EPJ Web of Conferences **108**, 02048 (2016)
DOI: 10.1051/epjconf/201610802048

Comparison of the Asynchronous Differential Evolution and JADE Minimization Algorithms

Mikhail Zhabitsky[1,a,b]

[1] *Joint Institute for Nuclear Research, 6, Joliot Curie St., 141980 Dubna, Moscow Region, Russia*

Abstract. Differential Evolution (DE) is an efficient evolutionary algorithm to solve global optimization problems. In this work we compare performance of the recently proposed Asynchronous Differential Evolution with Adaptive Correlation Matrix (ADE-ACM) to the widely used JADE algorithm, a DE variant with adaptive control parameters.

1 Introduction

Global minimization of a real-valued function f, defined in the continuous parameter space Ω of dimension D, is a common mathematical problem

$$\vec{x}^* = \text{argmin} f(\vec{x}), \quad \vec{x} \in \Omega \subset \mathbb{R}^D, \quad \vec{x} = \{x_j\}\Big|_{j=0,\ldots,D-1}. \tag{1}$$

Thanks to its simple structure, Differential Evolution (DE) [1] is a widely used method to find the global minimum $f^* = f(\vec{x}^*)$. It has few control parameters, but some of them, the population size N_p and the crossover rate C_r, drastically change the performance of the algorithm. Moreover incompatible settings are efficient to solve different classes of problems, e.g. $C_r = 0$ for separable problems and $C_r \approx 1$ for non-separable ones. Therefore recent studies were focused on modifications of DE, which automatically adapt the control parameters during minimization [2]. We will compare the adaptive JADE algorithm [3] to the Asynchronous Differential Evolution with Adaptive Correlation Matrix [4], we will disentangle contributions due to differences in algorithms.

2 Asynchronous Differential Evolution

Asynchronous Differential Evolution (ADE) [5] is a steady state variant of the DE method, its general scheme is shown in Fig. 1. DE uses a population P of N_p vectors $\vec{x}_i$ to represent candidate solutions in the search domain. The initial population is formed by a uniform random sampling of each coordinate $x_{i,j}$ within requested initial boundaries $[\vec{x}_{\min}, \vec{x}_{\max}]$.

The standard DE is a generational algorithm: mutation and crossover operations are performed for all population members, then as a result of selection DE switches to a next generation. But in a steady state algorithm evolutionary operations are imposed on a selected member of the population, thus a

[a] e-mail: Mikhail.Zhabitsky@jinr.ru
[b] This work was in part supported by Russian Foundation for Basic Research (Grant No. 13-01-00060).

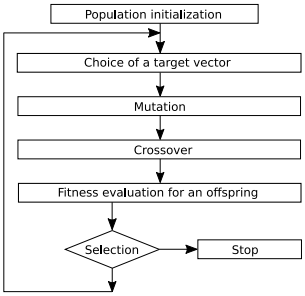

Figure 1: Scheme of a steady-state DE

Table 1: List of ADE variants, used for numerical tests

Abbr.	Mutation	Crossover	Pop. size N_p
ACM	current-to-$p$best	ACM	inflated
RND	rand	ACM	inflated
5D	current-to-$p$best	ACM	$5D$
JADE	current-to-$p$best	JADE	$5D$

new better vector will take part in evolution without a time lag. In a sequential mode both generational and steady state variants of DE show similar performance in terms of probability of convergence and number of function evaluations to reach the minimum. But for parallel calculations Asynchronous DE exhibits faster speed-up as the number of computing nodes is increased [5].

2.1 Choice of a Target Vector

The choice of a *target* vector is a feature which emerges as soon as we switch from a generational algorithm to a steady state variant. In this article we will pick a random member from a population as a target vector $\vec{x}_i$. A faster convergence rate can be tried through a choice of either of the worst population members, thus enforcing replacement of not-so-good candidates. While this choice is profitable for some optimization problems, we found that it reduces the dispersion within the population, and usually leads to a lower probability of convergence to the global minimum. We found that in the case of restart strategies (see Sec. 2.4) the overall performance of ADE weakly depends on a particular choice of the target vector.

