Application Domain Study of Evolutionary Algorithms in Optimization Problems

P. Caamaño F. Bellas J.A. Becerra R.J. Duro

Grupo Integrado de Ingeniería
Escola Politécnica Superior
Universidade da Coruña
Spain

pcsobrino@udc.es fran@udc.es ronin@udc.es richard@udc.es

ABSTRACT

This paper deals with the problem of comparing and testing evolutionary algorithms, that is, the benchmarking problem, from an analysis point of view. A practical study of the application domain of four representative evolutionary algorithms is carried out using a relevant set of real-parameter function optimization benchmarks. The four selected algorithms are the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and the Differential Evolution (DE), due to their successful results in recent studies, a Genetic Algorithm with real parameter operators, used here as a reference approach because it is probably the most familiar to researchers, and the Macroevolutionary algorithm (MA), which is not widely known but it shows a very remarkable behavior in some problems. The algorithms have been compared running several tests over the benchmark function set to analyze their capabilities from a practical point of view, in other words, in terms of their usability. The characterization of the algorithms is based on accuracy, stability and time consumption parameters thus establishing their operational scope and the type of optimization problems they are more suitable for.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence – *Problem Solving, Control Methods, and Search*

General Terms

Algorithms, Measurement, Performance, Experimentation.

Keywords

Evolutionary Algorithms, Algorithm Characterization, Optimization Benchmarks, Comparison Tests, Error Measures.

1. INTRODUCTION

The present work takes inspiration from the ever more frequent algorithm competitions in the field of Evolutionary Computation. These competitions are typically held during conferences in this area like CEC or GECCO, and the main objective of the organizers is to provide common benchmarks in order to fairly compare the large number of different algorithms that have been recently developed. Typically, when a new algorithm is presented in the literature, it is tested with problems that are chosen to highlight its capabilities. As a consequence, this kind of competitions are needed to evaluate the algorithms in a more systematic manner by specifying a common termination criterion, problem size, initialization scheme, linkages/rotation, etc [1][2][3].

The main problem we have found when analyzing the results of these general competitions and comparison papers [4], is that they don't provide useful conclusions or, in many cases, the appropriate data to characterize the algorithms from a practical point of view. Although typical optimization benchmarks classify functions depending on their features (separable, non-separable, continuous, discontinuous, etc), authors don't provide general conclusions about the type of function that is more suitable for each type of algorithm. In this paper, we try to bridge this gap between the theoretical studies and the practical optimization, by applying error and performance measures that permit characterizing the algorithms according to the type of function they perform better at, and the computational cost they imply.

The typical error measures established to compare the algorithms are related to the number of samples of the objective function necessary to achieve the optimum or to the mean error with respect to the optimal value after a fixed number of runs [5]. These are very relevant measures of the quality of the algorithm from a phenotypic (operational) point of view, but they don't provide any indication about how different the genotype (the encoding) of the solution is with respect to the optimal. In practical terms, this is very useful information because solutions that are successful from a phenotypic point of view (near the optimum function value) may be far from the optimum point. In this work, we propose a new error measure to compare the algorithms from this perspective.

The rest of the paper is structured as follows: section 2 presents in detail all of the elements involved in the tests that have been carried out. Section 3 contains the results obtained and their discussion and section 4 is devoted to the conclusions of this analysis from a practical point of view.

2. EXPERIMENTAL SETUP

This section is concerned with the presentation of all the elements involved in the comparisons we have carried out.

GECCO'08, July 12–26, 2008, Atlanta, Georgia, USA.

2.1 Benchmark function set

In comparison tests, the selection of a general and relevant benchmark function set is very important. In this case, 23 real-parameter optimization functions of different dimensionality were selected. This benchmark set was used by Yao in [6] and a modification of it in CEC 2005 competition [7]. The set has been divided into two main subsets based on the dimensionality of the benchmarks. Functions f_1-f_{13} represent high-dimensional problems while the remaining ten functions (f_{14}-f_{23}) could be considered low-dimensional problems. This function set is widely used on testing evolutionary algorithms because it contains functions with different features like dimensionality, separability, modality and continuity. Table 1 summarizes the benchmark function set and their features. Formal expression of these functions can be found in [6].

To study the application domain of the algorithms, the function set was grouped in terms of their mathematical characteristics into three main categories:

2.1.1 Unimodal vs. Multimodal

Functions f_1-f_4 and f_6-f_7 are unimodal functions, These functions do not present local minima and they only contain one global minimum. On the other hand, functions f_5 and f_8-f_{13} are multimodal functions, that is, they present more than one (several) local minima that increase exponentially with dimensionality. Some authors consider f_5 a unimodal function when working in 2 dimensions. In this paper it will be used with higher dimensionalities, so f_5 will be taken as multimodal [8].

Low-dimensional functions are all multimodal, functions f_{15} and f_{17} have no local minima and each one has three or four global minima. Function f_{16} presents four local minima and two global minima. The rest of the functions contain several local minima and only one global minimum.

