Supplemental Materials for "How Far Are We From an Optimal, Adaptive DE?"

Ryoji Tanabe[1] and Alex Fukunaga[2]

[1] Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency
[2] Graduate School of Arts and Sciences, The University of Tokyo

A Parameter adaptation methods in adaptive DE algorithms

This section describes five parameter adaptation methods of the scale factor $F \in (0, 1]$ and the crossover rate $C \in [0, 1]$ in representative adaptive DE algorithms (jDE [1], EPSDE [4], JADE [8], MDE [3] and SHADE [6]). Note that below, we describe *parameter adaptation methods* in adaptive DEs, *not adaptive DE itself* (see Section 2). Here, we say that a generation of trial vector is *successful* if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$[3]. Otherwise, we say that it is a *failure*.

A.1 jDE

jDE [1] assigns a different set of parameter values $F_{i,t}$ and $C_{i,t}$ to each $\boldsymbol{x}^{i,t}$ in $\boldsymbol{P}^t$. For $t = 1$, the parameters for all individuals $\boldsymbol{x}^{i,t}$ are set to $F_{i,t} = 0.5$ and $C_{i,t} = 0.9$. In each generation t, each parameter is randomly modified (within a pre-specified range) with some probability:

$$F'_{i,t} = \begin{cases} \mathrm{rand}[0.1, 1] & \text{if } \mathrm{rand}[0, 1] < \tau_F \\ F_{i,t} & \text{otherwise} \end{cases} \tag{1}$$

$$C'_{i,t} = \begin{cases} \mathrm{rand}[0, 1] & \text{if } \mathrm{rand}[0, 1] < \tau_C \\ C_{i,t} & \text{otherwise} \end{cases} \tag{2}$$

Where, τ_F and $\tau_C \in (0, 1]$ are control parameters for parameter adaptation. Each individual $\boldsymbol{x}^{i,t}$ generates the trial vector using $F'_{i,t}$ and $C'_{i,t}$. $F'_{i,t}$ and $C'_{i,t}$ are kept for the next generation (i.e., $F_{i,t+1} = F'_{i,t}$ and $C_{i,t+1} = C'_{i,t}$) only when a trial is successful.

The overall generalized parameter adaptation method in jDE is described in Algorithm 2.

[3] Without loss of generality, we deal with minimization problems in tihs paper.

Algorithm 2: The generalized parameter adaptation method in jDE

$t \leftarrow 1$, initialize $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ randomly;
$F_{i,t} = 0.5$, $C_{i,t} = 0.9$, $i \in \{1, ..., N\}$;
while *The termination criteria are not met* **do**
    for $i = 1$ **to** N **do**
        if $\text{rand}[0, 1] \leq \tau_F$ **then**
            $F'_{i,t} = \text{rand}[0.1, 1]$;
        else
            $F'_{i,t} = F_{i,t}$;
        if $\text{rand}[0, 1] \leq \tau_C$ **then**
            $C'_{i,t} = \text{rand}[0, 1]$;
        else
            $C'_{i,t} = C_{i,t}$;
        Generate the mutant vector $\boldsymbol{v}^{i,t}$ using an arbitrary mutation strategy (e.g., rand/1, best/1 and current-to-$p$best/1 [8]) with $F'_{i,t}$;
        Generate the trial vector $\boldsymbol{u}^{i,t}$ by crossing $\boldsymbol{x}^{i,t}$ and $\boldsymbol{v}^{i,t}$ using an arbitrary crossover method (e.g., binomial, exponential, and eigenvector-based crossover [2]) with $C'_{i,t}$;
    for $i = 1$ **to** N **do**
        if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t+1} = F'_{i,t}$, $C_{i,t+1} = C'_{i,t}$;
        else
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$, $F_{i,t+1} = F_{i,t}$, $C_{i,t+1} = C_{i,t}$;
    $t \leftarrow t + 1$;

A.2 EPSDE

EPSDE [4] uses F-pool and C-pool for parameter adaptation of F and C respectively. The F-pool stores the F values as $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, and the C-pool includes the C values as $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. At the beginning of the search, each individual $\boldsymbol{x}_{i,t}$ is randomly assigned values for $F_{i,t}$ and $C_{i,t}$ from each pool. During search, successful parameter sets are inherited by the individual in the next generation. Parameter sets that fail are reinitialized.

The overall generalized parameter adaptation method in EPSDE is described in Algorithm 3.

A.3 JADE

JADE [8] uses two adaptive meta-parameters $\mu_F \in (0, 1]$ and $\mu_C \in [0, 1]$ for parameter adaptation. At the beginning of the search, μ_F and μ_C are both initialized to 0.5, and adapted during the search. In each generation t, $F_{i,t}$ and

