Comparison of Parameter Control Mechanisms in Multi-objective Differential Evolution

Martin Drozdik *, Hernan Aguirre, Youhei Akimoto, and Kiyoshi Tanaka

Interdisciplinary Graduate School of Science and Technology, Shinshu University, Nagano, Japan (e-mail: martin@iplab.shinshu-u.ac.jp)

Abstract. Differential evolution (DE) is a powerful and simple algorithm for single- and multi-objective optimization. However, its performance is highly dependent on the right choice of parameters. To mitigate this problem, mechanisms have been developed to automatically control the parameters during the algorithm run. These mechanisms are usually a part of a unified DE algorithm, which makes it difficult to compare them in isolation. In this paper, we go through various deterministic, adaptive, and self-adaptive approaches to parameter control, isolate the underlying mechanisms, and apply them to a single, simple differential evolution algorithm. We observe its performance and behavior on a set of benchmark problems. We find that even the simplest mechanisms can compete with parameter values found by exhaustive grid search. We also notice that *self-adaptive* mechanisms seem to perform better on problems which can be optimized with a very limited set of parameters. Yet, *adaptive* mechanisms seem to behave in a problem-independent way, detrimental to their performance.

Keywords: differential evolution, multi-objective optimization, parameter control, comparative study

1 Introduction

Differential evolution (DE) [14] is a simple to understand, but nevertheless powerful optimizer. However, its performance is highly sensitive to the choice of parameters. Moreover, this dependency changes from problem to problem. Selection of well performing fixed parameters for a particular optimization problem is a relatively little understood subject, especially in the multi-objective realm. This motivated many researchers to develop techniques to set the parameters automatically, during the run of the DE algorithm.

According to the taxonomy in [5], parameter setting techniques are divided into *parameter tuning*, which happens before the run, and *parameter control*, which happens during the run. The former is a subset of a larger field, called *algorithm configuration*, which is itself a deeply researched subject [8]. In this work however, we study only parameter control mechanisms.

* The work of Martin Drozdik has been supported by the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

Parameter control (PC) techniques are further divided into *deterministic, adaptive*, and *self-adaptive*. *Deterministic* techniques apply the parameters according to a given rule, while ignoring any feedback from the search process. *Adaptive* techniques continually update their parameters using feedback from the population. *Self-adaptive* techniques attach different parameters to each individual. These parameters undergo mutation and recombination along with the individuals. Better parameter values lead to individuals with a higher chance to survive and therefore have a higher chance to propagate to the next generation.

Each mentioned paradigm of parameter control is represented by numerous algorithms in the literature. One of the first attempts to control parameters in DE is the (multi-objective) SPDE algorithm [1] belonging to the self-adaptive category. An adaptive mechanism based on population diversity for both single- and multi-objective DE has been proposed by Zaharie in [19]. The use of fuzzy controllers to adapt the parameters has been proposed by Liu et al. [11] The SaDE algorithm [15], originally proposed for single-objective DE, adapts the used DE strategies as well as the parameters. SaDE, which is an adaptive algorithm according to our classification, has been generalized to multi-objective realm and subsequently improved to OW-MOSaDE [6]. A comparison of *single-objective* adaptive and self-adaptive methods is presented in [2] and in [3].

A typical modern multi-objective algorithm is in fact an orchestra of sub-algorithms, each playing its own instrument. There is a sub-algorithm to initialize the population, a sub-algorithm to select individuals for reproduction, a sub-algorithm to maintain diversity, and so on. Various techniques for parameter control are usually published as a part of a unified production-ready algorithm. Apart from the parameter control mechanism, this algorithm usually has its own sub-algorithms to perform tasks *not* related to parameter control. These sub-algorithms usually vary from algorithm to algorithm and make the comparison of algorithms difficult, since it is not clear if the difference in performance should be attributed to the parameter control mechanism itself, or to the difference in sub-algorithms. For example, to estimate diversity of an individual, the OW-MOSaDE algorithm [6] uses the harmonic average distance measure, while the JADE2 algorithm [21] uses the product of distances. In order to isolate these effects, we implement all the parameter control methods within a simple multi-objective DE algorithm DEMO [16].