2.1.2 Separable vs. Non-separable

The high-dimensional function set could be divided into separable and non-separable functions. A general condition for a function to be separable may be established by stating that [9]:

$$\arg\min_{x_1} f(x_1,\ldots)\ldots,\arg\min_{x_n} f(\ldots,x_n)=\arg\min_{x_1,\ldots,x_n} f(x_1,\ldots,x_n)$$

Which means that f can be optimized through a sequence of n independent 1-D optimization processes. More specifically, separable functions may be classified into linearly separable functions and logarithmic separable functions. A function $f:\Re^n\longrightarrow\Re$ is linearly separable if there exist functions $f_1,\ldots,f_n$ such that:

$$f(\vec{x})=\sum_{i\in S} f_i(x_i)$$

On the other hand, we say that a function $g:\Re^n\longrightarrow\Re$ is logarithmic separable if there exist functions $g_1,\ldots,g_n$ such that:

$$g(\vec{x})=\prod_{i\in S} g_i(x_i)$$

In our benchmark function set f_1, f_2, f_4, f_6, f_7, f_8, and f_9 are all separable functions and f_3, f_5 and f_{10}-f_{13} are non separable functions.

Table 1. The benchmark function set

	Functions	*Dimension*	*Unimodal*	*Separable*	*Continuous*	*Local Minima*	*Global Minima*
f_1	Sphere	10-30-50	Yes	Yes	Yes	0	1
f_2	Schwefel 2.22	10-30-50	Yes	Yes	No	0	1
f_3	Schwefel 1.2	10-30-50	Yes	No	Yes	0	1
f_4	Schwefel 2.21	10-30-50	Yes	Yes	No	0	1
f_5	Rosenbrock	10-30-50	No	No	Yes	Several	1
f_6	Step	10-30-50	Yes	Yes	No	0	1
f_7	Quartic i.e. noise	10-30-50	Yes	Yes	Yes	0	1
f_8	Schwefel 2.26	10-30-50	No	Yes	Yes	Several	1
f_9	Rastrigin	10-30-50	No	Yes	Yes	Several	1
f_{10}	Ackley	10-30-50	No	No	Yes	Several	1
f_{11}	Griewank	10-30-50	No	No	Yes	Several	1
f_{12}	Penalized 1	10-30-50	No	No	No	Several	1
f_{13}	Penalized 2	10-30-50	No	No	No	Several	1
f_{14}	Shekel's Foxholes	2	No	---	---	25	1
f_{15}	Kowalik	4	No	---	---	0	3
f_{16}	Six-Hump Camel-Back	2	No	---	---	4	2
f_{17}	Branin	2	No	---	---	0	3
f_{18}	Goldstein-Price	2	No	---	---	4	1
f_{19}	Hartman	3	No	---	---	4	1
f_{20}	Hartman	6	No	---	---	6	1
f_{21}	Shekel (m=5)	4	No	---	---	5	1
f_{22}	Shekel (m=7)	4	No	---	---	7	1
f_{23}	Shekel (m=10)	4	No	---	---	10	1

2.1.3 Continuous vs. Discontinuous

Finally, functions have been classified into continuous and discontinuous. f_1, f_3, f_5 and f_7-f_{11} are continuous functions, and f_2, f_4, f_6, f_{12} and f_{13} are discontinuous. Obviously, discontinuous functions are more difficult to solve when running algorithms that use the current solution to generate a new one by performing small changes on the current solution.

2.2 Evolutionary Algorithms

To carry out the comparison tests, we have selected four evolutionary algorithms with very different features in order to perform a more general classification: a real-parameter Genetic algorithm (GA-REAL), which will be used as a reference, the CMA-ES algorithm, the Differential Evolution algorithm (DE) and the Macroevolutionary algorithm (MA). The CMA-ES and the DE algorithms were selected because they have provided the best results in recent similar studies [1], and their application is continuously increasing in recent works. Finally, the inclusion of the MA algorithm in this kind of comparison work has seldom been undertaken, but we consider that this algorithm is very interesting due to its differential features with respect to the typical evolutionary algorithms and because it has provided successful results in optimization problems [10][11].

2.2.1 Genetic Algorithms with real parameter operators

Genetic algorithms (GAs) [12] have had a great deal of success in solving search and optimization problems [13]. They have been successfully applied to real parameter function optimization, which has implied the development of new operators for mutation and crossover. See [14] for a survey of this kind of operators. The version considered in this paper applies a BLX- α crossover operator [15] and a non-uniform mutation operator [16].

2.2.2 CMA-ES

The CMA-ES (Covariance Matrix Adaptation Evolution Strategy) is an evolutionary algorithm for difficult non-linear non-convex optimization problems in continuous domains. It was presented first by Nikolaus Hansen and Andreas Ostermeier [17].

The CMA-ES is a second order approach and estimates a covariance matrix that is closely related to the inverse Hessian if it exists within an iterative procedure. For that reason, the method is feasible on non-separable and/or badly conditioned problems. CMA-ES does not use or approximate gradients and does not even presume or require their existence. Therefore, the CMA-ES is feasible on non-smooth, non-continuous, multimodal and noisy problems. This algorithm has several invariance properties [17].