2.2 Mutation

In DE a *mutation* vector $\vec{v}_i$ is constructed by adding to a selected population member a scaled difference vector, which is formed by a simple difference between randomly picked vectors from the population. In this article we will analyze the following strategies:

$$\text{‘rand’ [1]:} \qquad \vec{v}_i = \vec{x}_s + F_i(\vec{x}_r - \vec{x}_q); \tag{2}$$

$$\text{‘current-to-}p\text{best’ [3]:} \qquad \vec{v}_i = \vec{x}_i + F_i(\vec{x}^{p}_{\text{best}} - \vec{x}_i) + F_i(\vec{x}_r - \vec{x}_q). \tag{3}$$

Here r and s denote random members of the population. The index q corresponds to a randomly selected $\vec{x}_q \in P \cup A$, with a so-called *archive* A, which stores N_p former population members recently discarded by selection. The index p is a random index within $0.1N_p$ best candidates. All vectors in the right sides of Eqs. (2–3) are enforced to be distinct. The *scale factor* F_i is sampled for each mutation according to a Cauchy distribution with the location parameter μ_F and the scale parameter $\sigma_F = 0.1$ [3]. Iff a trial vector $\vec{u}_i$ is selected to replace a target vector $\vec{x}_i$ (see Sec. 2.4), the location parameter is updated

$$\mu'_F = (1 - c_F)\mu_F + c_F L_2(\{F\}). \tag{4}$$

Here $c_F = 0.01$ is called a learning rate to update the location parameter, $L_2(\{F\})$ is a contraharmonic mean of a set of all scale factors associated with the current population.

2.3 Crossover

In DE the coordinates of the *trial* vector $u_{i,j}$ are picked either from the mutant vector $\vec{v}_i$ or from the target vector $\vec{x}_i$ – the so called *crossover* operation. We will compare the uniform crossover with the crossover rate C_r, adapted by the JADE scheme [3], to a crossover based on an adaptive correlation matrix (ACM) [4].

The coordinates of the trial vector $\vec{u}_i$ after uniform crossover are

$$u_{i,j} = \begin{cases} v_{i,j} & \text{rand}[0,1) < C_{r,i} \text{ or } j = j_{\text{rand}}, \\ x_{i,j} & \text{otherwise.} \end{cases} \tag{5}$$

Here $C_{r,i} \in [0,1]$ indicates an average proportion of coordinates selected from the mutant vector into the trial vector. To ensure distinct trial and target vectors, at least one coordinate j_{rand} is taken from the mutant vector. In JADE the rate $C_{r,i}$ is generated for each crossover according to a normal distribution with the mean μ_c and the standard deviation 0.1. The mean μ_c is updated as a result of successful iterations as

$$\mu_c' = (1 - c_\mu)\mu_c + c_\mu \langle \{C_r\} \rangle \tag{6}$$

with a learning factor $\mu_c = 0.01$, $\langle \{C_r\} \rangle$ is a mean over all crossover rates within the current population.

While the above JADE scheme treats all coordinates uniformly, ADE with the Adaptive Correlation Matrix [4] uses information of pairwise correlations between parameters. The current population is used to calculate a *sample correlation matrix* S, its elements read

$$s_{jk} = \frac{q_{jk}}{\sqrt{q_{jj} q_{kk}}}; \quad q_{jk} = \frac{1}{N_p - 1} \sum_{i=0}^{N_p - 1} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k). \tag{7}$$

Successful steps are used to cumulatively update an estimation of the correlation matrix – an *adaptive correlation matrix* C:

$$C' = (1 - c)C + cS. \tag{8}$$

The coefficient $c = 0.01$ is a *learning rate* for updating the correlation matrix. From the adaptive correlation matrix, the algorithm identifies a group of variables correlated to a selected variable m

$$\{I_m\} = \forall j : |c_{mj}| > c_{\text{thr}}, \quad c_{\text{thr}} = \text{rand}(0,1). \tag{9}$$

The above set of correlated variables $\{I_m\}$ defines a subspace Ω_m in the search domain. All components of the mutant vector $\vec{v}_i$, the indices of which are in the set $\{I_m\}$, are propagated into a trial vector $\vec{u}_i$, while other components are taken from the target vector $\vec{x}_i$:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } j \in \{I_m\}, \\ x_{i,j} & \text{otherwise.} \end{cases} \tag{10}$$

2.4 Selection and Restart

The Differential Evolution uses a greedy algorithm for the selection: a trial vector $\vec{u}_i$ will replace a target vector $\vec{x}_i$ iff there is an improvement in corresponding objective function values.