In this paper, we want to find out if some parameter control paradigm is inherently better in terms of performance and whether the parameter control mechanisms can find favorable parameters in problems which can be successfully optimized only with a limited set of parameters. We are also interested in finding an explanation of the observed performance. We do this by observing the evolution of parameters used by the parameter control methods throughout the optimization process. For this paper, we tried to choose representative examples from each group. We compare one deterministic, three adaptive, and four self-adaptive methods. Some of the methods we present here are originally used only for single-objective optimization, but they can be easily adopted to multi-objective optimization, which we do in this paper.

We conclude, that *self-adaptive* methods are the most robust methods, while performing on a par with the best fixed parameter settings. We found out that adaptive methods may have big problems to find good parameters. Moreover, they seem to adapt their parameters in patterns *independent* of the problem.

However, our conclusions come from empirical results with a single (DEMO) algorithm. It is possible, that applying the studied PC mechanisms to a different algorithm may yield very different results. Such was the case in [13] where the authors studied parameter control in ant colony optimization algorithms.

In the following section we introduce the various mechanisms we use in this work. In Section 3 we explain the details of our experimental setup. In Section 4 we discuss and interpret the empirical results and we conclude in Section 5.

2 Approaches to Parameter Control in DE

In this section we describe DE in more detail and introduce its parameters. Then we introduce the mechanisms that we examine in this paper.

2.1 Differential Evolution Parameters

The fundamental principle of DE is to create new individuals by adding scaled *differences* of individuals to each other. Let $P = \{X_1, ..., X_{\mathrm{NP}}\}$, where $X_i = (x_{i,1}, \ldots, x_{i,n}) \in \mathbb{R}^n$, be the population. In its most basic form, DE traverses through P, attempting to improve each individual X_{target} by generating a *new* individual X_{trial} in the following way: First, three distinct individuals, $X_{r_1}, X_{r_2}, X_{r_3}$ are chosen from P. Then a *scaled* difference of two of these individuals is added to the third one and an intermediate individual X_{mutant} is created:

$$X_{\mathrm{mutant}} := X_{r_1} + \mathrm{F}(X_{r_2} - X_{r_3}). \tag{1}$$

The scaling factor F is the first parameter of DE. Then the X_{trial} is generated by randomly inheriting variables from either X_{mutant} or from X_{target}. One variable with a randomly chosen index *inv* is automatically inherited from X_{mutant} to avoid generating a copy of X_{target}. This is described in (2), where $\mathrm{rand}_{\mathrm{U}}(0,1)$ is a generator of uniformly randomly distributed numbers in $[0;1]$.

$$x_{\mathrm{trial},i} := \begin{cases} x_{\mathrm{mutant},i} & \text{if } \mathrm{rand}_{\mathrm{U}}(0,1) < \mathrm{Cr} \text{ or } i = inv \\ x_{\mathrm{target},i} & \text{else} \end{cases} \tag{2}$$

The number Cr in (2) is the second parameter of DE and it is called the *crossover probability*. Cr controls the proportion of variables that are perturbed in an incumbent individual X_{target} to create a new individual. When $\mathrm{Cr} = 0$, only one variable changes at a time, hence $\mathrm{Cr} = 0$ is well suited for *separable* problems.

Very significant work on understanding the theoretical properties of F and Cr has been done by Zaharie in [18]. An empirical analysis has been performed by Kukkonen in [10]. The population size NP is also considered a parameter of

DE, and there have been attempts to adapt the population size as well [17], but in this paper we restrict ourselves to parameters F and Cr. Moreover, strategies to generate X_{trial}, different than the one in (1) and (2) have been proposed, but in this work we shall consider *only* the default strategy. Next, we present all the parameter control mechanisms that we consider in this study.

2.2 Deterministic Mechanism for Parameter Control

The MDDE algorithm [22] initializes the parameters as relatively big values $\mathrm{F}_0, \mathrm{Cr}_0$, to prevent premature convergence. Then it monotonically decreases them with respect to the generation g, in a geometric sequence, according to:

$$\mathrm{F}_g := \mathrm{F}_0 \exp(-a_0 \frac{g}{g_{\max}})$$
$$\mathrm{Cr}_g := \mathrm{Cr}_0 \exp(-a_1 \frac{g}{g_{\max}}),$$

where $g_{\max}$ is the maximum number of generations.

