Immune Algorithm Versus Differential Evolution: A Comparative Case Study Using High Dimensional Function Optimization

Vincenzo Cutello[1], Natalio Krasnogor[2], Giuseppe Nicosia[1], and Mario Pavone[1]

[1] Department of Mathematics and Computer Science
University of Catania
Viale A. Doria 6, 95125 Catania, Italy
{vctl,nicosia,mpavone}@dmi.unict.it
[2] Automated Scheduling, Optimisation and Planning Research Group
School of Computer Sciences and IT
Jubilee Campus, University of Nottingham
Nottingham, NG81BB, UK
Natalio.Krasnogor@nottingham.ac.uk

Abstract. In this paper we propose an immune algorithm (IA) to solve high dimensional global optimization problems. To evaluate the effectiveness and quality of the IA we performed a large set of unconstrained numerical optimisation experiments, which is a crucial component of many real-world problem-solving settings. We extensively compare the IA against several Differential Evolution (DE) algorithms as these have been shown to perform better than many other Evolutionary Algorithms on similar problems. The DE algorithms were implemented using a range of recombination and mutation operators combinations. The algorithms were tested on 13 well known benchmark problems. Our results show that the proposed IA is effective, in terms of accuracy, and capable of solving large-scale instances of our benchmarks. We also show that the IA is comparable, and often outperforms, all the DE variants, including two Memetic algorithms.

1 Introduction

Since in many real-world engineering and technology applications analytical solutions, even for simple problems, are not allways available, numerical continuous optimisation is often the only viable alternative.

A global minimization problem can be formalized as a pair (S, f), where $S \subseteq \mathbb{R}^n$ is a bounded set on $\mathbb{R}^n$ and $f : S \to \mathbb{R}$ is an n-dimensional real-valued function. The goal is to find a point $x_{\min} \in S$ such that $f(x_{\min})$ is a global minimum on S, i.e. $\forall x \in S : f(x_{\min}) \leq f(x)$.

The problem of continous optimisation is a difficult one not least because it is difficult to decide when a global (or local) optimum has been reached but also because there could be very many local optima that traps the search algorithm. Furthermore, as the dimensionality of the problem increases the number of local optima grows dramatically.

B. Beliczynski et al. (Eds.): ICANNGA 2007, Part I, LNCS 4431, pp. 93–101, 2007.

In this work we consider the following numerical minimization problem: $\min(f(\boldsymbol{x})), \quad \boldsymbol{B_l} \leq \boldsymbol{x} \leq \boldsymbol{B_u}$ where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is the variable vector in $\mathbb{R}^n$, $f(\boldsymbol{x})$ denotes the objective function to *minimize* and $\boldsymbol{B_l} = (B_{l_1}, B_{l_2}, \ldots, B_{l_n})$, $\boldsymbol{B_u} = (B_{u_1}, B_{u_2}, \ldots, B_{u_n})$ represent, respectively, the variables' lower and the upper bounds, such that $x_i \in [B_{l_i}, B_{u_i}]$.

We use the above formulation to evaluate our immune algorithm (IA), first proposed in [1], for high dimensional problems. Moreover, we compare the results of the IA with several *Differential Evolution* (DE) variants as those proposed in [6], [4] and [5]. DE were chosen as they typically show better convergence behaviour than other well-known EAs [8], [9].

2 Differential Evolution

Differential Evolution (DE) is an effective and efficient method which has been proposed to solve optimization problems in continuous search spaces [8], [9]. The main advantage of DE is its simple structure, because it has few control variables, and it uses common concepts of Evolutionary Algorithms (EA). DE[1] is based on a population of individuals, generated through a similar operation to the classical *mutation*, where, for each individual x_i^t, a new individual y_i^{t+1} is generated as follows: $y_i^{t+1} = x_j^t + F(x_k^t - x_l^t)$, and F is called *scaling factor*. To have high diversity into the population, DE uses a *crossover* operator, between x_i^t and y_i^{t+1}, to generate the offspring x_i^{t+1}. Finally, the offspring is evaluated and it replaces its parent if its fitness is better than its parent (selection process).