During successive iterations the algorithm analyzes spreads of population members in each coordinate Δx_j and in the function values Δf to avoid stagnation. If at least one of the spreads is too small

$$\exists j \quad \Delta x_j < \varepsilon_x \max_i \{|x_{i,j}|\},\ \varepsilon_x = 10^{-12} \quad \text{or} \quad \Delta f < \varepsilon_f \max_{i=0,\ldots,N_p-1} \{|f_i|\},\ \varepsilon_f = 10^{-11} \tag{11}$$

an independent restart is initiated [6]. In this work we analyze two restart strategies. One, named '5D', uses $N_p = 5D$ as a constant population size. Another strategy selects $N_p^{\min} = 10$ as an initial size of the population, at each restart the population is increased by a factor 2, if after inflation a population size exceeds $20D$, an independent restart with the size $N_p^{\min}$ is enforced.

3 Numerical Tests

The performance of several variants of ADE-ACM and JADE (Tab. 1) are compared on the set of real-parameter black-box optimization problems BBOB-2015 [7], which is a widely used test bench: more than 100 articles and algorithms have been benchmarked. The test bench contains 24 functions: separable, non-separable, weakly-structured, unimodal, multimodal, and/or ill-conditioned for dimensions 2, 3, 5, 10, 20 and 40. The performance is measured by the number of successful trials #succ, when an algorithm has reached the function values below $f^* + 10^{-8}$, and the *expected running time* $\mathrm{ERT}(\Delta f^*)$ to reach values better than $f^* + \Delta f^*$, which is calculated as a ratio of the sum of the evaluations number before the above target value has been reached over the number of successful trials. The maximal number of function evaluations is limited to $10^6 D$.

The graphical representation of ERT for all 4 variants is shown in Fig. 2. As the dimension of the problem is increased, ADE-ACM usually performs better than JADE. To exclude differences due to restart procedures, we cite results for ADE-ACM variant '5D' with the fixed population size, which differs from JADE only by the crossover operator. The better performance by ADE-ACM is mainly due to the new crossover, which takes into account the correlations between variables. Convergence rates are presented in Table 2 for the dimension $D = 20$. ADE-ACM algorithm solves 17 of 24 functions with probability higher than 0.5 within the allocated number of function evaluations. If restarts are used, both 'current-to-$p$best' and 'rand' strategies have similar performance. For two multimodal functions: Büche-Rastrigin (f_4) and Schwefel (f_{20}) ADE-ACM outperforms previously tested algorithms.

4 Conclusions

In this work we have compared the performance of the recently proposed minimization algorithm of Asynchronous Differential Evolution with Adaptive Correlation Matrix to the widely used JADE method. By using additional information about linear correlations between variables, which is learned during successful iterations, the new algorithm shows better convergence probabilities and faster convergence rates as the dimension D of the problem exceeds 10. The ADE-ACM can competitively solve a wide range of global optimization problems, both separable, non-separable or partially-separable thanks to the new type of crossover, based on the estimation of the correlation matrix. The new algorithm has a simple structure and is quasi parameter-free from the user's point of view.

References

[1] K. Price and R. Storn, J. of Global Optimization **11**, 341–359 (1997)
[2] S. Das and P. N. Suganthan, IEEE Trans. Evol. Comput. **15**, 4–31 (2011)
[3] J. Zhang and A.C. Sanderson, IEEE Trans. Evol. Comput. **13**, 945–958 (2009)
[4] E. Zhabitskaya and M. Zhabitsky, Proceedings of the 15th annual conference on Genetic and evolutionary computation, GECCO'13, ACM, 455–462 (2013)
[5] E. Zhabitskaya and M. Zhabitsky, Lect. Notes in Comp. Sci. **7125**, 328–333 (2012)
[6] E. Zhabitskaya and M. Zhabitsky, Lect. Notes in Comp. Sci. **8236**, 555–561 (2013)
[7] S. Finck, N. Hansen, R. Ros, and A.Auger, Black-Box optimization benchmarking, COCO Documentation, v15.02 (2015); `http://coco.gforge.inria.fr`