In the CMA-ES, the sampling of a multivariate normal distribution generates a population of new search points:

$$x_k \sim N(m,\sigma^2 C), k = 1\mathrm{K}\ \lambda$$

Where:

- x_k, is the k-th individual of the population.
- m, is the mean value of the search distribution.
- σ^2, is the "overall" standard deviation or step size.
- C, is the covariance matrix.
- λ, is the population size.

Every generation a new population is generated using the same normal distribution but updating m, σ and C using the current population. The process followed by this algorithm consists in updating the distribution mean m as a weighted average of μ selected points of the current population. This mean m implements a truncation selection by choosing $\mu < \lambda$. Therefore, assigning different weights to the points must also be interpreted as a selection mechanism. The best μ points (μ parents) are selected from the population (non-elitist) and weighted intermediate recombination is applied.

The next algorithm step consists on the step size control. Step size control is necessary because the covariance matrix update can hardly increase the variance in all directions simultaneously. To control step size, CMA-ES utilizes an evolution path, i.e. a sum of successive steps. The method can be applied independently of the covariance matrix update and is denoted as cumulative path length control.

Finally, the update of the covariance matrix is performed. The covariance matrix adaptation learns all pairwise dependencies between variables; off-diagonal entries in the covariance matrix reflect the dependencies. It learns a rotated problem representation and a scaling of the independent components. The update of the covariance matrix uses the evolution path too.

In this paper, the Restart CMA-ES with Increasing Population (here after IPOP-CMA-ES) is used. Anne Auger and Nikolaus Hansen designed this algorithm for the special session on real-parameter optimization of CEC 2005 [18]. When one of the stopping criteria is met, an independent restart is performed with the population size increased by a factor of two.

2.2.3 Differential Evolution

Rainer Storn and Kenneth Price introduced the Differential Evolution algorithm (DE) in 1995 as a very simple population based, stochastic function minimizer [19]. The basic strategy it employs consists in generating a new chromosome vector, named trial vector, by adding the weighed difference of two randomly selected chromosome vectors to a third one, hereafter target vector. If the resulting chromosome vector yields a lower objective function value than a target, the newly generated chromosome, the trial vector, replaces the original one to which it was compared.

The main feature of this algorithm is that it performs mutation based on the distribution of the solution in the current population. Search directions and possible step sizes depend on the location of the individuals selected to calculate the mutation values. Thus, extracting distance and direction information from the population to generate random deviations results in an adaptive scheme with excellent convergence properties. In order to increase the diversity of the population, it applies a differential evolution crossover operator to the trial vector, which consists in choosing some genes from the target vector and some from the trial vector. The execution of this operator is regulated by parameter CR that controls the influence of the parent over the generation of the offspring. Higher values for CR mean less influence of the parent. An F parameter scales the influence of the set of pairs of solutions selected to calculate the mutation value.

Currently, there exist several variants of the original Differential Evolution algorithm. In this paper, we will apply the *DE/rand/1/bin* scheme, where "DE" means Differential

Evolution, "rand" indicates that the individuals to compute the mutation values are chosen at random, "1" is the number of pairs of solutions chosen and "bin" means that a binomial crossover is considered. In binomial crossover, the crossover is performed on each of the D variables whenever a randomly picked number between 0 and 1 is within the CR value. In our tests, the two parameters controlling the DE were chosen as follows [20]:

- F = 0.9 for all functions.
- CR = 0.1 for all separable functions
  0.9 for all non-separable functions.

2.2.4 Macroevolutionary Algorithm

The Macroevolutionary algorithm model (MA) was proposed by Marín and Solé [21] in 1999. It draws inspiration from the dynamics of an ecosystem based only on the relationship between species. Here, the individuals in the population are referred to as species. The biological model of macroevolution simulates the dynamics of species extinction and diversification for large time scales. The links between species are essential to determine the new state of every species each generation, the state of a species *i* is defined as:

$$S_i(t)=\begin{cases}1, \text{ if the state is "alive"}\\ 0, \text{ if the state is "extinct"}\end{cases}$$

The state of a species depends on its relationship to the other

species in the population. This relationship is represented by the connectivity matrix W, where each item $W_{i,j}(t)(i,j \in \{1,\mathsf{K},P\})$ measures the influence of species j on species i in generation t with a continuous value within the interval [-1,1].

Each generation in the Macro Evolutionary algorithm consists of the following steps:

1) *Selection operator:* it permits calculating the surviving species through their relations, i.e. as a sum of penalties and benefits. The state of a given individual S_i will be given by:

$$S_i(t+1)=\begin{cases}1, \text{ if } \sum_{j=1}^{p} W_{i,j}(t)\geq 0\\ 0, \text{ otherwise}\end{cases}$$

Where t is the generation and $W_{i,j}=W(p_i,p_j)$ is calculated as:

$$W_{i,j}=\frac{f(p_i)-f(p_j)}{|p_i-p_j|}$$

being *f* the fitness of the species.

