Elitistic Evolution: a Novel Micro-Population Approach for global optimization problems

Francisco Viveros-Jiménez
Universidad del Istmo Campus Ixtepec
Cd. Ixtepec, Oaxaca, México
Email: pacovj@hotmail.com

Efren Mezura-Montes
Laboratorio Nacional de
Informática Avanzada (LANIA A.C.)
Xalapa, Veracruz, México
Email: emezura@lania.mx

Alexander Gelbukh
Centro de Investigación en Computación
Instituto Politécnico Nacional
D.F. México
Email: gelbukh@gelbukh.com

Abstract—**Micro-population Evolutionary Algorithms (μ-EAs) are useful tools for optimization purposes. They can be used as optimizers for unconstrained, constraint and multi-objective problems. μ-EAs distinctive feature is the usage of very small populations. A novel μ-EA named Elitistic Evolution (EEv) is proposed in this paper. EEv is designed to solve high-dimensionality problems ($N \geq 30$) without using complex mechanisms e.g. Hessian or covariance matrix. It is a simple heuristic that does not require a careful fine-tunning of its parameters. EEv principal features are: adaptive behavior and elitism. Its evolutionary operators: mutation, crossover and replacement, have the ability to search either locally (near a current point) or globally (on a distant point). This ability is controlled by a single adaptive parameter. EEv is tested on a set of well-known optimization problems and its performance is compared with respect to state-of-the-art algorithms, such as Differential Evolution, μ-PSO and Restart CMA-ES.**

Index Terms—**Optimization methods, Micro-population algorithms, Evolutionary Computation.**

I. INTRODUCTION

Due to the necessity of a simple, fast and robust optimizer, several approaches have been proposed in the recent years. Evolutionary Algorithms (**EA**s) are efficient heuristics that accomplish this necessity [1]. EAs find solutions by emulating natural evolution. In this way, EAs *evolve* a population of candidate solutions in order to improve them. It is a well-known fact that EAs usually utilize large populations. Some researchers have implemented EAs which can work with small populations [7], [8]. They were called *Micro-population Evolutionary Algorithms* (μ-**EA**s).

A μ-EA is an EA which evolves a small population ($P < 10$). From the specialized literature, two representative μ-EAs are (1) the micro-Genetic Algorithm (μ-**GA**) [7] and (2) the micro-Particle Swarm Optimization (μ-**PSO**) [9]. μ-EAs can be used as optimizers for unconstrained [7], constrained [9] and multi-objective optimization problems[10]. Aditionally, μ-EAs can be used either as Local Improvement Processes (**LIPs**) to create efficient memetic algorithms [11] or as part of cooperative evolutionary algorithms [13].

In this work we propose a novel but simple μ-EA to solve unconstrained optimization problems, called Elitistic Evolution (**EEv**). EEv was created to solve high-dimensionality problems ($N \geq 30$) without using additional mechanisms to guide the search, such as a Hessian or a covariance matrix. The variation operators used in EEv are mutation and crossover. Furthermore, a replacement process takes place. As a combined effect, EEv has the ability to search either locally (near a current point) or globally (on a distant point). This behavior is controlled by a single adaptive parameter. Thus, EEv will conduct the search according to the current situation of the optimization process. These features make EEv competitive on solving global optimization problems. We test EEv in order to analyze its robustness, speed and average performance.

The contents of this paper are organized as follows: First, we describe EEv and the proposed evolutionary operators in Section II. After that, Section III contains the experimental design and obtained by each compared technique. Finally, section IV concludes this paper.

II. ELITISTIC EVOLUTION

The main features of EEv are:

1) Two variaton operators: mutation and crossover
2) A replacement mechanishm which sorts the individuals (solutions) based on fitness, in such a way that the first solution is the fittest.
3) Two user-defined parameters: population size ($P \in \mathbb{N}, P \in [3, 10]$) and the initial (base) value for a set of stepsizes ($B \in \mathbb{R}, B \in [0.0, 1.0]$).
4) A hill-climbing-like mutation operator.