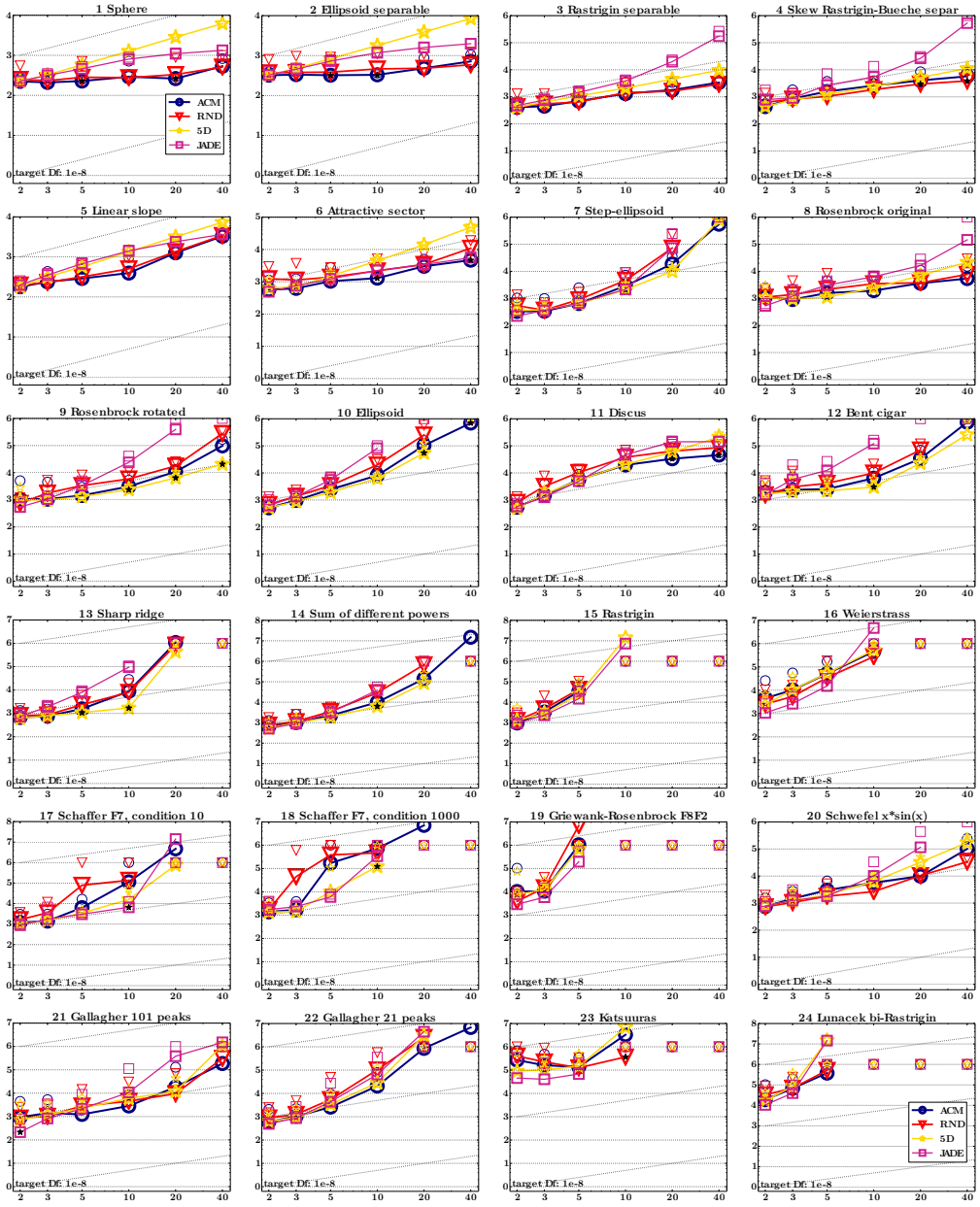

Figure 2: Expected running time (as $\log_{10}$ value of the number of function evaluations), divided by dimension, for target function value f^*+10^{-8}, versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms. Legend: ○: ACM, ▽: RND, ★: 5D, □: JADE

Table 2: Expected running time (in number of function evaluations) in the dimension $D = 20$. Different target Δf^*-values are shown in the top row, #succ is the number of trials that reached the final target $f^* + 10^{-8}$.