Algorithm 3: The generalized parameter adaptation method in EPSDE

$t \leftarrow 1$, initialize $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ randomly;
F-pool $= \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$;
C-pool $= \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$;
For each individual $\boldsymbol{x}^{i,t}$, assign the $F_{i,t}$ and $C_{i,t}$ values randomly from each pool;
while *The termination criteria are not met* **do**
  for $i = 1$ **to** N **do**
    Generate the mutant vector $\boldsymbol{v}^{i,t}$ using an arbitrary mutation strategy (e.g., rand/1, best/1 and current-to-$p$best/1 [8]) with $F_{i,t}$;
    Generate the trial vector $\boldsymbol{u}^{i,t}$ by crossing $\boldsymbol{x}^{i,t}$ and $\boldsymbol{v}^{i,t}$ using an arbitrary crossover method (e.g., binomial, exponential, and eigenvector-based crossover [2]) with $C_{i,t}$;
  for $i = 1$ **to** N **do**
    if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t+1} = F_{i,t}, C_{i,t+1} = C_{i,t}$;
    else
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
      For $\boldsymbol{x}^{i,t+1}$, reassign the $F_{i,t+1}$ and $C_{i,t+1}$ values randomly from each pool;
  $t \leftarrow t + 1$;

$C_{i,t}$ are generated according to the following equations:

$$F_{i,t} = \mathrm{randc}(\mu_F, 0.1) \tag{3}$$

$$C_{i,t} = \mathrm{randn}(\mu_C, 0.1) \tag{4}$$

Here, $\mathrm{randc}(\mu_F, \sigma)$ is values selected randomly from Cauchy distributions with location μ_F and scale parameters σ. $\mathrm{randn}(\mu_C, \sigma^2)$ is values selected randomly from normal distributions with mean μ_C and variance σ^2. When $F_{i,t} > 1$, $F_{i,t}$ is truncated to 1, and when $F_{i,t} \leq 0$, Eq. (3) is repeatedly applied to try to generate a valid value. In case a value for $C_{i,t}$ outside of $[0, 1]$ is generated, it is replaced by the limit value (0 or 1) closest to the generated value.

In each generation, successful parameter sets F and C are recorded as $\boldsymbol{S}^F$ and $\boldsymbol{S}^C$, and at the end of the generation, μ_F and μ_C are updated as:

$$\mu_F \leftarrow (1 - c)\, \mu_F + c\, \mathrm{mean}_L(\boldsymbol{S}^F) \tag{5}$$

$$\mu_C \leftarrow (1 - c)\, \mu_C + c\, \mathrm{mean}_A(\boldsymbol{S}^C) \tag{6}$$

Here, the meta-level control parameter $c \in [0, 1]$ is a learning rate. $\mathrm{mean}_A(\cdot)$ is an arithmetic mean, and $\mathrm{mean}_L(\cdot)$ is a Lehmer mean which is computed as:

$$\mathrm{mean}_L(\boldsymbol{S}) = \frac{\sum_{s \in \boldsymbol{S}} s^2}{\sum_{s \in \boldsymbol{S}} s} \tag{7}$$

Algorithm 4: The generalized parameter adaptation method in JADE

$t \leftarrow 1$, initialize $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ randomly;
$\mu_F \leftarrow 0.5$, $\mu_C \leftarrow 0.5$;
while *The termination criteria are not met* **do**
  $\boldsymbol{S}^F \leftarrow \emptyset$, $\boldsymbol{S}^C \leftarrow \emptyset$;
  for $i = 1$ **to** N **do**
    $F_{i,t} = \mathrm{randc}(\mu_F, 0.1)$;
    $C_{i,t} = \mathrm{randn}(\mu_C, 0.1)$;
    Generate the mutant vector $\boldsymbol{v}^{i,t}$ using an arbitrary mutation strategy (e.g., rand/1, best/1 and current-to-$p$best/1 [8]) with $F_{i,t}$;
    Generate the trial vector $\boldsymbol{u}^{i,t}$ by crossing $\boldsymbol{x}^{i,t}$ and $\boldsymbol{v}^{i,t}$ using an arbitrary crossover method (e.g., binomial, exponential, and eigenvector-based crossover [2]) with $C_{i,t}$;
  for $i = 1$ **to** N **do**
    if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$;
    else
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
  if $\boldsymbol{S}^F, \boldsymbol{S}^C \neq \emptyset$ **then**
    $\mu_F \leftarrow (1-c) \cdot \mu_F + c \cdot \mathrm{mean}_L(\boldsymbol{S}^F)$;
    $\mu_C \leftarrow (1-c) \cdot \mu_C + c \cdot \mathrm{mean}_A(\boldsymbol{S}^C)$;
  $t \leftarrow t + 1$;

$\boldsymbol{S}$ refers to either $\boldsymbol{S}^F$ or $\boldsymbol{S}^C$. As the search progresses, μ_F and μ_C should gradually approach the appropriate values for the given problem.

The overall generalized parameter adaptation method in JADE is described in Algorithm 4.

A.4 MDE

A parameter adaptation method in MDE [3] is similar to JADE, and uses the meta-parameters μ_F and μ_C for parameter adaptation of F and C respectively. In each generation t, $F_{i,t}$ and $C_{i,t}$ are generated according to the equations (3) and (4) respectively. At the end of each generation, μ_F and μ_C are updated as:

$$\mu_F \leftarrow (1 - c_F)\, \mu_F + c_F\, \mathrm{mean}_P(\boldsymbol{S}^F) \tag{8}$$

$$\mu_C \leftarrow (1 - c_C)\, \mu_C + c_C\, \mathrm{mean}_P(\boldsymbol{S}^C) \tag{9}$$

