JADE, an Adaptive Differential Evolution Algorithm, Benchmarked on the BBOB Noiseless Testbed

Petr Pošík
Czech Technical University in Prague
FEE, Dept. of Cybernetics
Technická 2, 166 27 Prague 6, Czech Republic
petr.posik@fel.cvut.cz

Václav Klemš
Czech Technical University in Prague
FEE, Dept. of Cybernetics
Technická 2, 166 27 Prague 6, Czech Republic

ABSTRACT
JADE, an adaptive version of the differential evolution (DE) algorithm, is benchmarked on the testbed of 24 noiseless functions chosen for the Black-Box Optimization Benchmarking workshop. The results of full-featured JADE are then compared with the results of 3 other DE variants ("downgraded" JADE variants) to reveal the contributions of the algorithm components. Another adaptive DE variant benchmarked during BBOB 2010 is used as a reference algorithm. The results confirm that the original JADE outperforms the other (JA)DE versions, while the comparison with the other adaptive DE shows that the different sources of adaptivity make the algorithms suitable for different functions.

Categories and Subject Descriptors
G.1.6 [Numerical Analysis]: Optimization—global optimization, unconstrained optimization; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems

General Terms
Algorithms

Keywords
Benchmarking, Black-box optimization, Differential evolution, Adaptation

1. INTRODUCTION
Differential Evolution (DE) [9] is a population-based optimization algorithm popular thanks to its simple structure and wide applicability. Similarly to other optimizers, it has a few parameters which must be properly chosen for the particular task being solved. This fact led to the birth of adaptive versions of DE [8, 1, 2] differing in (1) what they adapt and (2) how. For this article, we chose the JADE algorithm which was shown [10] to be more efficient than the approaches in [8, 1] on a set of several benchmark functions.

The purpose of this paper is to evaluate the performance of the JADE algorithm using the COCO framework [5] and to assess the benefits of its individual parts. We also compare the JADE algorithm against the DE-F-AUC [2], another adaptive DE benchmarked in the COCO framework recently.

In Sec. 2 we briefly reiterate the DE algorithm and describe the JADE algorithm in more detail. In Sec. 3, we present the experiment design together with the algorithm parameters settings. Sec. 4 then presents the results and Sec. 5 discusses them.

2. ALGORITHM PRESENTATION
Differential Evolution (DE) [9] is a population-based optimization algorithm. Each generation, for each population member \( x_i \) (the parent), a donor \( v_i \) is created using a mutation operator. The donor \( v_i \) is then crossed over with its parent \( x_i \) to create the offspring \( u_i \). The offspring \( u_i \) then replaces its parent \( x_i \), if it is better.

DE mutation operators create the donor individual \( v_i \) as a linear combination of several individuals in the current population. Eq. 1 describes one of the possible mutation operators, the so called "best" mutation operator:

\[
v_i = x_{\text{best}} + F \cdot (x_{1} - x_{2}),
\]  

where \( F \) is the mutation factor (a positive number typically chosen from \([0.5, 1]\)) and \( x_{1} \) and \( x_{2} \) are randomly chosen population members.

The crossover creates the offspring \( u_i \) by taking some solution components from the parent \( x_i \) and other components from the donor \( v_i \). Eq. (2) describes the binomial crossover:

\[
u_{i,j} = \begin{cases} 
  v_{i,j} & \text{if } r_{j} \leq CR_{i} \text{ or } j = j_{i,\text{rand}}, \\
  x_{i,j} & \text{otherwise},
\end{cases}
\]

where \( r_{j} \) is a random number uniformly distributed in \([0, 1]\), \( CR_{i} \in [0, 1] \) is the crossover probability representing the average proportion of components the offspring gets from its donor, and \( j_{i,\text{rand}} \) is the randomly chosen index of the solution component surely taken from the donor.

JADE [10] is an adaptive version of DE. It was shown to have better performance than other adaptive DE versions (jDE, SaDE) on many benchmark functions. It uses a simple form of adaptation, see Alg. 1. The \( \leftarrow \) symbol represents the assignment, while the \( \rightarrow \) symbol means addition of a new member to a set . The functions \( r_{\text{n}} \) and \( r_{\text{c}} \) are Gaussian and Cauchy random number generators, respectively, while \( \text{meanA} \) and \( \text{meanL} \) designate the arithmetic and Lehmer (contraharmonic) mean, respectively.
Algorithm 1: JADE

1. Set $\muCR \leftarrow 0.5$, $\muF \leftarrow 0.5$, archive $A \leftarrow \emptyset$.
2. Initialize the population $\{x_i\}_{i=1}^{NP}$.
3. for $g \leftarrow 1$ to $G$ do
   4. $S_F \leftarrow \emptyset$; $S_{CR} \leftarrow \emptyset$.
   5. for $i \leftarrow 1$ to $NP$ do
      6. $F_i \leftarrow r\epsilon(\muF, 0.1)$, $CR_i \leftarrow r\epsilon(\muCR, 0.1)$.
      7. $v_i \leftarrow \text{mutate}(x_i)$ (Eq. 3).
      8. $u_i \leftarrow \text{crossover}(x_i, v_i)$ (Eq. 2).
      9. if $f(u_i) < f(x_i)$ then
         10. $x_i \rightarrow A$; $CR_i \rightarrow S_{CR}$; $F_i \rightarrow S_F$.
         11. $x_i \leftarrow u_i$.
   12. Randomly remove members of $A$ while $|A| > NP$.
   13. $\muCR \leftarrow (1 - c) \cdot \muCR + c \cdot \text{mean}(S_{CR})$.
   14. $\muF \leftarrow (1 - c) \cdot \muF + c \cdot \text{mean}(S_F)$.