2.3 Adaptive Mechanisms for Parameter Control

JADE2 The adaptive mechanism in the JADE2 algorithm generates new values of F and Cr for each new X_{trial}. If a particular X_{trial} Pareto dominates the X_{target}, the combination of F and Cr which generated the X_{trial} is recorded as a *successful* one. The values of F are generated from a Cauchy distribution with median μ_{F} and scale $\gamma = 0.1$, the values of Cr from a normal distribution with mean μ_{Cr} and $\sigma = 0.1$. At the end of each generation, the parameters of these distributions are updated using the following rules:

$$\mu_{\mathrm{F}} := (1-c)\mu_{\mathrm{F}} + c.\mathrm{avg}_L(\mathrm{F}_s)$$
$$\mu_{\mathrm{Cr}} := (1-c)\mu_{\mathrm{Cr}} + c.\mathrm{avg}_A(\mathrm{Cr}_s),$$

where $c \in [0; 1]$ is a learning factor, $\mathrm{avg}_L(\mathrm{F}_s)$ is the Lehmer mean of all successful F's and avg_A is the arithmetic mean of successful Cr's in the previous generation. In our experiments we used $c = 0.1$, as recommended by the authors.

OW-MOSaDE Objective-wise MOSaDE [6] attempts to learn which value of Cr is good for a *particular objective*. For each objective $f_i \in (f_1, \ldots, f_m)$ OW-MOSaDE holds one value of $\mu_{i,\mathrm{Cr}}$. These values are updated at the end of each generation if the X_{trial} generated by a particular Cr improves objective f_i. In addition, a master μ_{Cr} is updated if *all* objectives are improved simultaneously. At each generation, one of these $m + 1$ values is randomly chosen to serve as the mean of a normal random distribution with $\sigma = 0.1$, which is sampled to generate the values of Cr. That is, each generation the algorithm concentrates on either one randomly chosen objective or attempts to improve all objectives at once. As opposed to JADE2, there is no learning factor, but the successful values of Cr are retained for lp generations, where $lp = 50$ is a learning period. The value F is not adapted, but generated randomly from a fixed set of normal distributions for each individual.

Control of Diversity Adaptation Algorithm (PDCaDE) Zaharie discovered a simple algebraic relationship between the expected variance of the DE population before and after the generation of new individuals [18]. Based on these results, she developed an algorithm which monitors the variance of the population in the decision space and alters the parameters according to this relationship, so that the variance of the population decreases in a specified, steady manner throughout the entire run. The motivation is to prevent premature convergence and to use the allocated budget of generations evenly.

The algorithm does not have a specific name, so we call it *Population Diversity Control Adaptive DE (PDCaDE)* in the rest of this article. PDCaDE introduces a new parameter γ, which we held constant at $\gamma = 1.25$ for all our experiments. This value was determined by some limited tuning, since the author does not provide a recommendation for multi-objective problems.

2.4 Self-adaptive Mechanisms for Parameter Control

The main idea behind self adaptive mechanisms is that each individual carries the set of parameters *by which it was created*. This way, if an individual is created by a good set of parameters and survives into the next generation, the parameters it carries survive too. Conversely, bad parameter combinations get pruned away.

In all self-adaptive DE mechanisms considered in this paper, the principle is the same. New individuals X_{trial} are generated using (1) and (2), where the F and Cr are not fixed, but replaced by F_{trial} and Cr_{trial}. These values are generated on the spot and then carried by the newly generated X_{trial}. Let us denote by F_i and Cr_i the parameter values carried by individual X_i. Then the methods to generate F_{trial} and Cr_{trial} can be described by simple equations in Table 1.

3 Experimental Design

3.1 Algorithmic Framework

Algorithm 1 shows the unified framework used to compare the selected parameter control mechanisms. The lines that apply *only* to self-adaptive mechanisms are highlighted in *yellow*, while the ones that apply *only* to adaptive mechanisms are highlighted in *purple*.

If we want to draw conclusions about PC mechanisms in general using this methodology, we rely on the following assumption: Let A, B be two PC mechanisms and X, Y be two DE algorithms. If $X(A)$ (algorithm X with mechanism A) is better than $X(B)$ in some regard, then $Y(A)$ is better than $Y(B)$. Surely the validity of this assumption depends on many factors. Some research in this direction has been done in [13]. Since all our experiments are performed within this single algorithm, the most important task for future work is to explore the validity of our assumption.