2) *Colonization operator:* this operator allows for filling vacant sites that are freed by extinct individuals. This operator is applied to each extinct species in two ways. With a probability τ, a totally new solution $p_n \in \Omega$ will be generated. Otherwise exploitation of surviving solutions takes place through colonization. A surviving solution is randomly chosen and the extinct solution is "attracted" towards it. The parameter τ may act as a temperature as it can decrease in evolutionary time so as perform a type of simulated annealing process. That is, when the temperature is low, the randomness is low, and, consequently, there is a tendency towards increased exploitation around the surviving individuals, and reduced exploration of new species. Mathematically, the colonization of extinct solution reads:

$$p_i(t+1)=\begin{cases}p_b(t)+\rho\lambda(p_b(t)-p_i(t)), \text{ if } \xi>\tau\\ p_n, \text{ if } \xi\leq\tau\end{cases}$$

Where $\xi \in [0,1]$ is a random number, $\lambda \in [-1,+1]$ is also a random number and both with uniform distribution. ρ and τ are given constants of the algorithm. ρ describes a maximum radius around the surviving solution. For the tests here a value of 0.5 was used. A linear function of time (generations) was used to decrease τ:

$$\tau(t;G)=1-\frac{t}{G}$$

2.3 Error and performance measurement

This section deals with the three measures used in this work to compare the algorithms: the *success rate*, the *genotypic normalized root mean squared error* (G_{NE}) and the *computational complexity*.

2.3.1 Success rate

The absolute error is a typical error measure in comparison tests [7], and provides an indication of the difference between the target and the solution provided by the algorithm. From an evolutionary point of view, it is a phenotypic measurement. The absolute error is calculated as follows:

$$Abs\_error=|f(\vec{x})-f(\vec{x}^*)|$$

Where:

- $\vec{x}$ is the solution obtained.
- $\vec{x}^*$ is the optimum solution.

From the absolute error one can easily derive a performance measure called the *success rate* [7] that provides very useful information about how stable an algorithm is. We consider a solved run when the absolute error value is less than 1.0e-6. The success rate is a percentage value that is calculated using:

$$Success\_rate=\frac{\#Solved\_runs}{\#Total\_runs}\cdot 100\cdot$$

Where:

- *#Solved_runs,* is the number of runs that reach an absolute error of 1.0e-6.
- *#Total_runs.* is the number of runs for each algorithm.

2.3.2 Genotypic Error

Most comparisons in the literature deal with the errors obtained by the different algorithms with respect to an objective value (or function), but it is often as important to know how close to the optimum is the point achieved not in terms of the functional value but of its distance to the optimal point in genotypic space. To this end we have established the genotypic normalized root mean square error (G_{NE}), which is a genotypic measure of the quality of a solution. As we are using an optimization function set with known solutions, it is easy to measure the distance between the expected variable values and the ones encoded in the chromosomes. The G_{NE} calculation is performed as follows:

$$G_{NE} = \frac{\sum_{i=1}^{n} \left(x_i - x_i^*\right)^2}{n}$$

Where:

- $\vec{x}$ is the solution obtained.
- $\vec{x}^*$ is the optimum solution.
- n, is the solution's size.

This error measure is more reliable than the absolute error when we need accuracy in terms reaching the optimum point instead of reaching the optimum function value. For example, when trying to optimize a multimodal function, an algorithm could reach a local minimum with an absolute error similar to another solution that is closer to the global minimum.

2.3.3 Computational complexity

Computational complexity is another typical measure when comparing the performance of algorithms [7]. This parameter was calculated by obtaining values for these 3 different temporal parameters for each algorithm:

1) *Computing time T0*: it is computed using a set of programming instructions that includes all the basic mathematical operations (this time depends on the machine and the programming language). See [7] for details.
2) *Computing time T1*: using benchmark function f_{10} to compute the time to run 200000 evaluations for each dimension *D*.
3) *Computing time T2:* it is the complete computing time for the algorithm with 200000 evaluations of the same dimension *D* with benchmark function f_{10}. This step is executed five times to obtain *T2' = mean(T2)*.

The complexity of the algorithm is calculated from these three parameters *T2'*, *T1*, *T0* through:

$$Complexity = \frac{T2' - T1}{T0}$$

2.4 Experimental procedure

After presenting the benchmark function set, the selected algorithms and the error and performance measures, we can now explain the experimental procedure followed to perform the tests:

Table 2. **Population size results.**

	Algorithms			
n	***DE***	***IPOP CMA***	***MA***	***GA REAL***
	Functions f_1-f_{13}			
10	10	10	20	200
30	30	30	30	300
50	50	50	50	250
	Functions f_{14}-f_{23}			
---	30	300	20	100

- For each pair, algorithm and benchmark function, 25 independent runs were performed.
- The high dimensional function set (f_1-f_{13}) was used to analyze the algorithm's behavior when the complexity of the problem changes because the dimensionality could be scaled in these functions. Three complexity levels have been chosen for this function benchmark set: low, medium and high complexity, which correspond to 10, 30 and 50 dimensions respectively. These dimensionality values are the most commonly used in the analysis of evolutionary algorithms [7].
- The other function subset (f_{14}-f_{23}) is made up of low dimensional functions. This set of functions was used to analyze the behavior of the algorithms when changing the initial population size but not the problem dimensionality. All of these functions are multimodal, and consequently they are more complicated that some functions in the other subset such as f_1-f_4 and f_6-f_7, even with a lower dimensionality.
- A population analysis was performed for each algorithm problem dimension pair in order to choose the population size that provides the best results (see Table 2).
- The number of function evaluations was fixed to n*10000. Where n is the dimensionality of the problem.
- As commented before, the accuracy level was fixed to 1.0e-6 on the absolute error value, so a problem is solved when the absolute error reaches this accuracy level.