EEv has the ability to search either locally (near a current point) or globally (on a distant point) according to the success of the optimization process. This ability is implemented through an adaptive parameter (termed $C \in \mathbb{N}, C \in [1, P]$) and a set of special evolutionary operators. The C value indicates the number of individuals to be affected by a local search process. In this way, lower C values promote global exploration while higher C values promote local exploitation. Table I illustrates the effects of C parameter over the search.

Figure 1 describes the EEv optimization process. EEv approach optimizes through evaluations, instead of optimizing through populations as in regular EAs. To avoid premature convergence EEv maintains diversity through the selection of $P - C$ random offspring in the replacement stage.

1 Set $X_i^0, i = 1, \ldots, P$ as a random population;
2 Evaluate each X_i^0;
3 C=1;
4 Set adaptive step sizes $\vec{b_j} = B, j = 1, \ldots, N$;
5 **for** *g=1 To G* **do**
6   Each solution $X_i^g, i = 1 \ldots, P$ will generate a mutant $O_i^{'g}, i = 1, \ldots, P$ by mutation (see Figure 2);
7   Use the three-individual crossover operator to generate P offspring O_i^g (see Figure 7). The parents will be selected from $X_i^g \cup O_i^{'g}$;
8   Evaluate each offspring $O_i^g, i = 1, \ldots, P$;
9   The new population $X_i^{g+1}, i = 1, \ldots, P$ consists on the C best individuals from $X_i^g \cup O_i^g$ and $P - C$ individuals randomly selected from O_i^g;
10 C and $\vec{b}$ values are updated (see Figures 3 and 9);

Fig. 1. Algorithm for EEv. G is the maximum generation number.

TABLE I
IMPLICATIONS OF C PARAMETER.

Stage	$C = 1$	$C = P$
Mutation	Global search	Local Search
	Dynamic stepsizes	Adaptive stepsizes
	A few significant	Many slight
	changes to variables	changes to variables
Crossover	Random parent selection	Elite as main parent
Restart	Total restart	No restart

A. Mutation operator

The mutation operator used in EEv is based on the mutation technique proposed in [4]. This operator provides a balance between exploration and exploitation. Figure 4 illustrates the general idea of the operator: In Figure 4 (left) exploration is promoted because a few solutions use small stepsizes while the remaining ones use large stepsizes (see equation 1). The opposite occurs in Figure 4 (right) where all solutions use small stepsizes to promote local exploitation. Figure 2 describes the mutation operator, which performs a random number of alterations in some variables on each individual X_i^g. The search space can be sampled by moving in all dimensions of the search space as seen in Figure 5.

By analyzing Equation 1 more in depth, this is the mechanism to bias that mutation operator to use the $\vec{b}$ step sizes for the first C individuals (exploitation). The adaptive stepsizes of all dimensions are stored in $\vec{b}$. $\vec{b}$ values change depending on the previous success of the search, determined by the comparison of the best fitness values of the current and previous generations. Figure 3 shows the update process for $\vec{b}$, which has the following behaviors:

- When the population maintains the same best individual, $\vec{b}$ values become smaller in an effort to improve the current solution.
- When the population updates its best individual, $\vec{b}$ values are adapted according to the current situation.
- When a $\vec{b_j}$ value reaches a zero value, it is restarted to

1 $alterations = rand(\lceil N \times (C/P) \rceil, N)$;
2 **for** *all alterations* **do**
3   Select a random k dimension;
4   Calculate M with equation 1 ;
5 $O_{i,j}^g = O_{i,j}^g + (up_j - low_j) \times Rand(-M, M)$;

Fig. 2. Mutation algorithm for an i individual. $i = 1, \ldots, P$. $j = 1, \ldots, N$. rand(L,U) returns a random integer value within L and U. Rand(L,U) returns a random real value within L and U. up_j is the upper bound for the j dimension. low_j is the lower bound for the j dimension.