JADE differs from DE in 3 aspects. First, JADE can optionally use an archive of parent solutions recently replaced with more successful offspring. The archive is used in the JADE mutation operator.

The second difference from DE is a special mutation operator called "current-to-pbest":

$$v_i = x_i + F_i \cdot (x_{p_{best}}^i - x_i) + F_i \cdot (x_1 - x_2),$$

(3)

where $x_i$ is the parent individual, $x_{p_{best}}^i$ is an individual randomly chosen from the best 100p% individuals in the current population, $p \in [0, 1]$, $x_1$ and $x_2$ are individuals randomly chosen from the population and from the union of the current population and the archive, respectively. The $F_i$ is the mutation factor. The individuals $x_{p_{best}}^i$, $x_1$, and $x_2$, and the value of $F_i$ are chosen anew for each mutation.

The third and most important difference is the adaptation of $F$ and $CR$. In classic DE, both factors are usually constant (or sampled from a static distribution). In JADE, the crossover probability $CR_i$ is sampled from a normal distribution with mean $\muCR$ and standard deviation of 0.1. Similarly, $F_i$ is sampled from a Cauchy distribution with the location parameter $\muF$ and scale parameter 0.1. The parameters $\muCR$ and $\muF$ are updated each generation using the arithmetic and contraharmonic mean, respectively, of the $CR_i$ and $F_i$ values used to create the successful offspring individuals (successful = better than the respective parent).

DE-F-AUC is a DE algorithm able to choose among several (4 in this case) available mutation strategies based on their previous success using a technique called F-AUC-Bandit [3]. The results from BBOB 2010 article [2] are used. The algorithm does not contain the crossover operator and relies only on the rotationally invariant mutations.

3. EXPERIMENT DESIGN

The goal of the experiment is to assess the benefits of (1) using the "current-to-pbest" mutation strategy (referred to also as "ctpb") as opposed to the "best" strategy, and (2) using the JADE parameter adaptation. We thus designed 4 algorithms:

1. JADEctpb, adaptive with "ctpb" (the original JADE),
2. JADEb, adaptive with "best" (a downgraded JADE),
3. DEctpb, non-adaptive with "ctpb", and
4. DEb, non-adaptive with "best" (a conventional DE).

The evaluations budget was set to $5 \cdot 10^4D$ for each run. For most of the parameters, default values from the literature were used. For DE: $CR = 0.5$, $F \sim U(0.5, 1)$ (sampled anew each generation). For JADE; initial $\muCR = 0.5$, initial $\muF = 0.5$, $p = 0.1$, $|A| = 0.1NP$. The population size was set to $NP = 5D$ for all 4 algorithms after a small systematic study performed on JADEctpb and DEb using the values (3, 4, 5, 6, 8, 10, 15, 20) $\cdot D$. Values lower than 5D gave erratic behavior even on uni-modal functions, values larger than 5D wasted evaluations on uni-modal functions and did not bring significant advantages on multi-modal functions. All algorithms were restarted when they stagnate for more than 30 generations and the population diversity measure $\frac{1}{N} \sum_{i=1}^{D} \text{Var}(X_i) < 10^{-10}$.

4. RESULTS

Results from experiments according to [5] on the benchmark functions given in [4, 6] are presented in Figures 1, 2 and 3 and in Tables 1 and 2. The expected running time (ERT) used in the figures and table, depends on a given target function value, $f_t = f_{max} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach $f_t$, summed over all trials and divided by the number of trials that actually reached $f_t$ [5, 7]. Statistical significance is tested with the rank-sum test for a given target $\Delta f_i$ ($10^{-8}$ as in Figure 1) using, for each trial, either the number of needed function evaluations to reach $\Delta f_i$ (inverted and multiplied by $-1$), or, if the target was not reached, the best $\Delta f_i$-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

4.1 CPU Timing Experiments

The timing experiments were carried out with $f_8$ on a machine with Intel Core 2 Duo processor, 2.4 Ghz, with 4 GB RAM, on Windows 7 64bit in MATLAB R2009b 64bit. The average time per function evaluation in 2, 3, 5, 10, 20, 40 dimensions was about 52, 35, 21, 12, 8, and 7 $\times 10^{-6}$ s for both DE variants, and about 70, 45, 28, 16, 9, 10 $\times 10^{-6}$ s for both JADE variants.