Some methods have their own parameters, which we held constant at the values recommended by their authors. Some methods also use several strategies to generate new individuals, but in this work we limited ourselves to the default strategy described in Equations (1) and (2).

Table 1: Summary of used *self-adaptive* mechanisms

Name	Main formula	additional parameters
SPDE [1]	$\mathrm{F}_{\mathrm{trial}} := \mathrm{rand}_{\mathrm{N}}(0,1)$ $\mathrm{Cr}_{\mathrm{trial}} := \mathrm{Cr}_{r_1} + \mathrm{rand}_{\mathrm{N}}(0,1)(\mathrm{Cr}_{r_2} - \mathrm{Cr}_{r_3})$	$\mathrm{Cr}_{\mathrm{init}} := \mathrm{rand}_{\mathrm{N}}(\mu,\sigma)$ $\mu = 0.5, \sigma = 0.15$
jDE [3]	$\mathrm{F}_{\mathrm{trial}} := \begin{cases} \mathrm{rand}_{\mathrm{U}}(0.1,1.0) & \text{if } \mathrm{rand}_{\mathrm{U}}(0,1) < \tau_1 \\ \mathrm{F}_{\mathrm{target}} & \text{else} \end{cases}$ $\mathrm{Cr}_{\mathrm{trial}} := \begin{cases} \mathrm{rand}_{\mathrm{U}}(0.1,1.0) & \text{if } \mathrm{rand}_{\mathrm{U}}(0,1) < \tau_2 \\ \mathrm{Cr}_{\mathrm{target}} & \text{else} \end{cases}$	$\tau_1 = 0.1$ $\tau_2 = 0.1$
DEMOwSA [20]	$\mathrm{F}_{\mathrm{trial}} = \frac{\mathrm{F}_i + \mathrm{F}_{r_1} + \mathrm{F}_{r_2} + \mathrm{F}_{r_3}}{4} e^{\tau \mathrm{rand}_{\mathrm{N}}(0,1)}$ $\mathrm{Cr}_{\mathrm{trial}} = \frac{\mathrm{Cr}_i + \mathrm{Cr}_{r_1} + \mathrm{Cr}_{r_2} + \mathrm{Cr}_{r_3}}{4} e^{\tau \mathrm{rand}_{\mathrm{N}}(0,1)}$	$\tau = \frac{1}{\sqrt{2n}}$
SAMDE [12]	$\mathrm{F}_{\mathrm{trial}} = \mathrm{F}_{r_1} + \mathrm{F}'(\mathrm{F}_{r_2} - \mathrm{F}_{r_3})$ $\mathrm{Cr}_{\mathrm{trial}} = \mathrm{Cr}_{r_1} + \mathrm{F}'(\mathrm{Cr}_{r_2} - \mathrm{Cr}_{r_3})$	$\mathrm{F}' := \mathrm{rand}_{\mathrm{U}}(0.7,1.0)$

$\mathrm{rand}_{\mathrm{N}}(\mu,\sigma)$ - generator of normal random numbers
$\mathrm{rand}_{\mathrm{U}}(a,b)$ - generator of uniform random numbers

Algorithm 1: Adaptive and self-adaptive DEMO [16] algorithm

initialize $P = \{X_1, ..., X_{\mathrm{NP}}\}$ uniformly randomly in the decision space
initialize F and Cr generators
initialize values of F_i and Cr_i for $i = 1, \ldots, \mathrm{NP}$
for generation := 1 *to* G_{max} **do** *Evolutionary loop*
  for target := 1 *to* NP **do** *Generational loop*
    generate $\mathrm{F}_{\mathrm{trial}}$ and $\mathrm{Cr}_{\mathrm{trial}}$
    compute $\mathrm{F}_{\mathrm{trial}}$ and $\mathrm{Cr}_{\mathrm{trial}}$ using Table 1
    generate X_{trial} using $\mathrm{F}_{\mathrm{trial}}$ and $\mathrm{Cr}_{\mathrm{trial}}$ from (1) and (2)
    attach $\mathrm{F}_{\mathrm{trial}}$ and $\mathrm{Cr}_{\mathrm{trial}}$ to X_{trial}
    project X_{trial} to decision space
    if X_{target} *dominates* X_{trial} **then**
      discard X_{trial}
    else if X_{trial} *dominates* X_{target} **then**
      replace X_{target} with X_{trial}
    else if X_{target} *and* X_{trial} *are mutually non-dominated* **then**
      add X_{trial} to the end of the population
    end
    update success memories
  end
  update parameter generators
  Trim P to size NP using non-dominated sorting [16] and MNN diversity [9]
end