There are several variants of DE, depending on the the selection for mutation operator, and the crossover scheme used. To distinguish its several variants we use the notation *a/b/c*, where *"a"* denotes the way an individual is selected to be mutated (random or best individual); *"b"* is the number of pairs of solutions chosen and *"c"* represent the crossover scheme used (binomial or exponential). Therefore, using this notation we have the following variants [6]: *rand/1/bin*, *rand/1/exp*, *best/1/bin* and *best/1/exp*; *current-to-rand/1* and *current-to-best/1*, which use an arithmetic recombination; *current-to-rand/1/bin*, which presents a combined discrete-arithmetic recombination; and *rand/2/dir*, which includes fitness function information to the mutation and recombination operators.

3 Immune Algorithms

The immune algorithms are inspired by the human's clonal selection principle, which suggests that among all possible cells, B lymphocytes, with different receptors circulating in the host organism, only those who are actually able to recognize the antigen will start to proliferate by cloning.

The proposed algorithm is population based, like any typical evolutionary algorithm and it is based on two entity types: antigens (Ag) and B cells receptor. The Ag's represent the problem to tackle, i.e. the function to optimize, whereas

[1] for a deeply knowledge on DE see [8], [9].

the B cells are a population of points of the search space of the given function. At each time step t the algorithm presents a population $P^{(t)}$ of size d. The initial population of candidate solutions at time $t = 0$ is randomly generated using uniform distribution in the relative domains of each function. The proposed algorithm generates each gene $x_i = B_{l_i} + \beta \cdot (B_{u_i} - B_{l_i})$, where $\beta \in [0, 1]$ is a random value, B_{l_i} and B_{u_i} are the lower and upper bounds of the real coded variable x_i respectively.

The function *Evaluate*($P^{(t)}$) (see Table 3) computes the fitness function value of each B cell $\boldsymbol{x} \in P^{(t)}$. The evolution cycle ends when the maximum number of fitness function evaluations, T_{max}, is reached.

We used the classical cloning operator, which clones each B cell *dup* times producing an intermediate population $P^{(clo)}$, assigning to each clone the same age of its parent. The age of B cells, determines their life span into the population: when a B cell reaches the maximum allowed age, it dies, i.e. it is eliminated from the population. Subsequentely, if a cloned B cell undergoes a successfully mutation (called *constructive mutation*), i.e. its fitness value has increased, it will be considered to have age equal to 0. Such a scheme, as proposed in [2], intends to give an equal opportunity to each new B cell to effectively explore the search landscape.

The hypermutation operators act on the B cell receptor of $P^{(clo)}$. We used the inversely proportional hypermutation operator, which tries to mutate each B cell receptor M times without the explicit usage of a mutation probability. The feature of this operator is that the number of mutations is inversely proportional to the fitness value, i.e. as the fitness function value of the current B cell increases, the number of mutations performed decreases. The mutation potential used was $\alpha = e^{(-\rho \cdot f)}$, where α represents the mutation rate and f is the fitness function value normalized in $[0, 1]$. The perturbation operator choose randomly a variable x_i, with $i \in \{1, \ldots, \ell\}$ (ℓ is the length of B cell) and replace it with $x_i^{new} = ((1 - \beta) \cdot x_i) + (\beta \cdot x_{random})$, where $x_{random} \neq x_i$ is a randomly chosen variable and $\beta \in [0, 1]$ is a random number obtained with uniform distribution. As proposed in [1], the proposed algorithm does not use any additional information concerning the problem. For example the global optima is not considered when normalizing the fitness function value, but the best current fitness value decreased of a threshold θ.

The aging operator, used by the algorithm, eliminates old B cells, in the populations $P^{(t)}$, and $P^{(hyp)}$, maintaining high diversity in the current population, in order to avoid premature convergence. The maximum number of generations the B cells are allowed to remain in the population is determined by the τ_B parameter: when a B cell is $\tau_B + 1$ old it is erased from the current population, independently from its fitness value. The algorithm makes only one exception: when generating a new population the selection mechanism does not allow the elimination of the B cell with the best fitness function value (*elitist aging*). After the application of the immune operators the best surviving B cells are selected from the populations $P_a^{(t)}$ and $P_a^{(hyp)}$. In this way, the new population $P^{(t+1)}$, of d B cells, for the next generation $t + 1$, is obtained. If only $d' < d$ B cells have