Table 3. **Success rate values obtained by the four algorithms applied to benchmark functions f_1-f_{13}**

Alg.	***n***	***p***	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}
DE	***10***	***10***	92%	76%	76%	100%	---	100%	---	---	20%	56%	8%	72%	80%
	30	***30***	100%	100%	100%	100%	---	100%	---	---	88%	100%	100%	100%	100%
	50	***50***	100%	100%	---	---	---	100%	---	---	100%	100%	100%	100%	100%
IPOP CMA	***10***	***10***	100%	100%	100%	100%	100%	100%	---	---	---	100%	100%	100%	100%
	30	***30***	100%	100%	100%	100%	100%	32%	---	---	---	100%	100%	100%	100%
	50	***50***	100%	100%	100%	100%	84%	---	---	---	---	100%	100%	100%	100%
GA REAL	***10***	***200***	100%	100%	---	32%	---	100%	---	16%	64%	100%	44%	100%	92%
	30	***300***	100%	100%	---	---	---	100%	---	---	---	100%	96%	100%	100%
	50	***250***	100%	8%	---	---	---	100%	---	---	---	---	100%	100%	100%
MA	***10***	***20***	100%	100%	---	---	---	100%	---	---	---	100%	---	100%	100%
	30	***30***	100%	---	---	---	---	100%	---	---	---	---	40%	64%	92%
	50	***50***	100%	---	---	---	---	100%	---	---	---	---	52%	20%	72%

Table 4. Success rate classified by function type on benchmark functions f_1-f_{13}

Algorithm	*DE*			*IPOP-CMA*			*GA-REAL*			*MA*		
n	***10***	***30***	***50***	***10***	***30***	***50***	***10***	***30***	***50***	***10***	***30***	***50***
p	***10***	***30***	***50***	***10***	***30***	***50***	***200***	***300***	***250***	***20***	***30***	***50***
Unimodal	74%	83%	50%	83%	72%	66%	55%	50%	35%	50%	45%	33%
Multimodal	34%	70%	71%	71%	71%	71%	59%	57%	43%	43%	28%	21%
Separable	55%	70%	57%	57%	47%	43%	59%	43%	30%	43%	38%	29%
Non-Separable	49%	83%	66%	100%	100%	97%	56%	66%	50%	50%	33%	24%
Continuous	31%	61%	50%	63%	63%	61%	41%	37%	25%	25%	18%	19%
Discontinuous	86%	100%	80%	100%	86%	80%	85%	80%	62%	80%	65%	38%

3. Comparison tests

We have divided the experiments into three sections: high-dimensional, low-dimensional and computational cost tests.

3.1.1 High-dimensional function set

Table 3 shows the success rate values obtained by the four algorithms when applied to the functions of the first benchmark subset (f_1-f_{13}). As observed in Table 3, for the low complexity case (n=10), the IPOP-CMA-ES algorithm obtains the best results providing a 100% success rate for all solved functions (f_7-f_9 are never solved). The DE algorithm results in a 20% success rate for function f_9, which is never solved by IPOP-CMA-ES but it performs worse than IPOP-CMA-ES in the other functions. For functions f_8 and f_9, the GA_REAL algorithm presents the best results, with a success rate of 16% and 64%, respectively. However, the GA-REAL displays worst results than IPOP-CMA-ES for the remaining functions, much in the same way as the DE. The MA algorithm provides a success rate of 100% in all the functions it solves, which are only 6 out of 13 functions.

When the dimensionality is increased to n=30, the DE algorithm improves its results. In this case a more detailed analysis should be performed as the IPOP-CMA-ES success rate and the DE success rate present similar values. Table 4 shows a summary of the success rate classified by function type as established in section 2.1. The grey-colored cells represent the best result for each type of function and each dimension. There is no remarkable difference between DE and IPOP-CMA-ES using the modality feature as a function classifier. The success rate has more or less the same value in these algorithms with a dimensionality of 30, but in the case of unimodal functions, the DE algorithm has a success rate of 83% as compared to 72% for the IPOP-CMA-ES. The difference is not significant on multimodal functions with a success rate of approximately 70% for the two algorithms. However, taking into account the separability feature, the IPOP-CMA-ES performs better than the DE in non-separable functions with a success rate of 100% as compared to 83% for the DE. However, the DE performs better than the IPOP-CMA-ES on separable functions obtaining a success rate of 70% on this kind of functions. Here the IPOP-CMA-ES reaches only a value of 47%. Finally, regarding the continuity feature, the DE is the best performer when solving discontinuous functions achieving a success rate of 100%.