Where, c_F and c_C are uniformly selected random real numbers from $(0.0, 0.2]$ and $(0.0, 0.1]$ respectively. In contrast to JADE, the learning rate c_F and c_C are randomly assigned in each generation t. $\mathrm{mean}_P(\cdot)$ is power mean with $n = 1.5$ as follows:

$$\mathrm{mean}_P(\boldsymbol{S}) = \Big(\frac{1}{|\boldsymbol{S}|} \sum_{s \in \boldsymbol{S}} s^n\Big)^{\frac{1}{n}} \tag{10}$$

Algorithm 5: The generalized parameter adaptation method in MDE

$t \leftarrow 1$, initialize $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ randomly;
$\mu_F \leftarrow 0.5$, $\mu_C \leftarrow 0.5$;
while *The termination criteria are not met* **do**
    $\boldsymbol{S}^F \leftarrow \emptyset$, $\boldsymbol{S}^C \leftarrow \emptyset$;
    for $i = 1$ **to** N **do**
        $F_{i,t} = \mathrm{randc}(\mu_F, 0.1)$;
        $C_{i,t} = \mathrm{randn}(\mu_C, 0.1)$;
        Generate the mutant vector $\boldsymbol{v}^{i,t}$ using an arbitrary mutation strategy (e.g., rand/1, best/1 and current-to-$p$best/1 [8]) with $F_{i,t}$;
        Generate the trial vector $\boldsymbol{u}^{i,t}$ by crossing $\boldsymbol{x}^{i,t}$ and $\boldsymbol{v}^{i,t}$ using an arbitrary crossover method (e.g., binomial, exponential, and eigenvector-based crossover [2]) with $C_{i,t}$;
    for $i = 1$ **to** N **do**
        if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$;
        else
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
    if $\boldsymbol{S}^F, \boldsymbol{S}^C \neq \emptyset$ **then**
        $c_F \leftarrow \mathrm{rand}(0.0, 0.2]$, $c_C \leftarrow \mathrm{rand}(0.0, 0.1]$;
        $\mu_F \leftarrow (1 - c_F) \cdot \mu_F + c_F \cdot \mathrm{mean}_P(\boldsymbol{S}^F)$;
        $\mu_C \leftarrow (1 - c_C) \cdot \mu_C + c_C \cdot \mathrm{mean}_P(\boldsymbol{S}^C)$;
    $t \leftarrow t + 1$;

The overall generalized parameter adaptation method in MDE is described in Algorithm 5.

A.5 SHADE

SHADE [6,7] uses historical memories $\boldsymbol{M}^F$ and $\boldsymbol{M}^C$ for parameter adaption of F and C, where $\boldsymbol{M}^F = (M_1^F, ..., M_H^F)$ and $\boldsymbol{M}^C = (M_1^C, ..., M_H^C)$. H is a memory size, and the all elements in $\boldsymbol{M}^F$ and $\boldsymbol{M}^C$ are initialized to 0.5.

In each generation t, the control parameters $F_{i,t}$ and $C_{i,t}$ used by each individual $\boldsymbol{x}^{i,t}$ are generated by randomly selecting an index r_i from $\{1, ..., H\}$, and then applying the formulas below:

$$F_{i,t} = \mathrm{randc}(M_{r_i}^F, 0.1) \tag{11}$$

$$C_{i,t} = \mathrm{randn}(M_{r_i}^C, 0.1) \tag{12}$$

If the values generated for F_i and C_i are outside the range $[0, 1]$, they are adjusted/regenerated according to the procedure described above for JADE.

Algorithm 6: The generalized parameter adaptation method in SHADE

$t \leftarrow 1$, initialize $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ randomly;
Set all values in $\boldsymbol{M}^F = (M_1^F, ..., M_H^F)$, $\boldsymbol{M}^C = (M_1^C, ..., M_H^C)$ to 0.5;
$k \leftarrow 1$;
while *The termination criteria are not met* **do**
    $\boldsymbol{S}^F \leftarrow \emptyset$, $\boldsymbol{S}^C \leftarrow \emptyset$;
    for $i = 1$ **to** N **do**
        Select the memory index r_i from $\{1, ..., H\}$ randomly;
        $F_{i,t} = \mathrm{randc}(M_{r_i}^F, 0.1)$;
        $C_{i,t} = \mathrm{randn}(M_{r_i}^C, 0.1)$;
        Generate the mutant vector $\boldsymbol{v}^{i,t}$ using an arbitrary mutation strategy (e.g., rand/1, best/1 and current-to-$p$best/1 [8]) with $F_{i,t}$;
        Generate the trial vector $\boldsymbol{u}^{i,t}$ by crossing $\boldsymbol{x}^{i,t}$ and $\boldsymbol{v}^{i,t}$ using an arbitrary crossover method (e.g., binomial, exponential, and eigenvector-based crossover [2]) with $C_{i,t}$;
    for $i = 1$ **to** N **do**
        if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$;
        else
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
    if $\boldsymbol{S}^F, \boldsymbol{S}^C \neq \emptyset$ **then**
        $M_k^F \leftarrow \mathrm{mean}_L(\boldsymbol{S}^F)$;
        $M_k^C \leftarrow \mathrm{mean}_L(\boldsymbol{S}^C)$;
        $k \leftarrow (k \text{ modulo } H) + 1$;
    $t \leftarrow t + 1$;