Table 1. Pseudo-code of the proposed Immune Algorithm

```
Immune Algorithm(d, dup, ρ, τ_B, T_max)
  FFE ← 0;
  N_c ← d · dup;
  t ← 0;
  P^(t) ← Init_Population(d);
  Evaluate(P^(t));
  FFE ← FFE + d;
  while (FFE < T_max)do
      P^(clo) ← Cloning (P^(t), dup);
      P^(hyp) ← Hypermutation(P^(clo), ρ);
      Evaluate(P^(hyp));
      FFE ← FFE + N_c;
      (P_a^(t), P_a^(hyp)) = Aging(P^(t), P^(hyp), τ_B);
      P^(t+1) ← (μ + λ)-Selection(P_a^(t), P_a^(hyp));
      t ← t + 1;
  end_while
```

survived, the $(\mu + \lambda)$-*Selection operator* randomly selects $d - d'$ "old" B cells from $P_a^{(t)} \sqcup P_a^{(hyp)}$.

In Table 3 is showed the pseudo-code of the proposed immune algorithm.

4 Experimental Results

To better understand the search ability of the proposed IA and its real performances, we used a large set of experiments on two different categories of functions with different features, using high different dimensional values. Moreover, we compared our algorithm to several DE variants, as proposed in [6], [5], [4], to evaluate the goodness of the proposed solutions by IA. We used the first 13 well-known benchmark functions from [3]: *unimodal functions*, that are relatively easy to optimize, but their difficulty increases as the dimensional space increase, and *multimodal functions*, with many local minima, that represent the most difficult class of problems for many optimization algorithms. Last category is the most important, because the quality of the final results reflects the ability of the given algorithm to *escape* from local optima.

For all experiments we used the following values for IA: $d = 100, dup = 2$, $\tau_B = 15$, and $\theta = 75\%$, where θ represents the threshold used to decrease the best fitness function value obtained in each generation to normalize the fitness in the range $[0, 1]$ [1]. Moreover, we used $\rho = 7$ for high dimension values, whilst lower ρ values for smaller dimensions.

Figure 1 shows the mutation number obtained at different fitness function values, from worst to best. These experiment were obtained using small and high dimensional values. In the inset plot we give a zoom reducing the normalized fitness function in the range $[0.4, 1]$.

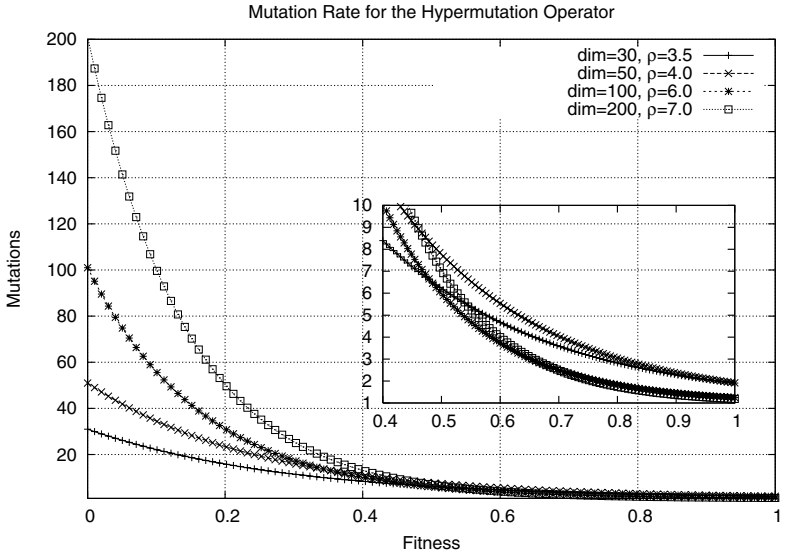

Fig. 1. Mutation Potential behavior using different dimension values

At each generation we computed the *mean value* of the best fit individuals for all independently runs and the *standard deviation*, to indicate the consistency of the algorithm.

In the first experiment, IA is compared with the 8 DE variants, proposed in [6], where T_{max} was fixed to 12×10^4. For each function 100 independent runs were performed. The dimension of the functions was fixed to 30, and the results are shown in Table 2. Because in [6] the authors have modified the function f_8 to have its minimum at zero (rather than -12569.5), it is not included into the same table. The best results obtained by each algorithm are shown in boldface. From this table, one can see that IA outperforms the majority of the DE variants, either in unimodal class or multimodal, and it is comparable with the best DE algorithm.