Table 2: Characteristics of the selected WFG problems

	WFG4	WFG6	WFG7	WFG9
separable	yes	no	yes	no
unimodal	no	yes	yes	no

3.2 Problems

WFG problems To test the mechanisms in various conditions, we chose a subset of the WFG [7] test suite with the same concave Pareto front and all possible combinations of separability and modality characteristics. These problems are summarized in Table 2. We held the number of variables fixed at 10 and performed tests for 2 and 3 objectives.

Quadratic problems As we shall later see, even the non-separable multimodal WFG problems can be successfully optimized using many combinations of fixed parameters. To test the ability of parameter control mechanisms to solve challenging problems we developed a scalable problem, that can be solved by relatively few combinations of F and Cr, called Q. The problem Q consists of m functions: $Q = (q_1, \ldots, q_m)$. Each function is a quadratic form $q_m(X) = (X - c_m)D_m(X - c_m)^T$ where

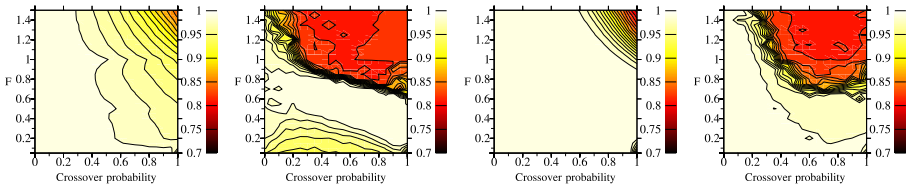
$$\begin{aligned} D_1 &= \mathrm{diag}(1, 2, 4, \ldots, 2^{n-1}), \\ D_2 &= \mathrm{diag}(2^{n-1}, 1, 2, \ldots, 2^{n-2}), \\ &\ldots \end{aligned}$$

and the vectors c_i are generated uniformly randomly in a unit sphere. The resulting problem is then rotated in the decision space around all $n-2$ rotation subspaces by 45 degrees. [1] Moreover, the population for this problem is generated randomly uniformly in a sphere of radius 2^{10} which is shifted from the origin in a random direction by 2^{14}. In this work, we explore the Q problem for 2, 3, and 4 objectives, while the number of variables remains fixed at 10.

3.3 Observed Statistics

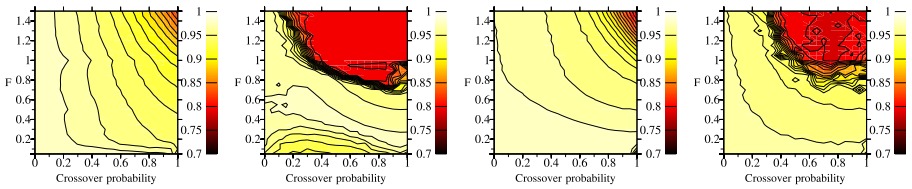
We are interested in the *performance* of the various methods as well as in their *behavior*. To measure the performance, we use the hypervolume [23] metric, since it measures both convergence and diversity of the resulting Pareto front approximation. As a reference point for both types of problems we first construct

[1] Details on this methodology can be found in [4].



(a) WFG4 (S-MM) (b) WFG6 (S-UM) (c) WFG7 (NS-UM) (d) WFG9 (NS-MM)

Fig. 1: Average normalized hypervolume for 2 objectives



(a) WFG4 (S-MM) (b) WFG6 (S-UM) (c) WFG7 (NS-UM) (d) WFG9 (NS-MM)

Fig. 2: Average normalized hypervolume for 3 objectives

the hyperbox which contains the entire true Pareto front and add a unit vector to its upper corner.