At the end of the generation, the memory contents in $\boldsymbol{M}^F$ and $\boldsymbol{M}^C$ are updated using Lehmer mean as follows:

$$M_k^F \leftarrow \mathrm{mean}_L(\boldsymbol{S}^F) \quad (13)$$

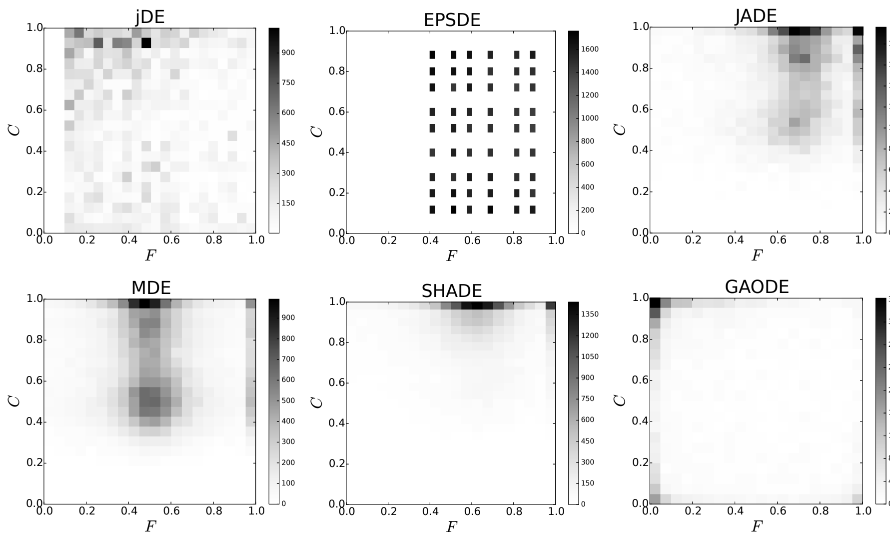
$$M_k^C \leftarrow \mathrm{mean}_L(\boldsymbol{S}^C) \quad (14)$$

An index $k \in \{1, ..., H\}$ determines the position in the memory to update. At the beginning of the search k is initialized to 1. k is incremented whenever a new element is inserted into the history. If $k > H$, k is set to 1.

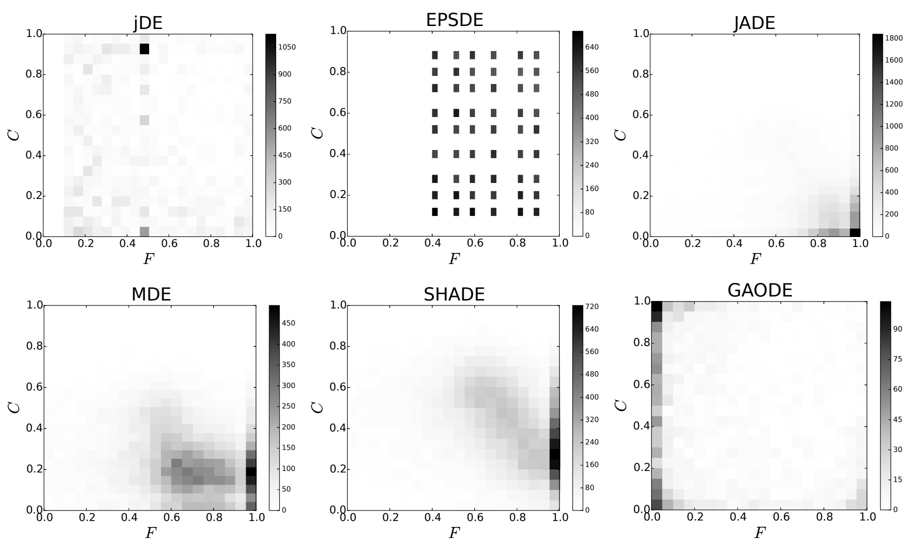
The overall generalized parameter adaptation method in SHADE is described in Algorithm 6.

Table A.1: Six popular benchmark functions used in Section 3. $\boldsymbol{z} = (z_1, ..., z_D)^{\mathrm{T}}$, $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$, and for each function, the location of the global optimal solution has been shifted by offset $\boldsymbol{o} = (o_1, ..., o_D)^{\mathrm{T}}$, where each component of $\boldsymbol{o}$ is a uniformly generated random offset. The optimal solution locates $\boldsymbol{z} = (0, ..., 0)^{\mathrm{T}}$, and its objective function value is 0 for all functions. The $D \times D$ rotation matrix $\boldsymbol{R}$ in the Rotated-Ellipsoid function is uniformly generated according to Salomon's method [5].