In Table 3 IA is compared to *rand/1/bin* variant, using a different experimental protocol, proposed in [5]. For each experiment the maximum number of fitness function evaluations T_{max} was fixed to 5×10^5, for function dimension 30 or less, whilst using 100 dimension, T_{max} was set to 5×10^6. For each benchmark function, 30 independent runs were performed. For each function the mean best function values found in the last generation is shown in the first row, and standard deviation in the second. The results obtained for both 30 and 100 variables IA are comparable to the results obtained by *rand/1/bin*. In this comparison all results below 10^{-20} were reported as 0.0.

Recent developments in EAs field, have shown that to tackle complex search spaces, pure genetic algorithms (GA) need to use local search operators and specialized crossover [7]. In [4] two memetic versions of DE, which used *crossover based local search* (XLS), were proposed. As a last experiment, IA was compared

Table 2. Immune Algorithm versus DE variants on the unimodal and multimodal functions, using 30 dimension

Unimodal Functions						
	f_1	f_2	f_3	f_4	f_6	f_7
IA	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.0000489
rand/1/bin	**0.0**	**0.0**	0.02	1.9521	**0.0**	**0.0**
rand/1/exp	**0.0**	**0.0**	**0.0**	3.7584	0.84	**0.0**
best/1/bin	**0.0**	**0.0**	**0.0**	0.0017	**0.0**	**0.0**
best/1/exp	407.972	3.291	10.6078	1.701872	2737.8458	0.070545
current-to-best/1	0.54148	4.842	0.471730	4.2337	1.394	**0.0**
current-to-rand/1	0.69966	3.503	0.903563	3.298563	1.767	**0.0**
current-to-rand/1/bin	**0.0**	**0.0**	0.000232	0.149514	**0.0**	**0.0**
rand/2/dir	**0.0**	**0.0**	30.112881	0.044199	**0.0**	**0.0**
Multimodal Functions						
	f_5	f_9	f_{10}	f_{11}	f_{12}	f_{13}
IA	11.69	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
rand/1/bin	19.578	**0.0**	**0.0**	0.001117	**0.0**	**0.0**
rand/1/exp	**6.696**	97.753938	0.080037	0.000075	**0.0**	**0.0**
best/1/bin	30.39087	**0.0**	**0.0**	0.000722	**0.0**	0.000226
best/1/exp	132621.5	40.003971	9.3961	5.9278	1293.0262	2584.85
current-to-best/1	30.984666	98.205432	0.270788	0.219391	0.891301	0.038622
current-to-rand/1	31.702063	92.263070	0.164786	0.184920	0.464829	5.169196
current-to-rand/1/bin	24.260535	**0.0**	**0.0**	**0.0**	0.001007	0.000114
rand/2/dir	30.654916	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

Table 3. Performance Comparison among IA and *"rand/1/bin"* on 13 functions benchmarks, using 30 and 100 dimensions

	30 *dimension*		100 *dimension*	
	IA	***rand/1/bin***	IA	***rand/1/bin***
f_1	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**
f_2	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**
f_3	**0.0** **0.0**	2.02×10^{-9} 8.26×10^{-10}	**0.0** **0.0**	5.87×10^{-10} 1.83×10^{-10}
f_4	**0.0** **0.0**	3.85×10^{-8} 9.17×10^{-9}	6.447×10^{-7} 3.338×10^{-6}	$\mathbf{1.128 \times 10^{-9}}$ $\mathbf{1.42 \times 10^{-10}}$
f_5	12 13.22	**0.0** **0.0**	74.99 38.99	**0.0** **0.0**
f_6	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**
f_7	$\mathbf{1.521 \times 10^{-5}}$ $\mathbf{2.05 \times 10^{-5}}$	4.939×10^{-3} 1.13×10^{-3}	$\mathbf{1.59 \times 10^{-5}}$ $\mathbf{3.61 \times 10^{-5}}$	7.664×10^{-3} 6.58×10^{-4}
f_8	$-1.256041 \times 10^{+4}$ 25.912	$\mathbf{-1.256948 \times 10^{+4}}$ $\mathbf{2.3 \times 10^{-4}}$	$-4.16 \times 10^{+4}$ $2.06 \times 10^{+2}$	$\mathbf{-4.1898 \times 10^{+4}}$ $\mathbf{1.06 \times 10^{-3}}$
f_9	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**
f_{10}	0.0 0.0	$\mathbf{-1.19 \times 10^{-15}}$ $\mathbf{7.03 \times 10^{-16}}$	**0.0** **0.0**	8.023×10^{-15} 1.74×10^{-15}
f_{11}	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**	5.42×10^{-20} 5.42×10^{-20}
f_{12}	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**	**0.0** **0.0**
f_{13}	0.0 0.0	**−1.142824** $\mathbf{4.45 \times 10^{-8}}$	0.0 0.0	**−1.142824** $\mathbf{2.74 \times 10^{-8}}$