1 **if** $F(X_{best}^{g-1}) > F(X_{best}^g)$ **then**
2 $\vec{b_j} = \left| \left(X_{best,j}^{g-1} - X_{best,j}^g \right) / (up_j - low_j) \right|$;
3 **else**
4 $\vec{b_j} = Rand(0.0, \vec{b_j})$;
5 **if** *a* $\vec{b_j}$ *value is equal to 0* **then**
6   Replace it with $B \times (1.0 - Rand(0.0, 1.0) \times g/G)$;

Fig. 3. Recalculation of $\vec{b}$. g is the current generation. G is the max generation. $j = 1, \ldots, N$.

maintain the movility of the solutions.

Equation 1 also allows the mutation operator to use the B stepsize value for the remaining $P - C$ individuals (global exploration). The B factor is affected by the generation number. Later iterations will imply smaller stepsizes. This will promote local exploitation in later stages and global exploration in the earlier ones.

$$M = \begin{cases} \vec{b_j} & i \leq C \\ B \times (1.0 - Rand(0.0, 1.0) \times g/G) & i > C \end{cases} \quad (1)$$

B. Crossover operator

The crossover operator requires 3 individuals to generate an offspring: 2 mutant individuals ($O_k^{'g}$ and $O_m^{'g}$) and an individual from the current population (X_l^g). The offspring O_i^g will be allocated between its parents as seen in Figure 8. New offspring can be selected as parents for the remaining offspring individuals.

The C parameter also affects the selection of the $O_k^{'g}$ and X_l^g individuals. $O_k^{'g}$ is the first mutant individual selected as parent and X_l^g is the individual from the current population which will be used as parent as well. When $C = 1$, any individual can be selected; when $C = P$, elitism is ensured, promoting exploration around the best individual. Figure 6 illustrates the effects of C in the crossover operator.

