is presented to deal with continuous problems through different problem sizes. The proposed approach is named MDE-DC that stands for memetic differential evolution with deterministic and cancelable local search. MDE-DC uses a local search mechanism to improve the solutions obtained by the differential evolution (DE) optimization. The rationale behind this hybrid optimization strategy is founded on the following facts: (i) DEs are good competitors with state-of-the-art evolution strategies (ESs), like CMA-ES [1], which are among the best performing techniques in low-dimensionality complex continuous problems and, (ii) local search methods, as for example, the multiple trajectory search algorithm (MTS) [2], have shown a great performance in large-scale optimization problems.

MDE-DC has been designed to use the DE search as the baseline algorithm, performing a local search after a certain number of generations. These local search phases are executed applying one of the MTS local search algorithms (LS1, which is actually the responsible of most of the MTS performance). If MDE-DC detects that an application of the local search does not improve the results, the rest of the local search iterations are cancelled.

In this paper we will show that the MDE-DC hybrid algorithm actually obtains the best results compared with their components (DE and LS1) using the problems already presented in the "*Special Session and Competition on Large Scale Global Optimization*" held at the CEC 2008 congress [3], together with additional problem functions. The experiments have been carried out on a wide range of dimensions in order to evaluate the behavior of the algorithms as the number of variables increases.

The rest of the paper is organized as follows: In Section II, relevant related work is briefly reviewed. Section III, presents the algorithm MDE-DC and its parameters. In Section IV, the experimental results are reported. Finally, in Section V, the conclusions and further research lines are discussed.

II. Preliminaries

In the "*Special Session on Real-Parameter Optimization*" held at the CEC 2005 Congress, a benchmark of 25 continuous functions was proposed [4]. In this competition, the CMA-ES algorithm [1], obtained the best results among all the evaluated techniques. However, the use of a covariance matrix makes it not appropiate for high dimensional functions. Differential evolution showed to be a competitive alternative in 10-dimensions: SaDE, a self-adaptive DE [5], obtained the third position in the final funcion value and a real-coded DE algorithm [6] was ranked third in success performance.

On the other hand, in the "*Special Session and Competition on Large Scale Global Optimization*" held at the

CEC 2008 Congress, the session focused on 7 functions with 100, 500 and 1000 dimensions [3]. The MTS algorithm [2], which combines several local search strategies using a small population, was the champion of the competition.

Considering both results, an intuitive approach would be to combine the local search algorithms, in particular those similar to the ones proposed by MTS, with an appropriate DE algorithm. This hybrid approach is equivalent to the memetic algorithms in classical genetic algorithms.

Memetic differential evolution has appeared in the literature in the last few years with interesting approaches. Gao and Wang [7] proposed CSDE1, a memetic DE, to optimize thirteen 30 dimensional continuous problems. CSDE1 uses simplex (Nelder-Mead method) to perform local search using also chaotic systems to create the initial population. CSDE1 applies the local search only to the best individual in the population of each generation.

Tirronen et al. [8] designed a hybrid DE algorithm that combined the Hooke-Jeeves algorithm (HJA) and stochastic local search (SLS) coordinated by an adaptive rule that estimates fitness diversity using the ratio between the standard deviation and the average fitness of the population. This algorithm was compared against a regular DE and an ES to weight coeficients to detect defects in paper production.

The SFMDE (super-fit memetic differential evolution) [9] hybridizes the DE framework with other three algorithms: the particle swarm optimization, the Nelder-Mead algorithm and the Rosenbrock algorithm. This algorithm was tested on two engineering problems: the optimal control drive design for a direct current motor and the design of a digital filter for image processing purposes.

The FAMA (fast adaptive memetic algorithm) [10], proposed by Caponio et al., is a memetic algorithm with a dynamic parameter setting and two local searchers adaptively launched, either one by one or simultaneously, according to the needs of the evolution. The employed local search methods are: the Hooke-Jeeves method and the Nelder-Mead simplex. The Hooke-Jeeves method is executed only on the elite individual while the Nelder-Mead simplex is carried out on 11 randomly selected individuals. FAMA includes a self-adaptive criterium based on a fitness diversity measure and the iteration number. Mutation probability and other search parameters depend also on the diversity measure. FAMA was compared against Tirronen's algorithm and SFMDE obtaining better results for the problem of permanent magnet synchronous motors [11].