Table 4. IA versus *rand/1/exp*, *best/1/exp* and their memetic versions by [4], with $n = 50$ dimensional search space

	IA	*rand/1/exp*	*best/1/exp*	DEfirDE	DEfirSPX
f_1	**0 ± 0**	0 ± 0 0 ± 0 0.0535 ± 0.0520	309.74 ± 481.05 0 ± 0 0.0027 ± 0.0013	0 ± 0 0 ± 0 0.0026 ± 0.0023	0 ± 0 0 ± 0 $1 \cdot 10^{-4} \pm 4.75 \cdot 10^{-5}$
f_5	**30 ± 21.7**	79.8921 ± 102.611 52.4066 ± 19.9109 90.0213 ± 33.8734	$3.69 \cdot 10^{+5} \pm 5.011 \cdot 10^{+5}$ 54.5985 ± 25.6652 58.1931 ± 9.4289	72.0242 ± 47.1958 53.1894 ± 26.1913 66.9674 ± 23.7196	65.8951 ± 37.8933 45.8367 ± 10.2518 52.0033 ± 13.6881
f_9	**0 ± 0**	0 ± 0 0 ± 0 0 ± 0	0.61256 ± 1.1988 0 ± 0 0 ± 0	0 ± 0 0 ± 0 0 ± 0	0 ± 0 0 ± 0 0 ± 0
f_{10}	**0 ± 0**	0 ± 0 $9.36 \cdot 10^{-6} \pm 3.67 \cdot 10^{-6}$ 0.0104 ± 0.0015	0.2621 ± 0.5524 $6.85 \cdot 10^{-6} \pm 6.06 \cdot 10^{-6}$ 0.0067 ± 0.0015	0 ± 0 $2.28 \cdot 10^{-5} \pm 1.45 \cdot 10^{-5}$ 0.0060 ± 0.0015	0 ± 0 $3.0 \cdot 10^{-6} \pm 1.07 \cdot 10^{-6}$ 0.0019 ± $4.32 \cdot 10^{-4}$
f_{11}	**0 ± 0**	0 ± 0 $9.95 \cdot 10^{-7} \pm 4.3 \cdot 10^{-7}$ 0.0053 ± 0.010	0.1651 ± 0.2133 0 ± 0 0.0012 ± 0.0028	0 ± 0 0 ± 0 $4.96 \cdot 10^{-4} \pm 6.68 \cdot 10^{-4}$	0 ± 0 0 ± 0 $5.27 \cdot 10^{-4} \pm 0.0013$

Table 5. IA versus *rand/1/exp*, *best/1/exp* and their memetic versions by [4], with $n = 100$ dimensional search space