In order to simplify the interpretation of the hypervolume, we normalize it by dividing it by the maximal attainable hypervolume in the case of WFG problems, and by the volume of the hyperbox between the origin and the reference point for the Q problem. This way we know that the maximal attainable normalized hypervolume, corresponding to complete convergence is 1.

In order to observe the *behavior* of the mechanisms, we log each combination of F and Cr that the algorithm uses in one generation.

4 Results and Discussion

4.1 Parameter Tuning

For each problem we performed a preliminary tuning of the F and Cr parameters by grid search. We explored the ranges $\mathrm{F} \in [0.05; 1.5]$ and $\mathrm{Cr} \in [0; 1]$ with a resolution of 0.05. For each combination of parameters, we ran 10 independent runs of Algorithm 1 with fixed parameters. The average normalized hypervolume from this tuning is presented in the form of heat-maps, with hot colors meaning good performance. The tuning results for the WFG problems are in figures 1 and 2. In each figure we see a bright, L-shaped region of favorable values. Some theoretical explanation of the shape of this region can be found in [10] and [4].

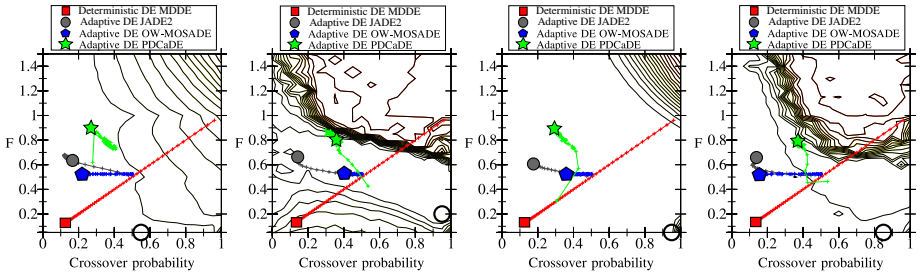
Table 3: Average normalized hypervolume for the WFG problems

		2 objectives			
		WFG4	WFG6	WFG7	WFG9
	start	0.774 (1.2e-02)	0.618 (1.5e-02)	0.706 (8.6e-03)	0.666 (2.2e-02)
	ideal	0.999 (6.9e-05)	0.999 (5.9e-04)	0.999 (3.5e-04)	0.996 (1.4e-03)
	MDDE	0.998 (3.4e-04)	0.820 (8.6e-04)	0.999 (7.5e-06)	0.905 (8.3e-02)
adaptive	JADE2	**0.999** (7.2e-06)	**0.992** (4.4e-03)	0.999 (1.0e-05)	**0.996** (8.7e-04)
adaptive	OW-MOSaDE	0.997 (6.6e-04)	0.975 (6.4e-03)	**0.999** (9.1e-06)	0.993 (2.0e-03)
adaptive	PDCaDE	0.998 (1.7e-03)	0.980 (7.1e-03)	0.999 (4.9e-05)	0.993 (2.2e-03)
self-adaptive	DEMOwSA	0.998 (2.9e-04)	**0.991** (3.8e-03)	0.999 (2.1e-05)	0.989 (3.5e-03)
self-adaptive	jDE	**0.999**(7.5e-06)	0.985 (1.7e-02)	**0.999** (1.4e-05)	**0.996** (1.0e-03)
self-adaptive	SAMDE	0.999 (8.0e-06)	0.980 (1.4e-02)	0.999 (1.5e-05)	0.995 (9.5e-04)
self-adaptive	SPDE	0.999 (1.1e-05)	0.970 (1.1e-02)	0.999 (8.8e-06)	0.996 (4.7e-04)
		3 objectives			
		WFG4	WFG6	WFG7	WFG9
	start	0.669 (1.9e-02)	0.563 (8.9e-03)	0.646 (1.0e-02)	0.575 (2.0e-02)
	ideal	0.987 (2.2e-04)	0.983 (1.6e-03)	0.988 (1.1e-04)	0.976 (2.9e-03)
	MDDE	0.978 (6.7e-04)	0.807 (3.6e-02)	0.986 (1.4e-04)	0.892 (8.4e-02)
adaptive	JADE2	**0.982** (4.3e-04)	**0.977** (3.5e-03)	0.978 (4.5e-04)	**0.965** (1.9e-03)
adaptive	OW-MOSaDE	0.968 (1.3e-03)	0.971 (8.3e-03)	0.977 (5.2e-04)	0.961 (1.6e-03)
adaptive	PDCaDE	0.974 (2.6e-03)	0.966 (8.4e-03)	**0.979** (9.6e-04)	0.962 (2.2e-03)
self-adaptive	DEMOwSA	0.972 (1.0e-03)	**0.979** (1.6e-03)	0.975 (9.3e-04)	0.959 (1.8e-03)
self-adaptive	jDE	0.982 (7.0e-04)	0.967 (1.3e-02)	**0.981** (5.3e-04)	**0.968** (2.7e-03)
self-adaptive	SAMDE	0.983 (5.8e-04)	0.968 (8.8e-03)	0.977 (5.4e-04)	0.963 (1.5e-03)
self-adaptive	SPDE	**0.985** (6.6e-04)	0.964 (1.1e-02)	0.980 (3.8e-04)	0.965 (1.6e-03)