C. Recalculation of C Adaptive Parameter

The C value changes depending on the search success in the last generation, determined by the comparison of the best fitness values of the current and previous generations. Figure 9 shows the recalculation of C parameter. If a "better" best result was found, then C is decreased, encouraging global exploration. Otherwise, C is increased to encourage local exploitation. C has the following behaviors:
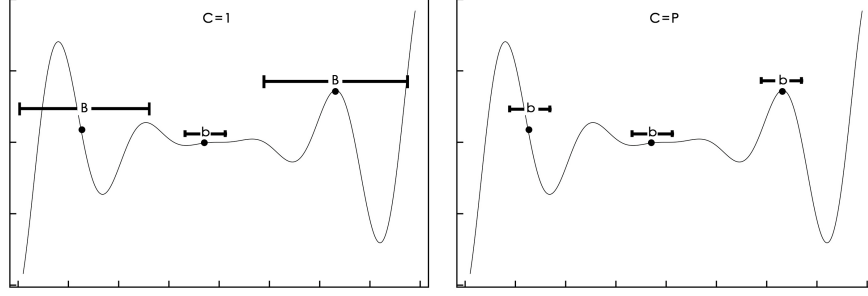
Fig. 4. Two different scenarios for mutation operator: $C = 1$ (left) and $C = P$ (right). The fittest C offspring perform local exploitation using $\vec{b}$ steps and the remaining ones perform global exploration using B steps.
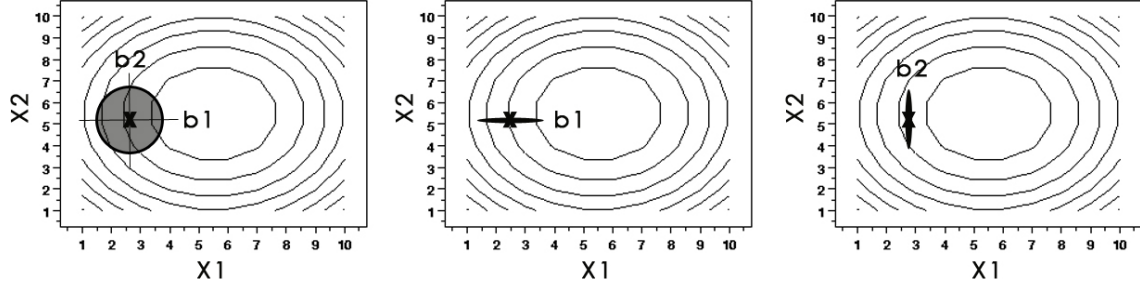
Fig. 5. Mutation operator can explore in all dimensions of the search space.
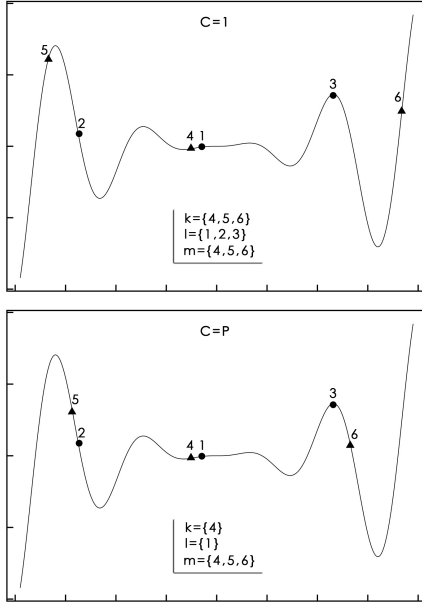
Fig. 6. Two different scenarios for crossover operator: $C = 1$ (up) and $C = P$ (down). C parameter affects the selection frames of the k and l individuals. k is the first mutant selected as parent. l is the population individual selected as parent. Dots represent population individuals and triangles represent mutants.

- When the population converges, $C = P$ to improve the elite individual with more precision.
- When the population has successive improvements, $C = 1$, encouraging the exploration of new search areas.

```
1 for i = 1 To P do
2    c_1 = Rand(0.0, 1.0);
3    c_2 = Rand(0.0, 1.0 - c_1);
4    c_3 = 1.0 - c_2 - c_1;
5    O_i^g = c_1 × O_rand(1,P-C+1)^g + c_2 ×
         X_rand(1,P-C+1)^g + c_3 × O_rand(1,P)^g;
```

Fig. 7. Crossover operator algorithm.
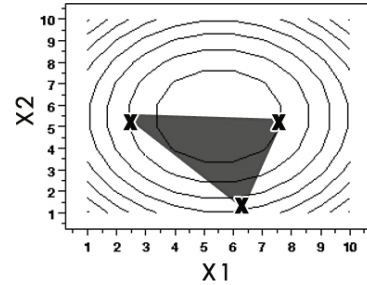
Fig. 8. The new offspring individual will be allocated in an area between its three parents.

III. EXPERIMENTS

The experiments aim to confirm that EEv is competitive against other EAs and also look to observe the differences against approaches which use other information such as a Hessian or covariance matrix. We measure the Error and Evaluation values for each trial in a similar way to the one proposed in the test suite for CEC 2005 special session on real-

TABLE II
TEST FUNCTIONS

Unimodal functions	
Separable	
f_{sph}	Sphere model
$f_{2.22}$	Schwefel's problem 2.22
$f_{2.21}$	Schwefel's problem 2.21
f_{stp}	Step function
f_{qtc}	Quartic function with noise
Non-separable	
$f_{1.2}$	Schwefel's problem 1.2
Multimodal functions	
Separable	
f_{sch}	Generalized Schwefel's problem 2.26
f_{ras}	Generalized Rastrigin's function
Non-separable	
f_{ros}	Generalized Rosenbrock's function
f_{ack}	Ackley's function
f_{grw}	Generalized Griewank's function
f_{sal}	Salomon's function
f_{whi}	Whitley's function
$f_{pen1,2}$	Generalized penalized functions