Finally, Subudhi et al. [12] applied memetic differential evolution methods to train neural networks. In this case, the local search algorithm is based on a back-propagation mechanism. The algorithm was tested against a genetic algorithm-based equivalent (using also neuronal back-propagation).

III. Proposal

The proposed memetic differential evolution algorithm (MDE-DC) combines a standard differential evolution algorithm with the first local search of the MTS algorithm.

The MTS algorithm was designed for multi-objective problems but it has also obtained very good results with large scale optimization problems. In fact, it was the best algorithm of the *Special Session and Competition on Large Scale Global Optimization* [3]. At each iteration, the MTS algorithm applies three different local searches to a single individual and selects the best from the new three solutions.

The DE algorithm is one of the recent algorithms that, due to its results, has quickly gained popularity on continuous optimization. In the last IEEE competitions on continuous optimization, a DE-based algorithm has always reached one of the best three positions. Nevertheless, DE is subject to stagnation problems which could heavily influence the convergence speed an the robustness of the algorithm [13]. Therefore, the proposed algorithm tries to assist the explorative power of the DE by hybridizing it with a exploitative local search which has proven to obtain good results with similar functions. The reason for selecting only the first of the three local searches of the MTS is that, in a previous study of the authors on the same set of functions, this local search was the one that achieved the greatest improvements. As a result of the constraint on the number of evaluations of the Workshop, it was decided to discard the other two local searches in order to provide the DE with more evaluations.

Algorithm 1 MDE-DC Algorithm

```
for $i = 1$ to $PopulationSize$ do
    $Improve[i] = TRUE$
    $SearchRange[i] = (UPPER\_BOUND - LOWER\_BOUND)/2$
    $Population[i] = RandomInitialization$
end for
while $\#Evaluations \leq max\_evaluations$ do
    $DE(Population)$
    if $generation \% localsearch\_freq = 0$ then
        for $j = 1$ to $j < max\_localsearch\_iters$ do
            $X_k = bestIndividualFromPopulation$
            $Improve[k], SearchRange[k] = LocalSearch1(X_k, Improve[k], SearchRange[k])$
            if $Improve[k] = FALSE$ then
                break
            end if
        end for
    end if
end while
```

The details of the algorithm are presented on algorithm 1 and can be summarized as follows: First, the population of individuals is uniformly initialized on the function search space range. Then, a generation of the DE shown on algorithm 2 is executed. After $localsearch\_freq$ generations of the DE, the local search of algorithm 3 is executed over the best solution of the population. The LS1 algorithm is executed for at most $max\_localsearch\_iters$ iterations depending on whether the individual is improved by the local search procedure or not. These two steps are repeated until the maximum number of evaluations is reached. Since the DE and the LS1 could exceed the number of evaluations until the next verification, both algorithms have been implemented so that they stop their execution if this condition is satisfied during one of their steps. In order to clarify the description of the algorithms, this detail of the implementation has not been included in Algorihtms 2 and 3.

Algorithm 2 LS1($X_k, Improve, SR$)

```
if $Improve = FALSE$ then
  $SR = SR/2$
  if $SR < 1e - 14$ then
    $SR = (UPPER\_BOUND - LOWER\_BOUND) * 0.4$
  end if
end if
$Improve = FALSE$
for $i = 1$ to $\#Dimensions$ do
  $X_k = X_k[i] - SR$
  if $X_k$ score is better than the current best solution then
    Update current best solution
  end if
  if score of $X_k$ is the same then
    Restore $X_k$ to its original value
  else
    if score of $X_k$ degenerates then
      Restore $X_k$ to its original value
      $X_k[i] = X_k[i] + 0.5 * SR$
      if $X_k$ is better than the current best solution then
        Update current best solution
      end if
      if score of $X_k$ has not been improved then
        restore $X_k$ to its original value
      else
        $Improve = TRUE$
      end if
    else
      $Improve = TRUE$
    end if
  end if
end for
return $SR, Improve$
```