The same analysis has been performed for high dimensionality (n=50). Taking into account the success rate, with this dimensionality the GA-REAL provides successful results. But performing a more detailed analysis (see Table 4), we observe that this algorithm is only competent solving separable functions. The IPOP-CMA-ES algorithm, on the other hand, stands out again on non-separable functions, and even behaving worse on discontinuous functions, it achieves the same success rate as the DE on them. Finally, the results obtained by the DE algorithm are not remarkable in this complexity level, although it is the best algorithm with the IPOP-CMA-ES on discontinuous functions and multimodal functions.

Table 5 shows the results obtained using the G_{NE} as comparison measure. The grey-colored cells represent the best result for each type of function and each dimension. For n=10, the GA-REAL obtains the best G_{NE} results for the functions for which it provides the best success rate results (functions f_1, f_2, f_8-f_{10} and f_{12}). For functions f_3-f_5, f_{11} and f_{13}, the IPOP-CMA-ES algorithm is the one with the best results. And for functions f_6 and f_7 the DE algorithm provides the best G_{NE} results. If the dimensionality is increased to

Table 5. G_{NE} values obtained by the four algorithms applied to benchmark functions f_1-f_{13}

Alg.	*n*	*p*	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}
DE	*10*	*10*	1,40e-3	7,04e-4	6,91e-1	5,18e-7	2,44e-1	0,00e+0	1,70e-2	9,34e+1	1,24e-1	7,16e-2	4,36e-1	4,86e-2	7,25e-3
	30	*30*	1,48e-41	3,38e-50	1,63e-7	4,98e-8	1,69e-1	6,43e-3	1,00e-2	1,39e+1	3,98e-3	3,98e-16	4,11e-9	4,31e-17	0,00e+0
	50	*50*	1,65e-20	3,45e-26	3,71e-2	8,17e-3	1,33e-1	5,99e-3	8,22e-3	8,92e+1	1,26e-10	5,49e-16	3,96e-9	3,39e-17	0,00e+0
IPOP CMA	*10*	*10*	6,56e-9	5,20e-13	1,21e-8	5,89e-13	3,63e-9	1,76e-3	3,33e-2	1,31e+2	1,82e-1	2,71e-10	2,36e-8	6,32e-8	3,38e-8
	30	*30*	8,05e-9	5,19e-13	1,57e-8	6,76e-13	4,05e-9	2,61e-2	1,73e-2	7,59e+1	1,44e-1	5,07e-13	5,87e-8	1,05e-7	2,77e-8
	50	*50*	8,30e-9	5,34e-13	1,37e-8	8,72e-13	6,38e-3	4,15e-2	1,14e-2	5,88e+1	1,62e-1	4,85e-13	7,53e-8	1,17e-7	2,53e-8
GA REAL	*10*	*200*	3,50e-20	2,29e-24	8,90e-2	1,10e-3	3,74e-1	1,21e-2	1,77e-2	3,84e+1	2,16e-2	1,77e-16	4,82e-1	5,83e-17	2,64e-3
	30	*300*	4,71e-13	2,06e-16	7,06e-1	6,21e-4	1,77e-1	7,01e-3	1,16e-2	7,57e+1	1,10e-1	1,29e-13	7,25e-3	1,53e-12	1,58e-12
	50	*250*	3,55e-6	3,93e-8	4,26e+0	1,30e-1	1,54e-1	5,45e-3	9,82e-3	5,65e+1	1,34e-1	9,37e-7	3,10e-5	6,18e-6	6,80e-6
MA	*10*	*20*	8,21e-9	4,64e-10	3,43e-3	5,08e-5	4,93e-1	7,59e-3	2,24e-2	1,12e+2	3,20e-1	4,71e-9	1,94e+0	3,52e-8	3,45e-8
	30	*30*	7,29e-7	1,22e-8	2,65e-1	2,89e-2	3,36e-1	7,62e-3	1,26e-2	8,39e+1	2,26e-1	1,70e-2	1,60e-1	6,35e-2	8,80e-4
	50	*50*	6,18e-6	1,54e-7	9,43e-1	4,68e-1	1,57e-1	6,06e-3	9,57e-3	6,74e+1	1,93e-1	1,73e-2	6,93e-2	1,54e-1	1,34e-3

Table 6. Success rate values obtained by the four algorithms applied to benchmark functions f_{14}-f_{23}

Algorithm	*p*	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}
DE	*30*	100%	---	100%	100%	100%	100%	100%	76%	80%	96%
IPOP CMA	*300*	---	64%	100%	100%	100%	100%	8%	32%	76%	88%
GA REAL	*100*	92%	---	100%	100%	100%	100%	56%	64%	84%	92%
MA	*20*	100%	---	100%	100%	100%	100%	64%	100%	96%	96%

n=30, as we can observe in Table 5, the DE provides the best results in general. For functions f_1, f_2 and f_{10}-f_{13} its results stand out against the other algorithms. The IPOP-CMA-ES algorithm stands out for functions f_3-f_5. Functions f_3 and f_5 are non-separable functions and, as commented before, the DE performs worse than the IPOP-CMA-ES in this kind of functions. With n=50, if the G_{NE} is analyzed, the DE provides the best results for functions f_1, f_2, f_7 and f_9-f_{13} while the IPOP-CMA-ES is again the best on functions f_3-f_5.