1 **if** $F(X_{best}^{g-1}) > F(X_{best}^g)$ **then**
2   **if** $C > 1$ **then**
3      $C = C - 1$;
4 **else**
5   **if** $C < P$ **then**
6      $C = C + 1$;

Fig. 9. Recalculation of C. X_{best}^g is the elite individual. $F(X_{best}^0) = F(X_{best}^1)$.

parameter optimization [6]. The benchmark functions [3] are specified in table II. We conducted 30 trials per test function. Error is equal to $(F(x^o) - F(x^*))$ where x^o is the best reported solution for the corrresponding algorithm and x^* is the global optimum value. Evaluation value is the number of function evaluations (**FE**s) required to reach an Error value of 10^{-8}. Furthermore, we measure the number of successful trials that reach the target accuracy value. N is the dimensionality of the test function. The stop condition criterion of each run was $10,000 \times N$ function evaluations (**FE**s). Due to the space limitation on this paper, we show a comparison between EEv and three state-of-the-art approaches. The selected approaches are:

- DE/rand/1/bin (**DE**) selected because it is a well-known EA [2].
- μ-PSO selected because it is a competitive micro-population approach [9].
- Restart CMA-ES [12] selected for measuring the gap against a technique that uses Hessian and covariance matrices. Also, it was best technique on CEC 2005 special session on real-parameter optimization.

All the experiments were performed using a Pentium 4 PC with 512 MB of RAM, in C language over a Linux environment. The parameter sets for the techniques were:

1) **EEv**: $P = 5, B = 0.6$.
2) **DE**: $P = N, CR = 0.9, F = 0.9$, based on [5].
3) μ-**PSO**: $P = 6, C_1 = C_2 = 1.8, Neighborhoods = 2$, Replacement generation $= 100$, Replacement particles$= 2$, Mutation $\% = 0.1$, based on [9].
4) **Restart CMA-ES**: set as in [12].

A. Performance evaluation

We performed a comparison of EEv against DE, μ-PSO and Restart CMA-ES. Table III shows the mean Error values and the number of successful trials (trials where the technique reach the target Error value) obtained on the benchmark functions with $N = 30, 50, 100$. Table IV shows the mean Evaluation values required to reach the target Error value. Tests showed that:

1) EEv outperformed DE and μ-PSO.
2) The better performance of EEv over DE and μ-PSO is more significant in functions with $N = 50$ and $N = 100$.
3) EEv found global optimum values on 10 out of 15 functions, equal to CMA-ES. It found more global optimum values than μ-PSO and DE.
4) EEv required less FEs to find global optimum values than DE and μ-PSO.
5) EEv maintained its performance on problems with $N \geq 50$ like μ-PSO.

We confirmed that EEv has a competitive performance in global optimization problems with a high dimensionality by requiring the fine-tuning of just two parameters. However, EEv is still surpassed by CMA-ES. This fact is a confirmation of the efficiency of using extra information to conduct the search.

B. Analysis of EEv's behavior

The results obtained suggested that EEv searched locally most of the time (see Table V). This behavior has two meanings: (1) EEv performed few significant improvements to population; and (2) EEv used most of the time for searching on nonpromising regions.

We detected some deficiencies on the crossover operator. Crossover operator has a similar behavior to bisection and false position methods for finding equation roots values: the solution has to be enclosed between the reference values. The parent selection on EEv is a random mechanism so the proper parent selections depends on the mutation operator exploration capabilities and on randomness.

We identified two different failure scenarios: (1) Max FEs were not enough to reach the target Error value; and (2) premature convergence was reached. The main cause of these scenarios was the reduction of global exploration capabilities in the last stages. This means that if EEv did not find the optimal value region on time, it got stuck over a local minimum value.