	Δf^*	1e1	1e-1	1e-3	1e-7	# succ
	ACM	6.0e2	1.6e3	**2.6e3**	**4.7e3**	15/15
f_1	R	**5.6e2**	**1.6e3**	2.8e3	5.9e3	15/15
	5D	3.4e3	1.5e4	2.7e4	5.1e4	15/15
	JADE	1.7e3	7.4e3	1.2e4	2.0e4	15/15
	ACM	3.5e3	4.6e3	6.2e3	9.0e3	15/15
f_2	R	**2.8e3**	**4.3e3**	**6.2e3**	**9.0e3**	15/15
	5D	2.5e4	3.6e4	4.8e4	7.2e4	15/15
	JADE	1.2e4	1.7e4	2.1e4	3.0e4	15/15
	ACM	1.5e4	2.7e4	2.9e4	3.4e4	15/15
f_3	R	**8.1e3**	**2.1e4**	**2.4e4**	**3.1e4**	15/15
	5D	2.8e4	4.7e4	6.0e4	8.4e4	15/15
	JADE	2.9e5	3.8e5	3.9e5	4.0e5	15/15
	ACM	1.7e4	6.1e4	6.7e4	7.7e4	15/15
f_4	R	**1.1e4**	**4.1e4**	**4.6e4**	**5.6e4**	15/15
	5D	3.4e4	5.5e4	6.9e4	9.5e4	15/15
	JADE	4.0e5	5.1e5	5.2e5	5.2e5	15/15
	ACM	3.2e3	**7.5e3**	**1.3e4**	**2.4e4**	15/15
f_5	R	**3.0e3**	1.1e4	1.6e4	2.6e4	15/15
	5D	1.0e4	2.3e4	3.4e4	5.8e4	15/15
	JADE	6.4e3	1.6e4	2.5e4	4.4e4	15/15
	ACM	7.8e3	1.8e4	**2.8e4**	**5.5e4**	15/15
f_6	R	**6.0e3**	**1.6e4**	3.1e4	5.6e4	15/15
	5D	3.1e4	8.9e4	1.5e5	2.5e5	15/15
	JADE	1.4e4	2.7e4	3.9e4	6.2e4	15/15
	ACM	1.9e4	2.8e5	3.6e5	3.7e5	15/15
f_7	R	1.4e4	1.0e6	1.4e6	1.4e6	15/15
	5D	1.6e4	**1.2e5**	**2.0e5**	**2.0e5**	15/15
	JADE	**1.1e4**	5.6e5	1.4e6	1.5e6	15/15
	ACM	**1.2e4**	4.8e4	6.3e4	7.2e4	15/15
f_8	R	1.6e4	**4.4e4**	**5.5e4**	**7.2e4**	15/15
	5D	5.5e4	1.0e5	1.1e5	1.3e5	15/15
	JADE	6.9e4	2.4e5	2.7e5	3.2e5	15/15
	ACM	**5.1e4**	1.5e5	1.7e5	2.0e5	15/15
f_9	R	8.8e4	2.1e5	2.6e5	3.4e5	15/15
	5D	6.0e4	**1.1e5**	**1.2e5**	**1.3e5**	15/15
	JADE	1.3e5	7.9e5	2.4e6	6.1e6	15/15
	ACM	5.4e5	1.1e6	1.3e6	1.9e6	15/15
f_{10}	R	1.2e6	2.7e6	3.4e6	4.9e6	15/15
	5D	**2.6e5**	**4.8e5**	**7.0e5**	**1.0e6**	15/15
	JADE	∞	∞	∞	∞	0/15
	ACM	**5.5e4**	**2.3e5**	**4.0e5**	**6.1e5**	15/15
f_{11}	R	9.8e4	3.1e5	5.6e5	1.1e6	15/15
	5D	9.5e4	4.0e5	6.2e5	1.1e6	15/15
	JADE	3.0e5	1.1e6	1.6e6	2.6e6	15/15
	ACM	**7.7e3**	1.4e5	4.1e5	6.5e5	15/15
f_{12}	R	1.5e4	4.2e5	9.1e5	1.4e6	15/15
	5D	3.0e4	**1.1e5**	**2.2e5**	**4.1e5**	15/15
	JADE	2.4e4	2.1e7	9.1e7	∞	0/15
	Δf^*	1e1	1e-1	1e-3	1e-7	# succ