Functions	Definitions	Range
Sphere	$f_{\mathrm{Sphere}}(\boldsymbol{x}) = \sum_{i=1}^{D} z_i^2$	$[-100, 100]^D$
Ellipsoid	$f_{\mathrm{Ellipsoid}}(\boldsymbol{x}) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}}(z_i^2)$	$[-5, 5]^D$
Rotated-Ellipsoid	$f_{\mathrm{Rot.Ellipsoid}}(\boldsymbol{x}) = f_{\mathrm{Ellipsoid}}(\boldsymbol{R}\boldsymbol{x})$	$[-5, 5]^D$
Rosenbrock	$f_{\mathrm{Rosenbrock}}(\boldsymbol{x}) = \sum_{i=1}^{D-1}\left(100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2\right)$	$[-30, 30]^D$
Ackley	$f_{\mathrm{Ackley}}(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + 20 + \mathrm{e}$	$[-32, 32]^D$
Rastrigin	$f_{\mathrm{Rastrigin}}(\boldsymbol{x}) = \sum_{i=1}^{D}\left(z_i^2 - 10\cos(2\pi z_i) + 10\right)$	$[-5.12, 5.12]^D$

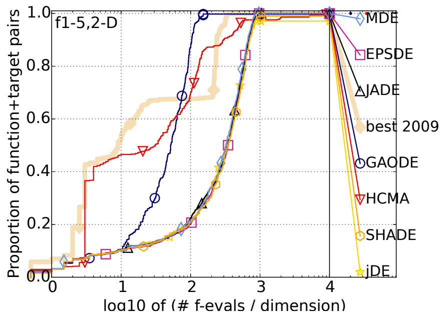


(a) Rosenbrock ($D = 10$)

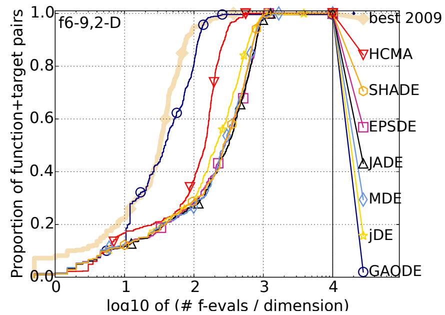


(b) Rastrigin ($D = 10$)

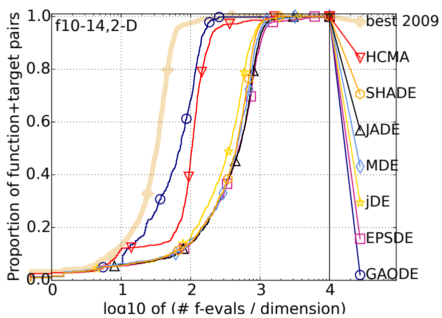
Fig. A.1: The frequency of appearance of $\{F, C\}$ value pairs during the search process for jDE, EPSDE, JADE, MDE, SHADE and GAO on the 10-dimensional (a) Rosenbrock and (b) Rastrigin functions. Since JADE had a success rate of 0 on the Rosenbrock function for 10-dimensions, we show the result for 5-dimensions. Darker colors indicate more frequent generation of the corresponding values by the parameter adaptation method. Data from the best run out of 51 runs is shown.



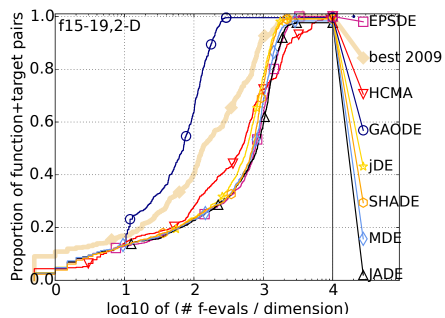
(a) $f_1 \sim f_5$ (separable fcts)



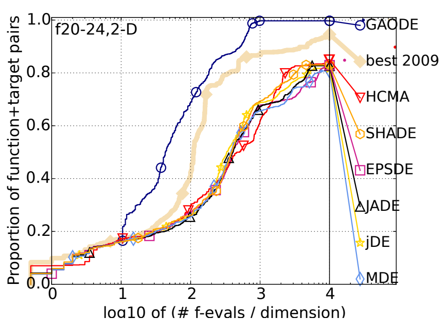
(b) $f_6 \sim f_9$ (moderate fcts)



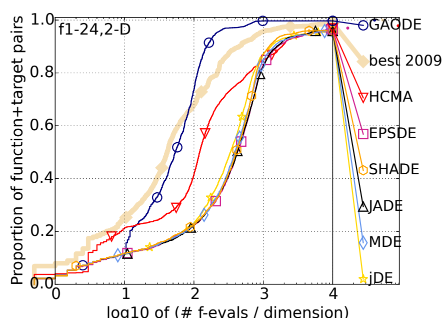
(c) $f_{10} \sim f_{14}$ (ill-conditioned fcts)



(d) $f_{15} \sim f_{19}$ (multi-modal fcts)

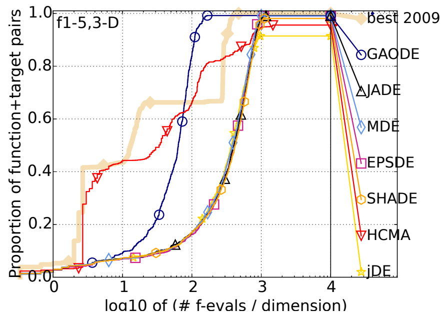


(e) $f_{20} \sim f_{24}$ (weakly structured multi-modal fcts)

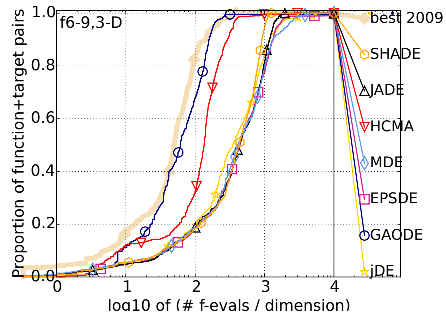


(f) $f_1 \sim f_{24}$ (all functions)