IV. CONCLUSIONS AND FUTURE WORK

This paper described a novel evolutionary method called Elitistic Evolution. EEv is a population-based technique which

TABLE III

MEAN ERROR VALUES OBTAINED ON FUNCTIONS WITH $N = 30, 50, 100$. A 0.0 VALUE MEANS THAT 10^{-8} WAS REACHED IN ALL RUNS (100% SUCCESS RATE). ON VALUES LIKE X.XXE+X(Y) Y REPRESENT THE NUMBER OF SUCCESSFUL TRIALS (ONLY WHEN $Y > 0$).

30	EEv	DE	μ-PSO	CMA-ES
f_{sph}	0.0	0.0	0.0	0.0
$f_{2.22}$	0.0	0.0	0.0	0.0
$f_{2.21}$	1.26E-2	1.41E+1	9.53E-2	0.0
f_{stp}	0.0	3.33E-2(28)	0.0	0.0
f_{qtc}	3.94E-3	1.63E-2	1.69E-2	3.89E-2
$f_{1.2}$	5.60E-3	1.12E-1	1.99E-1	0.0
f_{sch}	5.53E+3	1.38E+2	1.58E+3	1.24E+4
f_{ras}	0.0	2.53E+1	1.30E+1	7.27E+0(3)
f_{ros}	1.49E+1(4)	2.15E+0	5.98E+1	2.54E-3
f_{ack}	0.0	0.0	0.0	0.0
f_{grw}	3.44E-3(5)	3.44E-3(22)	3.84E-2(5)	0.0
f_{pen1}	0.0	1.03E-2(27)	0.0	0.0
f_{pen2}	0.0	7.32E-4(28)	0.0	0.0
f_{sal}	6.39E-1	2.48E-1	4.93E-1	2.04E-1
f_{whit}	1.60E+1(8)	3.37E+2	3.58E+2	4.93E+2
50	EEv	DE	μ-PSO	CMA-ES
f_{sph}	0.0	1.31E-2	0.0	0.0
$f_{2.22}$	0.0	3.36E-2	0.0	0.0
$f_{2.21}$	4.38E-2	2.11E+1	4.20E-1	0.0
f_{stp}	0.0	4.33E-1(20)	0.0	0.0
f_{qtc}	5.17E-3	6.68E-2	4.05E-2	9.14E-2
$f_{1.2}$	2.32E-1	4.53E+4	6.49E+0	0.0
f_{sch}	2.75E+3	6.66E+3	3.28E+3	2.07E+4
f_{ras}	0.0	9.32E+1	2.55E+1	2.54E+1
f_{ros}	4.32E+1(1)	3.78E+1	5.98E+1	1.46E-3
f_{ack}	0.0	6.90E-2	1.23E-8(25)	0.0
f_{grw}	1.29E-2(12)	1.41E-2	2.24E-2(11)	0.0
f_{pen1}	0.0	6.90E-2	0.0	0.0
f_{pen2}	0.0	3.81E-1	0.0	7.32E-4(28)
f_{sal}	9.99E-1	1.15E+0	8.46E-1	2.99E-1
f_{whit}	6.87E+1(3)	1.58E+5	6.82E+2	1.18E+3
100	EEv	DE	μ-PSO	CMA-ES
f_{sph}	0.0	4.59E+3	0.0	0.0
$f_{2.22}$	3.40E-6	5.64E+1	0.0	0.0
$f_{2.21}$	2.69E-1	1.41E+1	6.27E+1	5.94E-3(1)
f_{stp}	0.0	3.74E+3	1.00E-1(27)	0.0
f_{qtc}	8.47E-3	3.52E+0	1.28E-1	2.17E-1
$f_{1.2}$	1.43E+1	2.45E+5	2.45E+2	0.0
f_{sch}	5.53E+3	2.86E+4	8.38E+3	4.15E+1
f_{ras}	0.0	9.05E+2	5.37E+1	5.67E+1
f_{ros}	5.52E+1	2.41E+6	1.39E+2	7.74E-4
f_{ack}	3.53E-7	8.67E+0	2.24E-7(2)	0.0
f_{grw}	6.47E-3(17)	4.23E+1	1.11E-2(15)	0.0
f_{pen1}	0.0	4.36E+5	0.0	0.0
f_{pen2}	0.0	2.73E+6	0.0	1.79E-3(26)
f_{sal}	1.67E+0	9.45E+0	1.61E+0	1.04E+0
f_{whit}	2.39E+2(1)	4.35E+15	2.36E+3	8.95E+3