From these tests, we can conclude that the IPOP-CMA-ES algorithm is the most stable algorithm in high-dimensional problems. On average, it provides the best results based on success rate despite the fact that it has problems in solving functions such as f_6 and f_9 that are solved by other algorithms. Function f_6 requires a high accuracy level to be solved because it is a step function with high error values for each incorrect gene of chromosome. The DE algorithm is, in these conditions, the most accurate algorithm achieving the best values for the G_{NE} according to the tests.

3.1.2 Low-dimensional function set

As introduced in section 2.1, benchmark functions f_{14}-f_{23} are dimensionally fixed. These functions are all multimodal with different number of local and global minima, so it provides different degrees of difficulty to each function.

The results of this analysis are displayed on table 6 and 7. As in the high-dimensional functions, the success rate and the G_{NE} error are used to analyze the results. On average the most successful algorithm is the MA. This result is very relevant and requires a deeper analysis: functions that belong to this set are all multimodal functions. Functions f_{14} and f_{18}-f_{23} all present several local minima and only one global minimum. For this type of functions, the MA obtains the best results. This algorithm can solve all the functions with a success rate greater than 96%, except function f_{20} that is solved with a success rate of 64%. The other algorithms display success rates lower than the MA success rate. (see Table 6).

The DE algorithm obtains similar results to those of the MA in terms of success rate values, except for functions f_{21} and f_{22} where the results are worse and for function f_{20} where the DE algorithm performs better. If we observe the G_{NE} results obtained by the DE and the MA algorithms, the DE algorithm provides, in general, better results than MA.

It seems that the IPOP-CMA-ES has problems on this kind of functions where there are several local minima, as observed in the results of functions f_{14} and f_{21}-f_{23} in Table 6. The IPOP-CMA-ES never solves the f_{14} function and provides a low success rate value for functions f_{20}-f_{23}. In terms of the corresponding value in Table 7 related to the G_{NE} error, the conclusion is that this algorithm always reaches a local minimum. If we observe the results of function f_{15} and f_{17}, the IPOP-CMA-ES provides the highest success rate results. These functions present three global and no local minima. Although the other three algorithms provide good results for function f_{17}, these results seems to confirm that the IPOP-CMA-ES fails in functions with several local minima but not in functions with no local minima. The IPOP-CMA-ES provides the worst G_{NE} values for functions f_{14}, f_{16} and f_{18}-f_{23}. All of these functions have several local minima. The GA-REAL algorithm achieves a 100% success rate for functions f_{16}-f_{19} like the other three algorithms and fails in the remaining functions.

The main conclusions that can be extracted from these results are that when the objective functions are low-dimensional, the MA algorithm is the best choice to solve functions with several local minima. To solve functions with no local minima, the best choice is the IPOP-CMA-ES.

3.1.3 Computational cost tests

After comparing the algorithms from an error point of view, we have performed comparison tests focusing on the time consumption of their application. This is a very relevant aspect of these types of population-based algorithms, because they can solve optimization problems where other algorithms fail, but at the cost of high computational costs.

A MacOS X 10.5.1 based 2.2 GHz Intel Core 2 Duo with 2GB 667 MHz DDR2 SDRAM and Java 5 were used in these tests.

The results obtained using the computational complexity measure presented in section 2.3.3 for all the algorithms are shown on Table 8. All computing times are measured in milliseconds. As we can see from the table, for low dimensionality (n=10), the GA-REAL has the lowest complexity value followed by the MA. But increasing the dimensionality, the DE algorithm provides the lowest complexity values of the four. The mutation operator of this algorithm is based on mathematical operations with a low computational cost such as additions, subtractions and products. In the case of the IPOP-CMA-ES, it uses an incremental population and it is for this reason that it performs so poorly in terms of computational complexity. The population size reaches values of 640, 1920 and 3200 individuals

Table 7. G_{NE} values obtained by the four algorithms applied to benchmark functions f_{14}-f_{23}

Algorithm	*p*	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}
DE	*30*	5,28e-3	7,25e-1	3,74e-1	1,63e-2	2,57e-9	7,23e-3	4,23e-3	3,41e-3	3,91e-4	9,59e-4
IPOP CMA	*300*	2,57e+0	6,70e-1	3,45e-1	1,40e-2	3,64e-9	1,31e-2	1,69e-1	1,04e+0	3,30e-1	1,20e-1
GA REAL	*100*	6,52e-1	1,10e+0	2,87e-1	1,09e-2	1,41e-9	1,29e-2	8,32e-2	4,00e-1	1,25e-1	4,01e-2
MA	*20*	8,63e-3	9,74e-1	2,87e-1	9,08e-3	2,80e-9	9,85e-3	6,89e-2	6,44e-5	6,04e-2	2,02e-2

with initial populations of 10, 30 and 50 individuals. When the CMA-ES is used without incremental population, its computational complexity is similar to that of the DE but the results are worse.