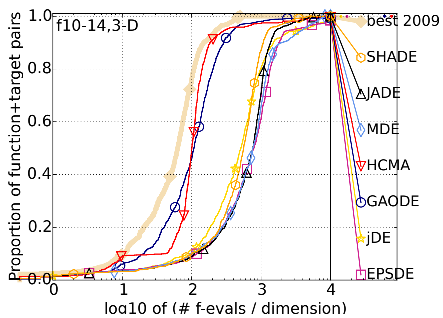
Fig. A.2: Comparisons of GAODE with the adaptive DE variants, HCMA, and best-2009 on BBOB benchmarks ($D = 2$). These figures show bootstrapped Empirical Cumulative Distribution Function (ECDF) of the FEvals divided by dimension for 50 targets in $10^{[-8..2]}$ for each function class (higher than better). For details of the ECDF, see a manual of `COCO software` (http://coco.gforge.inria.fr/).



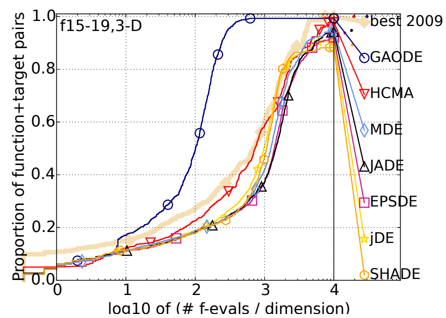
(a) $f_1 \sim f_5$ (separable fcts)



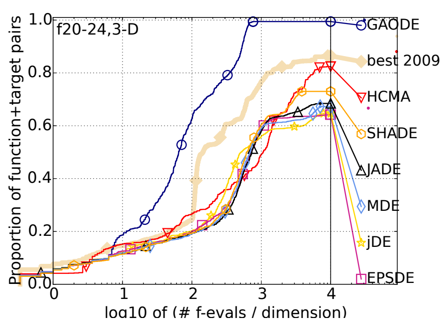
(b) $f_6 \sim f_9$ (moderate fcts)



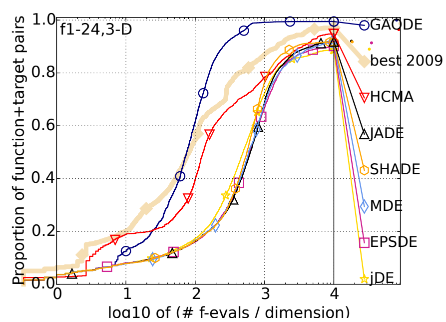
(c) $f_{10} \sim f_{14}$ (ill-conditioned fcts)



(d) $f_{15} \sim f_{19}$ (multi-modal fcts)

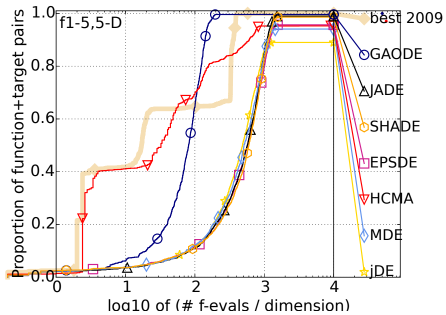


(e) $f_{20} \sim f_{24}$ (weakly structured multi-modal fcts)

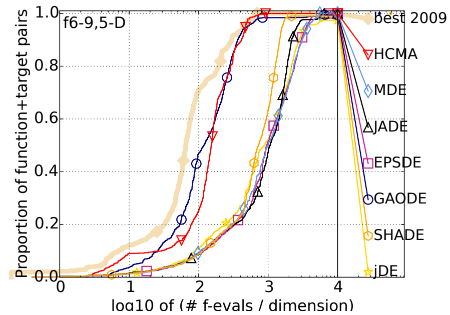


(f) $f_1 \sim f_{24}$ (all functions)

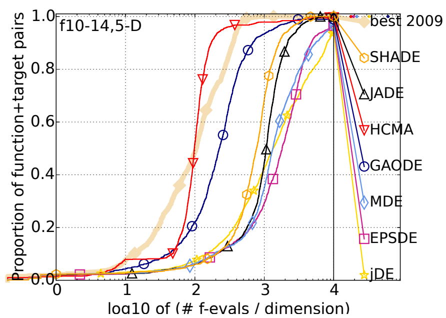
Fig. A.3: Comparisons of GAODE with the adaptive DE variants, HCMA, and best-2009 on BBOB benchmarks ($D = 3$). These figures show bootstrapped Empirical Cumulative Distribution Function (ECDF) of the FEvals divided by dimension for 50 targets in $10^{[-8..2]}$ for each function class (higher than better). For details of the ECDF, see a manual of `COCO software` (http://coco.gforge.inria.fr/).