TABLE IV

AVERAGE FEs REQUIRED IN SUCCESS RUNS.

30	EEv	DE	μ-PSO	CMA-ES
f_{sph}	1.77E+4	1.89E+5	1.08E+5	3.30E+3
$f_{2.22}$	8.11E+4	2.72E+5	1.73E+5	7.80E+3
$f_{2.21}$	–	–	–	1.15E+3
f_{stp}	3.17E+4	7.40E+4	7.88E+4	3.34E+2
$f_{1.2}$	–	–	–	3.60E+4
f_{ras}	9.20E+4	–	–	2.28E+5
f_{ros}	1.53E+5	–	–	–
f_{ack}	8.96E+4	2.86E+5	2.52E+5	6.46E+3
f_{grw}	4.53E+4	1.93E+5	1.25E+5	3.99E+3
f_{pen1}	3.14E+4	1.75E+5	1.11E+5	5.20E+3
f_{pen2}	3.50E+4	1.93E+5	1.12E+5	7.60E+3
f_{whit}	1.91E+5	–	–	–
50	EEv	DE	μ-PSO	CMA-ES
f_{sph}	6.82E+4	–	1.96E+5	5.20E+3
$f_{2.22}$	2.72E+5	–	3.14E+5	1.31E+4
$f_{2.21}$	–	–	–	2.44E+4
f_{stp}	5.09E+4	3.72E+5	1.67E+5	9.02E+2
$f_{1.2}$	–	–	–	9.51E+4
f_{ras}	1.87E+5	–	–	–
f_{ros}	1.75E+5	–	–	–
f_{ack}	2.09E+5	–	4.54E+5	9.85E+3
f_{grw}	7.42E+4	–	2.01E+5	7.10E+3
f_{pen1}	5.49E+4	–	02.03E+5	8.19E+3
f_{pen2}	6.37E+4	–	2.02E+5	1.05E+4
f_{whit}	3.96E+5	–	–	–
100	EEv	DE	μ-PSO	CMA-ES
f_{sph}	1.48E+5	–	4.07E+5	9.90E+3
$f_{2.22}$	–	–	7.26E+5	5.08E+4
$f_{2.21}$	–	–	–	8.60E+4
f_{stp}	1.13E+5	–	5.36E+5	4.12E+3
$f_{1.2}$	–	–	–	4.07E+5
f_{ras}	5.81E+5	–	–	–
f_{ack}	–	–	9.61E+5	1.80E+4
f_{grw}	1.47E+5	–	4.27E+5	1.56E+4
f_{pen1}	1.21E+5	–	4.27E+5	1.38E+4
f_{pen2}	1.36E+5	–	4.25E+5	5.31E+4
f_{whit}	8.94E+5	–	–	–

TABLE V

C PARAMETER AVERAGE FREQUENCY. THE TABLE CONTAINS THE RELATIVE FREQUENCY OF THE C VALUES AFTER THE G GENERATION TIME. A 100% VALUE MEANS THAT AN SPECIFIC VALUE WAS USED ON EVERY GENERATION OF THE TRIAL.