5. DISCUSSION

Influence of the Mutation Strategy. By comparing the algorithm pairs DEb vs. DEctpb and JADEb vs. JADEctpb, we can make some observation about the influence of the chosen mutation strategy. Generally speaking, the "best" strategy is very exploitative, it allows the algorithm to converge (and lose diversity) faster, while the "current-to-pbest" strategy preserves more diversity in the population which in turn can prevent the algorithm from restarting more often.

Regarding DEb, on uni-modal functions, there is usually not much difference between these two mutation strategies with the exception of the functions $f_6$ and $f_7$ where the increased diversity due to the "ctpb" strategy allowed the DE algorithm to solve these problems faster in dimensions $\geq 10$. For the multi-modal functions, the results are mixed: sometimes it is better to restart more often (and the "best" strategy allows for this), while sometimes the better preserved diversity ensures better results than restarts (and then the "ctpb" strategy works better).
Figure 1: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value $10^{-8}$ as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of $f_1$ and $f_{24}$. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Horizontal lines give linear scaling, slanted dotted lines give quadratic scaling. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ◦: DE-F-AUC, ▽: DEb, ★: DEcpte, □: JADEb, △: JADEcpte.
Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{1-8.2}$ for all functions and subgroups in 5-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.
Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{1-8.2}$ for all functions and subgroups in 20-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.
Table 1: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different $\Delta f$ values in dimension 5. The central 80% range divided by two is given in italics. The median number of conducted function evaluations is additionally given in italics, if ERT($10^{-5}$) = $\infty$. #succ is the number of trials that reached the final target $f_{opt} + 10^{-8}$.
Best results are printed in bold.
Table 2: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBBO-2009 (given in the respective first row) for different $\Delta f$ values in dimension 20. The central 80% range divided by two is given in braces. The median number of conducted function evaluation trials is additionally given in italics, if ERT(10$^{-2}$) = $\infty$. #succ is the number of trials that reached the final target $f_{opt} + 10^{-5}$. Best results are printed in bold.
Regarding the JADE algorithm, for dimensions $\leq 5$, the two strategies work similarly well in terms of the ERT needed to find $\Delta f = 10^{-8}$. The group of multi-modal functions is an exception where the JADEb algorithm was successful for problems related to larger number of functions and the bootstrapping procedure emphasized this fact. In larger dimensions, the difference is more pronounced and the “ctpb” strategy provides equal or better results in the vast majority of cases.

**Influence of the Parameter Adaptation.** Comparing the two variants of JADE with the two variants of DE reveals the pros and cons of the parameter adaptation as done in JADE. The JADE variant works significantly worse than JADEctpb for several functions with $D \geq 10$, while the opposite is only seldom true. The results of JADEctpb compared to both variants of DE are more consistent. Generally speaking, the parameter adaptation as done in JADE is profitable—it reached comparable or better results than both DE variants. The seldom cases where JADEctpb is boldly worse than any of the DEs are $f_1$ and $f_2$ which are probably misleading for the adaptation process and the static parameter settings used by DE is a better choice.

While in low-dimensional spaces, the results for JADEctpb are mixed, the results in 20D space suggest that JADEctpb is able to solve the largest proportion of functions using the smallest number of function evaluations among the two JADE and two DE variants.

**Comparison with DE-F-AUC.** On uni-modal functions, DE-F-AUC is a competent solver and is generally comparable or better than the JADE algorithm, especially in larger dimensions. The cases where DE-F-AUC is slower than JADE can be attributed to the 2 times larger population of DE-F-AUC, or to the initial adaptation phase.

On multi-modal functions, however, the results are not that clear. The DE-F-AUC algorithm misses the crossover operator which is a serious drawback in case of separable functions (see the results for $f_3$ and $f_4$). On non-separable functions, the results are mixed. DE-F-AUC is better for $f_{15}$, $f_{17}$, and $f_{18}$ (i.e. the group called “multi-modal” functions), while JADEctpb is better for $f_{20}$, $f_{21}$, and $f_{22}$ (i.e. the group of “multi-modal functions with weak structure”). The difference may be partially caused by the missing crossover operator, however, the exact cause remains to be investigated. The results over all functions in 20D suggest that JADEctpb is at least comparable to the DE-F-AUC.

6. SUMMARY AND CONCLUSIONS

We benchmarked the JADE algorithm, an adaptive version of DE, and compared it to a classic DE. JADE uses a different mutation operator and adapts its mutation and crossover parameters $F$ and $CR$. We assessed the influence of these two features. As another reference algorithm, DE-F-AUC—yet another adaptive DE variant benchmarked during BBOB 2010—was chosen.

The results for low-dimensional spaces ($D \leq 5$) were indiscernible, perhaps with the exception of the ill-conditioned functions where the non-adaptive DE variants were 2 to 10 times slower than the rest. In higher-dimensional spaces, the original JADE algorithm (here called JADEctpb) was more successful than its opponents, and comparable to the reference DE-F-AUC algorithm (which looses some “points” due to the absence of the crossover operator and its subsequent inability to solve separable problems efficiently).

The two adaptive DE variants, JADE and DE-F-AUC, use different sources of adaptivity: while JADE adapts only the strategy parameters, DE-F-AUC adapts the use of different strategies. The potential join of these algorithms remains to be investigated as a future work.

7. REFERENCES