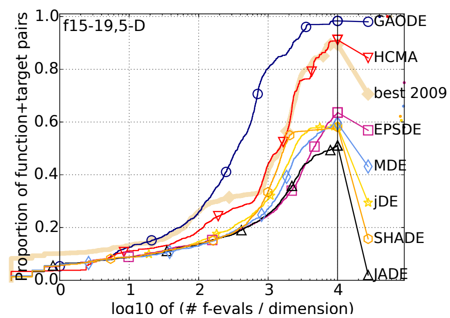
(a) $f_1 \sim f_5$ (separable fcts)



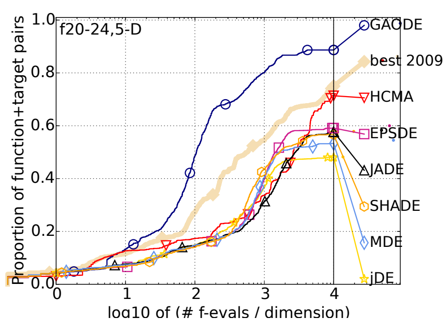
(b) $f_6 \sim f_9$ (moderate fcts)



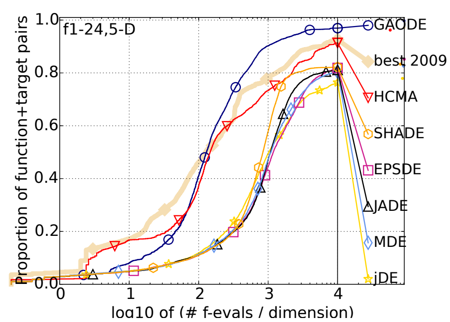
(c) $f_{10} \sim f_{14}$ (ill-conditioned fcts)



(d) $f_{15} \sim f_{19}$ (multi-modal fcts)

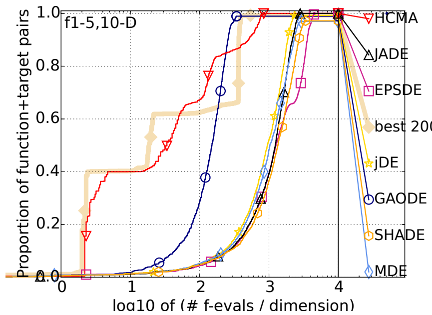


(e) $f_{20} \sim f_{24}$ (weakly structured multi-modal fcts)

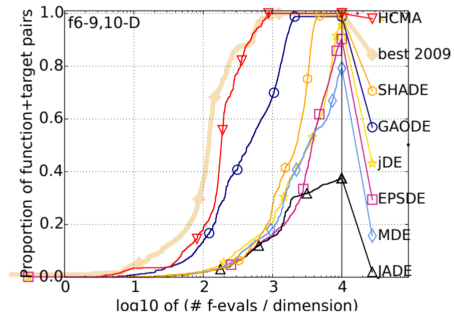


(f) $f_1 \sim f_{24}$ (all functions)

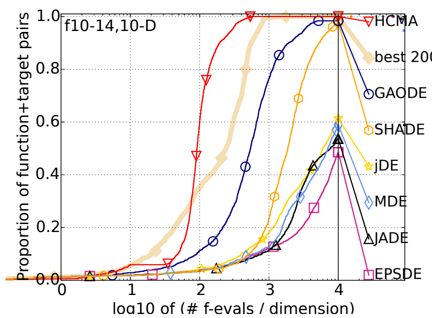
Fig. A.4: Comparisons of GAODE with the adaptive DE variants, HCMA, and best-2009 on BBOB benchmarks ($D = 5$). These figures show bootstrapped Empirical Cumulative Distribution Function (ECDF) of the FEvals divided by dimension for 50 targets in $10^{[-8..2]}$ for each function class (higher than better). For details of the ECDF, see a manual of `COCO software` (http://coco.gforge.inria.fr/).



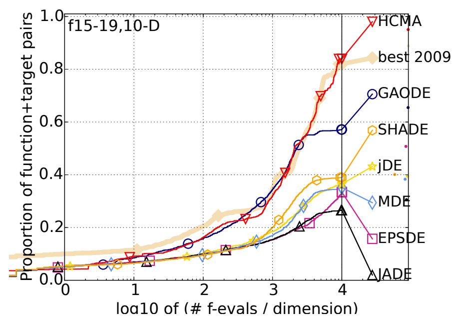
(a) $f_1 \sim f_5$ (separable fcts)



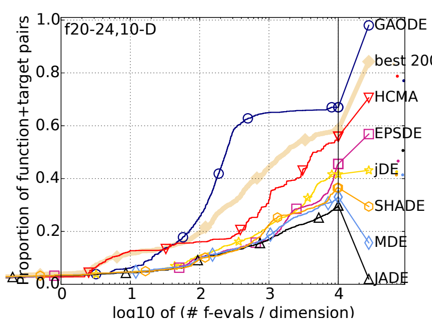
(b) $f_6 \sim f_9$ (moderate fcts)



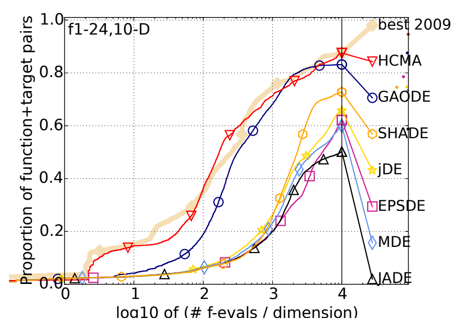
(c) $f_{10} \sim f_{14}$ (ill-conditioned fcts)



(d) $f_{15} \sim f_{19}$ (multi-modal fcts)