Algorithm 3 DE(Population)

```
for $i = 1$ to $PopulationSize$ do
  $X_1 = Select(Population)$
  $X_2 = Select(Population)$
  $X_3 = Select(Population)$
  for $j = 1$ to $\#Dimensions$ do
    $V_{i,j} = X_{1,j} + F \cdot (X_{2,j} - X_{3,j})$
  end for
  $U = DECrossover(Population[i], V, CR)$
  if $score(U) > score(Population[i])$ then
    $Population[i] = U$
  end if
end for
```

IV. Experimentation

A. Parameter Tuning

The parameters of the algorithm were tuned in a set of previous tests. The values of these parameters were selected based on previous studies and in the literature. Tables I and II display the values that were analyzed. The best combination of values is highlighted on the table.

B. Benchmark Suite

A total of 11 continuous optimization functions have been considered for this experimentation. The first 6 functions were originally proposed for the "*Special Session and Competition on Large Scale Global Optimization*" held at the CEC 2008 Congress [3]. The other 5 functions have been specially proposed for the Workshop on *Evolutionary Algorithms and other Metaheuristics for Continuous Optimization Problems - A Scalability Test* to be held at the ISDA 2009 Conference. They are all completely scalable functions, which makes possible the scalability test proposed for this workshop. A detailed description of the selected benchmark can be found at the web page of the organizers of the workshop[1].

The results reported in this section are the average of 25 independent executions executed on the computer configuration displayed on Table III. For each function, 4 different numbers of dimensions have been tested: $D = 50$, $D = 100$, $D = 200$ and $D = 500$. The maximum number of Fitness Evaluations has been fixed to $5000{*}D$. Due to the constraints of the framework employed, the maximum reachable error without loosing precision is 1e-14. Tables IV, V, VI and VII present the results obtained on the 50, 100, 200 and 500 dimensional functions. In order to analyze the results of the hybrid MDE-DC algorithm, the results of the algorithms which compose the MDE-DC are also presented in the tables.

Table I
DE VALUES THAT WERE USED FOR PARAMETER TUNING

Parameter	Values
Size	20,**25**,30,40,50,60
CR	0.1,**0.5**,0.9
F	0.1,**0.5**,0.9
Crossover Operator	**Exponential**, Binary
Selection Operator	Uniform, **Tournament 2**
Model	**classic**, current to best/2

Table II
LS1 VALUES THAT WERE USED FOR PARAMETER TUNING

Parameter	Values
$localsearch\_freq$	30,40,**50**,60
$max\_localsearch\_iters$	2,3,4,**5**

Table III
COMPUTER CONFIGURATION

PC	Intel Xeon 8 cores 1.86Ghz CPU
Operating System	Ubuntu Linux 8.04
Prog. Language	C++

Table IV
AVERAGE ERROR ON 50-D FUNCTIONS

Function	MDE-DC	DE	LS1
Shifted Sphere	0.0000e+00	2.7040e-02	0.0000e+00
Schwefel's Problem 2.21	9.6161e-12	2.1087e-01	0.0000e+00
Shifted Rosenbrock	4.8624e+00	6.0043e+01	1.1709e+02
Shifted Rastrigin	3.1839e-01	1.0469e-01	0.0000e+00
Shifted Griewank	0.0000e+00	0.0000e+00	7.9886e-03
Shifted Ackley	0.0000e+00	0.0000e+00	0.0000e+00
Schwefel's Problem 2.22	0.0000e+00	0.0000e+00	0.0000e+00
Schwefel's Problem 1.2	6.0076e-01	8.4199e+02	0.0000e+00
Extended f10	0.0000e+00	5.7240e-11	8.5569e+01
Bohachevsky	0.0000e+00	0.0000e+00	0.0000e+00
Schaffer	0.0000e+00	5.7680e-11	8.4234e+01