	$C = 1$	$C \in [2, P-1]$	$C = P$
Unimodal			
Separable	0.03%	3.97	96.01 %
Non-separable	0.03%	9.82%	90.16%
Multimodal			
Separable	0.01%	3.86%	96.13 %
Non-separable	0.02%	3.66%	96.32%

works better with small populations $P \leq 5$. EEv was created to solve high-dimensionality optimization problems ($N \geq 30$) without using complex mechanisms such as Hessian or covariance matrix. Instead, just two paramenters must be calibrated by the user: population size (P) and base step size (B).

EEv solved 15 well-known benchmark functions and its results were compared with those obtained by state-of-the-art techniques. We confirmed that EEv is competitive against evolutionary algorithms, but its performance was not better with respect to techniques which uses extra information such as the Restart CMA-ES. EEv performed well in most of the test cases and outpferformed, mostly in problems with a high dimensionality ($N \geq 30$), DE/rand/1/bin and μ-PSO by requiring less FEs. However, premature convergence was

detected when C got stuck in the P value before $\frac{1}{8}$ of the FEs period of time. More comparative studies and further analysis should be carried out to provide a more detailed understanding of EEv. We also plan to test EEv in constrained and in multi-objective optimizaton problems.

ACKNOWLEDGMENTS

REFERENCES

[1] Eiben, A.E.,Smith Smith, J.E.: Introduction to Evolutionary Computing. Springer. (2003)

[2] Storn, R., Price, K.: Differential Evolution - a simple and efficient heuristic for global optimization. Journal of Global Optimization, Volume 11, Number 4, Springer Netherlands. (1997) 341–359.

[3] Mezura-Montes, E., Coello, C CA., Velazquez, R J.: A comparative study of differential evolution variants for global optimization. Proceedings of the 8th annual conference on Genetic and evolutionary computation. (2006) 485–492.

[4] Viveros, J F.: DSE: A Hybrid Evolutionary Algorithm with Mathematical Search Method. Special issue journal Research in Computing Science. RCS. (2008)

[5] Noman, N., Iba, H.: Accelerating Differential Evolution Using an Adaptive Local Search. IEEE Transactions on Evol. Comput. Vol. 12, No. 1 IEEE Press. (2008) 107–125

[6] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y-P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Obtimization. Nanyang Technol. Univ., Singaporem IIT Kanpur, India, KanGal Rep. 2005005. (2005)

[7] Krishnakumar, K.: Micro-genetic algorithms for stationary and non-stationary function optimization. SPIE: Intelligent control and adaptive systems, 1196. (1989) 289–296.

[8] Goldberg, D-E.: Sizing Populations for Serial and Parallel Genetic Algorithms. Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kauffman Publishers. (1989) 70–79

[9] Fuentes-Cabrera, J-C., Coello-Coello, C-A.: Handling Constraints in Particle Swarm Optimization using a Small Population Size. LNCS. MICAI 2007: Advances in Artificial Intelligence. Springer-Verlag. vol 4827. (2007) 41–51

[10] Toscano-Pulido, G., Coello-Coello, C-A.: Multiobjective Optimization using a Micro-Genetic Algorithm. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001). Springer-Verlag. (2001) 126–140

[11] Kazarlis, S.E., Papadakis, S.E., Theocharis, J.B., Petridis, V.: Micro-genetic Algorithms as Generalized Hill-Climbing Operators for GA Optimization. Evol. Comput., vol 5, no. 3. IEEE Press. (2001) 204–217

[12] Auger, A., Kern, S., Hansen, N.: A Restart CMA Evolution Strategy with Increasing Population Size. CEC 2005 Special Session on Real-Parameter Obtimization. Nanyang Technol. Univ., Singaporem IIT Kanpur, India (2005)

[13] Parsopoulos, K.E.: Cooperative Micro-Particle Swarm Optimization. ACM 2009 World Summit on Genetic and Evolutionary Computation (2009 GEC Summit), Shanghai, China. ACM. (2009) 467–474