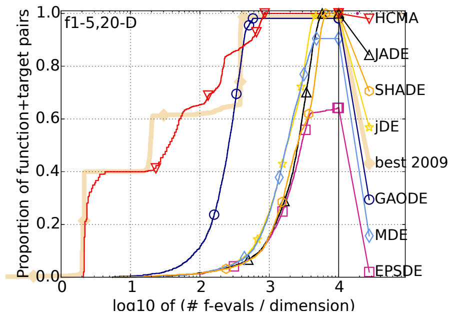


(e) $f_{20} \sim f_{24}$ (weakly structured multi-modal fcts)

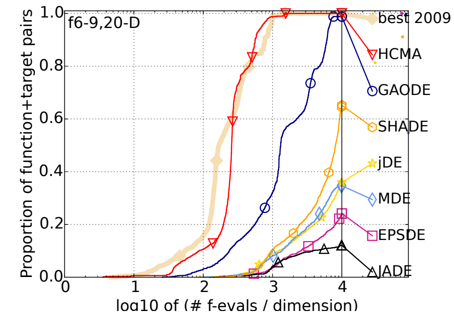


(f) $f_1 \sim f_{24}$ (all functions)

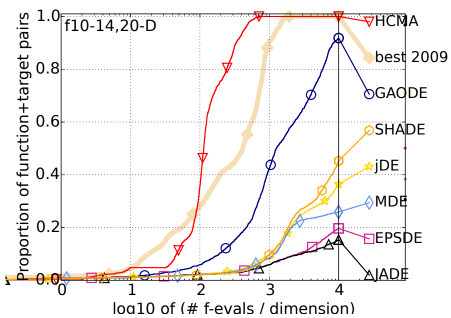
Fig. A.5: Comparisons of GAODE with the adaptive DE variants, HCMA, and best-2009 on BBOB benchmarks ($D = 10$). These figures show bootstrapped Empirical Cumulative Distribution Function (ECDF) of the FEvals divided by dimension for 50 targets in $10^{[-8..2]}$ for each function class (higher than better). For details of the ECDF, see a manual of `COCO software` (http://coco.gforge.inria.fr/).



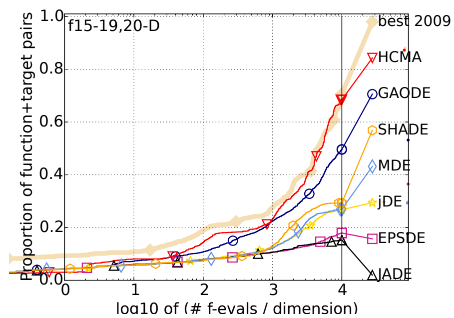
(a) $f_1 \sim f_5$ (separable fcts)



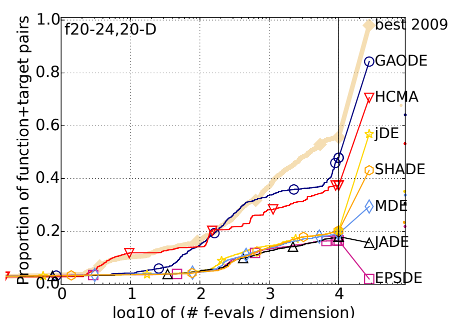
(b) $f_6 \sim f_9$ (moderate fcts)



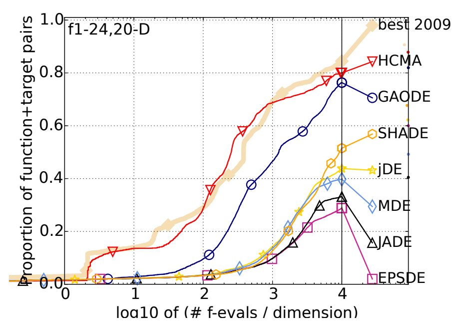
(c) $f_{10} \sim f_{14}$ (ill-conditioned fcts)



(d) $f_{15} \sim f_{19}$ (multi-modal fcts)



(e) $f_{20} \sim f_{24}$ (weakly structured multi-modal fcts)



(f) $f_1 \sim f_{24}$ (all functions)

Fig. A.6: Comparisons of GAODE with the adaptive DE variants, HCMA, and best-2009 on BBOB benchmarks ($D = 20$). These figures show bootstrapped Empirical Cumulative Distribution Function (ECDF) of the FEvals divided by dimension for 50 targets in $10^{[-8..2]}$ for each function class (higher than better). For details of the ECDF, see a manual of `COCO software` (http://coco.gforge.inria.fr/).

References

1. J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE TEVC*, 10(6):646–657, 2006.
2. S. Guo and C. Yang. Enhancing Differential Evolution Utilizing Eigenvector-Based Crossover Operator. *IEEE TEVC*, 19(1):31–49, 2015.
3. S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan. An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization. *IEEE Trans. on SMC. B*, 42(2):482–500, 2012.
4. R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *ASC*, 11:1679–1696, 2011.
5. R. Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.
6. R. Tanabe and A. Fukunaga. Success-History Based Parameter Adaptation for Differential Evolution. In *IEEE CEC*, pages 71–78, 2013.
7. R. Tanabe and A. S. Fukunaga. Improving the search performance of SHADE using linear population size reduction. In *IEEE CEC*, pages 1658–1665, 2014.
8. J. Zhang and A. C. Sanderson. JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE TEVC*, 13(5):945–958, 2009.