	IA	*rand/1/exp*	*best/1/exp*	DEfirDE	DEfirSPX
f_1	**0 ± 0**	$1.58 \cdot 10^{-6} \pm 3.75 \cdot 10^{-6}$ 59.926 ± 16.574 2496.82 ± 246.55	0.0046 ± 0.0247 30.242 ± 5.93 1729.40 ± 172.28	0 ± 0 11.731 ± 5.0574 358.57 ± 108.12	0 ± 0 1.2614 ± 0.4581 104.986 ± 22.549
f_5	**85.6 ± 31.758**	120.917 ± 41.8753 12312.16 ± 3981.44 $3.165 \cdot 10^{+6} \pm 6.052 \cdot 10^{+5}$	178.465 ± 60.938 7463.633 ± 2631.92 $1.798 \cdot 10^{+6} \pm 3.304 \cdot 10^{+5}$	107.5604 ± 28.2529 2923.108 ± 1521.085 $2.822 \cdot 10^{+5} \pm 3.012 \cdot 10^{+5}$	99.1086 ± 18.5735 732.85 ± 142.22 16621.32 ± 6400.43
f_9	**0 ± 0**	0 ± 0 2.6384 ± 0.7977 234.588 ± 13.662	0 ± 0 0.7585 ± 0.2524 198.079 ± 18.947	0 ± 0 0.1534 ± 0.1240 17.133 ± 7.958	0 ± 0 0.0094 ± 0.0068 27.0537 ± 20.889
f_{10}	**0 ± 0**	$1.02 \cdot 10^{-6} \pm 1.6 \cdot 10^{-7}$ 1.6761 ± 0.0819 7.7335 ± 0.1584	$9.5 \cdot 10^{-7} \pm 1.1 \cdot 10^{-7}$ 1.2202 ± 0.0965 6.7251 ± 0.1373	$1.2 \cdot 10^{-6} \pm 6.07 \cdot 10^{-7}$ 0.5340 ± 0.1101 3.7515 ± 0.2773	0 ± 0 0.3695 ± 0.0734 3.4528 ± 0.1797
f_{11}	**0 ± 0**	0 ± 0 1.1316 ± 0.0124 20.037 ± 0.9614	0 ± 0 1.0530 ± 0.0100 13.068 ± 0.8876	0 ± 0 0.7725 ± 0.1008 3.7439 ± 0.7651	0 ± 0 0.5433 ± 0.1331 2.2186 ± 0.3010

Table 6. IA versus *rand/1/exp*, *best/1/exp* and their memetic versions by [4], with $n = 200$ dimensional search space

	IA	*rand/1/exp*	*best/1/exp*	DEfirDE	DEfirSPX
f_1	**0 ± 0**	50.005 ± 16.376 $5.45 \cdot 10^{+4} \pm 2605.73$ $1.82 \cdot 10^{+5} \pm 6785.18$	26.581 ± 7.4714 $4.84 \cdot 10^{+4} \pm 1891.24$ $1.74 \cdot 10^{+5} \pm 6119.01$	17.678 ± 9.483 9056.0 ± 1840.45 44090.5 ± 6122.35	0.8568 ± 0.2563 2782.32 ± 335.69 9850.45 ± 1729.9
f_5	**165.1 ± 71.2**	9370.17 ± 3671.11 $4.22 \cdot 10^{+8} \pm 3.04 \cdot 10^{+7}$ $3.29 \cdot 10^{+9} \pm 2.12 \cdot 10^{+8}$	6725.48 ± 1915.38 $3.54 \cdot 10^{+8} \pm 3.54 \cdot 10^{+7}$ $3.12 \cdot 10^{+9} \pm 1.65 \cdot 10^{+8}$	5302.79 ± 2363.74 $2.39 \cdot 10^{+7} \pm 6.379 \cdot 10^{+6}$ $3.48 \cdot 10^{+8} \pm 1.75 \cdot 10^{+8}$	996.69 ± 128.483 $1.19 \cdot 10^{+6} \pm 4.10 \cdot 10^{+5}$ $1.21 \cdot 10^{+7} \pm 4.73 \cdot 10^{+6}$
f_9	**0 ± 0**	0.4245 ± 0.2905 1878.61 ± 60.298 5471.35 ± 239.67	0.2255 ± 0.1051 1761.55 ± 43.3824 5094.97 ± 182.77	0.1453 ± 0.2771 352.93 ± 46.11 1193.83 ± 145.477	0.0024 ± 0.0011 369.88 ± 136.87 859.03 ± 99.76
f_{10}	**0 ± 0**	0.5208 ± 0.0870 15.917 ± 0.1209 19.253 ± 0.0698	0.4322 ± 0.0427 15.46 ± 0.1205 19.138 ± 0.0772	0.3123 ± 0.0426 9.2373 ± 0.4785 14.309 ± 0.3706	0.1589 ± 0.0207 6.6861 ± 0.3286 9.4114 ± 0.4581
f_{11}	**0 ± 0**	0.7687 ± 0.0768 490.29 ± 21.225 1657.93 ± 47.142	0.5707 ± 0.0651 441.97 ± 15.877 1572.51 ± 53.611	0.5984 ± 0.1419 78.692 ± 11.766 368.90 ± 41.116	0.1631 ± 0.0314 28.245 ± 4.605 85.176 ± 12.824