Table V
AVERAGE ERROR ON 100-D FUNCTIONS

Function	MDE-DC	DE	LS1
Shifted Sphere	0.0000e+00	7.6211e-02	0.0000e+00
Schwefel's Problem 2.21	5.3250e-11	3.0337e+00	0.0000e+00
Shifted Rosenbrock	1.4225e+01	1.2682e+02	4.4314e+02
Shifted Rastrigin	2.7879e-01	4.9106e-02	0.0000e+00
Shifted Griewank	0.0000e+00	3.5153e-02	5.3264e-03
Shifted Ackley	0.0000e+00	7.6019e-02	0.0000e+00
Schwefel's Problem 2.22	0.0000e+00	0.0000e+00	0.0000e+00
Schwefel's Problem 1.2	1.3989e+01	8.3804e+03	2.6699e-03
Extended f10	0.0000e+00	3.1551e-10	1.8471e+02
Bohachevsky	0.0000e+00	0.0000e+00	0.0000e+00
Schaffer	0.0000e+00	3.2785e-10	2.3032e+02

C. Discussion

In order to analyze the results, a comparison with the algorithms which constitute the proposed hybrid (DE and LS1) has been carried out. Table VIII presents the average ranking of the MDE-DC, DE and LS1 algorithms. These rankings represent the mean ranking of the average error of each algorithm on every function. It is clearly shown that the MDE-DC algorithm is always ranked first on each dimension and also in the overall comparison.

Table IX displays the p-$values$ of the comparisons among the algorithms using a non-parametric Wilcoxon test. Even though LS1 is not statistically better than the DE, the MDE-DC algorithm (that hybridizes both approaches) outperforms both algorithms when compared on all dimensions. The significance test also shows that MDE-DC is better than each of the algorithms on certain dimensions.

[1]http://sci2s.ugr.es/programacion/workshop/Scalability.html

Table VI
AVERAGE ERROR ON 200-D FUNCTIONS

Function	MDE-DC	DE	LS1
Shifted Sphere	0.0000e+00	2.0320e-01	0.0000e+00
Schwefel's Problem 2.21	4.8608e-09	1.7212e+01	4.0000e-08
Shifted Rosenbrock	2.8950e+01	2.5392e+02	1.7237e+02
Shifted Rastrigin	1.6796e-01	4.9346e-02	0.0000e+00
Shifted Griewank	0.0000e+00	3.9092e-03	7.2006e-03
Shifted Ackley	0.0000e+00	3.3032e-02	0.0000e+00
Schwefel's Problem 2.22	0.0000e+00	0.0000e+00	0.0000e+00
Schwefel's Problem 1.2	2.7426e+02	4.4700e+04	1.5465e+01
Extended f10	3.0987e-07	1.2955e-09	4.2116e+02
Bohachevsky	0.0000e+00	0.0000e+00	0.0000e+00
Schaffer	3.9111e-07	1.3451e-09	4.1816e+02

Table VII
AVERAGE ERROR ON 500-D FUNCTIONS

Function	MDE-DC	DE	LS1
Shifted Sphere	0.0000e+00	2.8648e-01	0.0000e+00
Schwefel's Problem 2.21	3.3906e-04	5.0182e+01	6.1200e-06
Shifted Rosenbrock	2.9614e+01	7.0482e+02	2.7538e+02
Shifted Rastrigin	0.0000e+00	3.7446e-01	0.0000e+00
Shifted Griewank	0.0000e+00	3.2445e-02	6.8063e-03
Shifted Ackley	0.0000e+00	5.6647e-02	0.0000e+00
Schwefel's Problem 2.22	0.0000e+00	0.0000e+00	0.0000e+00
Schwefel's Problem 1.2	1.0958e+04	2.8707e+05	6.6316e+03
Extended f10	3.2455e+00	6.2471e-09	9.7798e+02
Bohachevsky	0.0000e+00	0.0000e+00	0.0000e+00
Schaffer	3.2687e+00	6.4134e-09	9.7745e+02

V. CONCLUSIONS

In the last years, several metaheuristic algorithms have shown a great performance in low-dimensionality continuous optimization problems. However, most of these algorithms get their performance reduced as the dimensionality of the problems grows. For this reason, it is important to analyze which factors help an algorithm to scale well, which is the objective of this workshop.