	Δf^*	1e1	1e-1	1e-3	1e-7	# succ
	ACM	**1.8e4**	1.2e6	4.0e6	1.4e7	10/15
f_{13}	R	1.2e5	1.8e6	4.2e6	1.5e7	11/15
	5D	2.6e4	**4.4e4**	**2.1e5**	**8.0e6**	12/15
	JADE	1.2e5	2.6e6	7.4e6	∞	0/15
	ACM	**3.5e2**	2.3e3	**1.7e4**	1.7e6	15/15
f_{14}	R	3.8e2	**2.2e3**	1.8e4	8.5e6	15/15
	5D	1.9e3	1.8e4	8.3e4	**4.5e5**	15/15
	JADE	9.0e2	8.2e3	2.5e4	∞	0/15
	ACM	∞	∞	∞	∞	0/15
f_{15}	R	∞	∞	∞	∞	0/15
	5D	∞	∞	∞	∞	0/15
	JADE	**5.5e7**	∞	∞	∞	0/15
	ACM	**1.8e4**	2.9e7	∞	∞	0/15
f_{16}	R	1.9e4	**9.0e6**	**9.3e7**	**3.0e8**	1/15
	5D	4.0e4	2.6e7	∞	∞	0/15
	JADE	3.1e5	∞	∞	∞	0/15
	ACM	1.9e2	3.2e6	5.8e6	6.6e7	4/15
f_{17}	R	1.8e2	1.6e6	1.2e7	∞	0/15
	5D	2.3e2	2.7e5	8.9e5	**1.2e7**	10/15
	JADE	**1.7e2**	**2.7e4**	**6.1e4**	8.0e7	3/15
	ACM	1.1e4	1.4e7	2.9e7	**1.4e8**	2/15
f_{18}	R	**4.0e3**	1.5e7	9.6e7	∞	0/15
	5D	1.1e4	**4.1e5**	**2.2e6**	∞	0/15
	JADE	4.5e3	1.5e6	2.8e8	∞	0/15
	ACM	**1.5e2**	∞	∞	∞	0/15
f_{19}	R	2.1e2	∞	∞	∞	0/15
	5D	5.4e2	∞	∞	∞	0/15
	JADE	5.0e2	∞	∞	∞	0/15
	ACM	**5.2e2**	1.6e5	**1.7e5**	**1.7e5**	15/15
f_{20}	R	5.8e2	**1.6e5**	1.7e5	2.2e5	15/15
	5D	4.5e3	3.1e5	5.0e5	6.2e5	15/15
	JADE	2.2e3	1.4e6	2.4e6	2.3e6	15/15
	ACM	**1.9e3**	3.4e5	3.5e5	3.7e5	15/15
f_{21}	R	2.6e3	**1.3e5**	**1.5e5**	**1.8e5**	15/15
	5D	7.3e3	2.1e5	2.2e5	2.3e5	15/15
	JADE	3.6e3	7.6e6	7.6e6	7.7e6	15/15
	ACM	**9.3e3**	**1.1e7**	**1.3e7**	**1.7e7**	9/15
f_{22}	R	1.2e4	4.9e7	4.9e7	4.9e7	5/15
	5D	1.1e5	6.1e7	6.1e7	6.1e7	4/15
	JADE	1.6e5	8.9e7	8.9e7	8.9e7	3/15
	ACM	8.0e0	1.0e7	∞	∞	0/15
f_{23}	R	7.0e0	**5.0e6**	∞	∞	0/15
	5D	7.0e0	6.5e6	∞	∞	0/15
	JADE	5.1e0	∞	∞	∞	0/15
	ACM	2.8e8	∞	∞	∞	0/15
f_{24}	R	2.9e8	∞	∞	∞	0/15
	5D	**1.3e8**	∞	∞	∞	0/15
	JADE	∞	∞	∞	∞	0/15
	Δf^*	1e1	1e-1	1e-3	1e-7	# succ