4. CONCLUSIONS

Comparative results were presented for four evolutionary algorithms, selected by their relevance in optimization problems.

Table 8. Computational complexity results

T0 (ms)	310		
	10	***30***	***50***
T1 (ms)	341	813	1251
	T2' (ms)		
DE	3913	8092	12999
IPOP-CMA	18767	62119	95315
MA	2766	12050	29716
GA-REAL	2075	12212	45022
CMA	4090	6569	18080
	Complexity		
DE	11,52	23,48	37,90
IPOP-CMA	59,44	197,76	303,43
MA	7,82	36,25	91,82
GA-REAL	5,59	36,77	141,20
CMA	12,09	18,57	54,29

They have allowed us to study their capabilities and application domains from a practical point of view. To do this, we have used a benchmark function set that covers a wide range of optimization problems. The algorithms were compared using accuracy, stability and time consumption parameters to establish the type of optimization function they are more suitable for. This way, in high dimensional problems, we have concluded that the IPOP-CMA-ES performs better than the others in non-separable functions while the DE provides the best results on separable and discontinuous functions. On low dimensional benchmarks, the MA algorithm provides the best results in functions with several local minima. To solve functions with no local minima and several global minima, the best choice is the IPOP-CMA-ES. Finally, regarding computational cost, the previous conclusions must be tuned taking into account that the computational cost of the DE algorithms is the lowest.

5. ACKNOWLEDGMENTS

This work was funded by the MEC of Spain through project DPI2006-15346-C03-01and DEP2006-56158-C03-02

6. REFERENCES

[1] Special Session on Real-Parameter Optimization at CEC-05, Edinburgh, UK, 2-5 Sept. 2005.

[2] Workshop on Parameter Setting in Genetic and Evolutionary Algorithms (PSGEA 2005), June, 25-29, 2005, Washington, D.C. USA

[3] Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, June 26, 2005, Washington D.C.

[4] Bui, L. T., Shan, Y., Qi, F., Abbass, H.A. Comparing Two Versions of Differential Evolution in Real Parameter Optimization, *Tech. Report TR-ALAR-200504009, School of ITEE, University of New South Wales,* 2005.

[5] Costa, L. A., Parameter-less Evolution Strategy for Global Optimization, *Proc. 6th WSEAS International Conference on Simulation, Modelling and Optimization*, 2006, 622-627.

[6] Yao, X., Liu, L., Lin G. Evolutionary Programming Made Faster, *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, 1999, 82-102.

[7] Suganthan, P. N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, A. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, *Technical Report*, Nanyang Technological University, Singapore, *and KanGAL Report #2005005, IIT Kanpur, India,* 2005.

[8] Shang, Y. and Qiu, Y. A Note on the Extended Rosenbrock Function. *Evol. Comput. 14, 1*, 2006, 119-126.

[9] Hansen, N. The CMA Evolution Strategy: A Tutorial, In *www.bionik.tu-berlin.de/user/niko/*, 2007

[10] Becerra, J. A., Santos, J., Duro, R.J., Robot Controller Evolution with Macroevolutionary Algorithms, *Information Processing with Evolutionary Algorithms From Industrial Applications to Academic Speculations*, 2005, 117-128

[11] Becerra, J. A., Díaz-Casás, V., Duro, R.J., Exploring Macroevolutionary Algorithms: Some Extensions and Improvements, *Lecture Notes in Computer Science*, vol. 4507, Springer-Verlag, 2007, 308-315

[12] Holland, J.H. Adaptation in natural and artificial systems, *Univ. Michigan Press*, 1975

[13] Goldberg, D.E. Genetic algorithms in search, optimization and machine learning, *Addison-Wesley,* 1989

[14] Herrera, F., Lozano, M., Verdegay, J.L. Tackling real-coded genetic algorithms: Operators and tools for behaviorial analysis. *Artificial Intellig. Review*, 12(4), 1998, 265-319.

[15] Eshelman, L.J., Schaffer, J.D. Real-Coded Genetic Algorithms and Interval Schemata, *Foundations of Genetic Algorithms 2*, 1993, 187-202.

[16] Michalewicz, Z. Genetic algorithms + Data Structures = Evolution Programs. *Springer-Verlag*, 1992.

[17] Hansen, N. and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation, 9(2)*, 2001, 159-195.

[18] Auger, A, and Hansen, N. A Restart CMA Evolution Strategy With Increasing Population Size. In *Proc. of the IEEE, CEC 2005*, 2005, 1769-1776.

[19] Storn, R. and Price, K. Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces *Technical Report TR-95-012, ICSI*, 1995.

[20] Rönkkönen, J., Kukkonen, S. Price, K. Real-parameter optimization with differential evolution, *Proc. of 2005 IEEE Congress on Evolutionary Computation*, 2005, 506-513.

[21] Marin, J., and Solé, R. V. Macroevolutionary algorithms: A new optimization method on tness landscapes. *IEEE Trans. on Evolutionary Computation 3, 4*, 1999, 272-286.