In this contribution, a hybrid memetic evolutionary algorithm is presented and its performance on problems of different sizes is studied. The hybridization with a exploita-

Table VIII
RANKINGS

Dimension	MDE-DC	DE	LS1
50	1.7272	2.2272	2.0454
100	1.6363	2.4545	1.9090
200	1.7272	2.2727	2.0000
500	1.6818	2.4545	1.8636
all	**1.6931**	**2.3522**	**1.9545**

Table IX
WILCOXON TEST P-VALUES

Dimension	MDE-DCvsDE		MDE-DCvsLS1		LS1vsDE
50	0.0640		0.1552		0.5832
100	0.0253	√	0.1552		0.4528
200	0.0429	√	0.1552		0.2386
500	0.0865		0.2315		0.2035
all	**0.0006**	√	**0.0288**	√	**0.1904**

√ represents that the $p\text{-}value$ is $\alpha = 0.05$ significant

tive local search helps the DE to avoid the stagnation. On the other hand, the DE allows the local search to find promising regions with a moderate consumption of fitness evaluations. The results from this study suggest that this memetic approach is appropiate to deal with problems of different dimensionality.

As a further research line, the inclusion of a dynamic self-adaptive approach is under consideration. This approach would provide fine tuning of the decisions whether the local search should be performed. In an extended study, it is planned to include the effects of a non-uniform initialization and a comparative evaluation of the different MTS local searchers.

ACKNOWLEDGMENT

The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa) and the Spanish Supercomputing Network. This work was supported by the Madrid Regional Education Ministry and the European Social Fund and financed by the Spanish Ministry of Science TIN2007- 67148.

REFERENCES

[1] A. Auger and N. Hansen, “A restart cma evolution strategy with increasing population size,” in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005*. IEEE Press, 2005, pp. 1769–1776.

[2] L. Tseng and C. Chen, “Multiple trajectory search for large scale global optimization,” in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 3052–3059.

[3] K. Tang, X. Yao, P. Suganthan, C. MacNish, Y. Chen, C. Chen, and Z. Yang, “Benchmark functions for the cec 2008 special session and competition on large scale global optimization,” Nature Inspired Computation and Applications Laboratory, USTC, Tech. Rep., 2007.

[4] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, “Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization,” 1School of EEE, Nanyang Technological University and Kanpur Genetic Algorithms Laboratory (KanGAL), Tech. Rep. 2005005, May 2005.

[5] A. Qin and P. Suganthan, “Self-adaptive differential evolution algorithm for numerical optimization,” in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005*. IEEE Press, 2005, pp. 1785–1791.

[6] J. Rönkkönen, S. Kukkonen, and K. Price, “Real-parameter optimization with differential evolution,” in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005*. IEEE Press, 2005, pp. 506–513.

[7] Y. Gao and Y.-J. Wang, “A memetic differential evolutionary algorithm for high dimensional functions’ optimization,” in *Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*. IEEE Press, 2007, pp. 188–192.

[8] V. Tirronen, F. Neri, T. Karkkainen, K. Majava, and T. Rossi, “A memetic differential evolution in filter design for defect detection in paper production,” in *Proceedings of EvoWorkshops 2007*. Springer-Verlag, 2007, pp. 330–339.

[9] A. Caponio, F. Neri, and V. Tirronen, “Super-fit control adaptation in memetic differential evolution frameworks,” *Soft Computing*, vol. 13, no. 8–9, pp. 811–831, 2009.

[10] A. Caponio, G. Cascella, F. Neri, N. Salvatore, and M. Sumner, “A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives,” *IEEE Transactions On Systems, Man and Cybernetics - Part B*, vol. 37, pp. 28–41, 2007.

[11] A. Caponio, F. Neri, G. Cascella, and N. Salvatore, “Application of memetic differential evolution frameworks to PMSM drive design,” in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence)*, 2008, pp. 2113–2120.

[12] B. Subudhi and D. J. amd M.M. Gupta, “Memetic differential evolution trained neural networks for nonlinear system identification,” in *Proceedings of the Third International Conference on Industrial and Information Systems*, 2008, pp. 1–6.

[13] J. Lampinen and I. Zelinka, “On stagnation of the differential evolution algorithm,” in *Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing*, 2000, pp. 76–83.