with *rand/1/exp*, *best/1/exp* and their memetic versions, called DEfirDE and DEfirSPX [4], respectively. For each experiment, the maximum number of fitness function evaluations T_{max} was fixed to 5×10^5, and 50, 100 and 200 dimension variables were used. For each instance 30 independent runs were performed. The same functions proposed in [4]: f_1, f_5, f_9, f_{10} and f_{11}, were used. In Tables 4, 5 and 6, for the two DE variants and their memetic versions, are reported the results obtained varying the population size, with $n, 5n$ and $10n$, where n is the dimensional search space, as showed in [4]. Moreover, for each algorithm, is showed the *mean value* of the best fit individuals and *standard deviation*

($mean \pm sd$), used to indicate the consistency of the algorithm. In Table 4 we report the results obtained using $n = 50$ dimensional search space. From this table one can see that IA is comparable and in many cases outperforms DE and its memetic variants, expecially for f_5.

In Tables 5 and 6 one can see the results obtained using 100 and 200 dimensional search space. From these two tables it is clear that IA outperforms the compared algorithms when increasing the function dimension. It is important to highlight that our proposed algorithm outperforms DE variants and their memetic versions, in each function using smaller population size for $n = 100$ and $n = 200$.

5 Conclusion

The main features of the IA can be sumarized as: (1) the *cloning* operator, which explores the neighbourhood of a given solution, (2) the *inversely proportional hypermutation* operator, that perturbs each candidate solution as a function of its fitness function (inversely proportional), and (3) the *aging operator*, that eliminates the oldest candidate solutions from the current population in order to introduce diversity and thus avoiding local minima during the search process.

In this research paper we presented an extensive comparative study illustrating the performance of a well-known immune algorithm [1], with the features mentioned above, and that of several differential evolution variants [6], [5] and their memetic versions [4]. We used the 13 classical benchmark functions from [3] (unimodal and multimodal functions) for our experiments. Furthermore the dimensionality of the problems was varied from $n = 30$ to $n = 200$ dimensions.

Our results suggest that the proposed immune algorithm is an effective numerical optimization algorithm(in terms of solution quality) particularly for the most challenging highly dimensional search spaces. In particular, increasing the dimension of the solutions space improves the performances of IA. Finally, the experimental results also show that the IA is comparable, and it often outperforms, all 8 DE variants as well as their memetic counterparts.

References

1. Cutello V., Nicosia G., Pavone M., Narzisi G.; "*Real Coded Clonal Selection Algorithm for Unconstrained Global Numerical Optimization using a Hybrid Inversely Proportional Hypermutation Operator*," in 21st Annual ACM Symposium on Applied Computing (SAC), vol. 2, pp. 950–954 (2006).
2. Cutello V., Morelli G., Nicosia G., Pavone M.; "*Immune Algorithms with Aging Operators for the String Folding Problem and the Protein Folding Problem*," in 5th European Conference on Computation in Combinatorial Optimization (EvoCOP), pp. 80–90 (2005).
3. Yao X., Liu Y., Lin G. M.; "*Evolutionary programming made faster*," IEEE Transaction on Evolutionary Computation, vol. 3, no. 2, pp. 82–102 (1999).
4. Noman N., Iba H.; "*Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization*," in Genetic and Evolutionary Computation Conference (GECCO), pp. 967–974 (2005).

5. Versterstrøm, J., Thomsen R.: "*A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems*," in Congress on Evolutionary Computing (CEC), vol. 1, pp. 1980-1987, (2004).
6. Mezura–Montes E., Velázquez–Reyes J., Coello Coello C.; "*A Comparative Study of Differential Evolution Variants for Global Optimization*," in Genetic and Evolutionary Computation Conference (GECCO), vol. 1, pp. 485–492 (2006).
7. Goldberg D. E., Voessner S.; "*Optimizing Global-Local Search Hybrids*," in Genetic and Evolutionary Computation Conference (GECCO), pp. 220–228 (1999).
8. Storn R., Price K. V.; "*Differential Evolution a Simple and Efficient Heuristic for Global Optimization over Continuos Spaces*," in Journal of Global Optimization, vol. 11, no. 4, pp. 341–359 (1997).
9. Price K. V., Storn M., Lampien J. A.; "*Differential Evolution: A Practical Approach to Global Optimization*," Springer-Verlag (2005).