4.2 Parameter Control on WFG problems

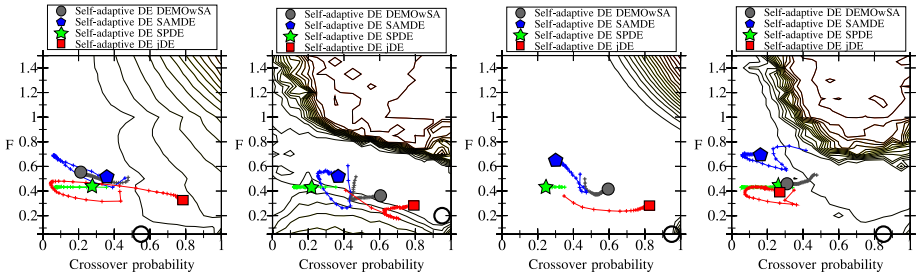
For each of the studied methods we ran 50 independent runs with a fixed population size (NP) of 500 individuals. Each run was limited by 500 generations. The average normal hypervolume along with the standard deviation across the 50 runs is presented in Table 3. The value of normalized hypervolume at the start of the run is denoted as *start*. For each problem, based on the initial tuning, we constructed an *ideal* set of fixed parameters and ran the algorithm for 50 independent runs with these settings. Within the group of adaptive methods and the group of self-adaptive methods we marked the highest value in bold.

We can see that both adaptive and self-adaptive methods are performing almost on a par with the ideal parameter set. The only exception is the determin-



(a) WFG4 (S-MM) (b) WFG6 (S-UM) (c) WFG7 (NS-UM) (d) WFG9 (NS-MM)

Fig. 3: [Cr; F] trajectories of adaptive methods for 2 objectives



(a) WFG4 (S-MM) (b) WFG6 (S-UM) (c) WFG7 (NS-UM) (d) WFG9 (NS-MM)

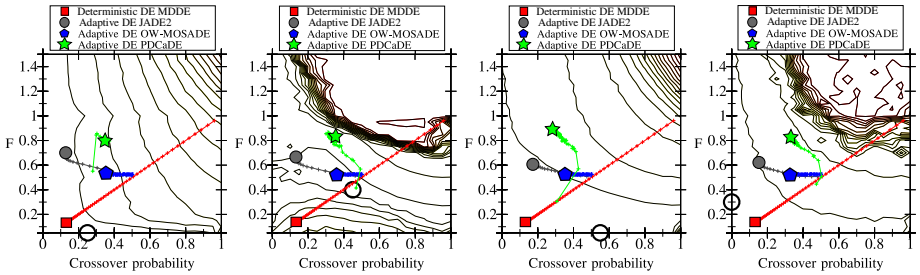
Fig. 4: [Cr; F] trajectories of self-adaptive methods for 2 objectives

istic MDDE algorithm, which shows significant problems for the non-separable WFG6 and WFG9 problems.

For each method, we plot the path of the average used F and Cr with respect to the generation. We call this plot the *trajectory* of that method. The averaged (over the 50 runs) trajectories of adaptive methods along with the MDDE method are plotted in Figure 3 and the trajectories of the self-adaptive methods are in Figure 4. The small crosses are plotted for each 10 generations and the *final* reached value is marked by a large *symbol*. The optimal value of F and Cr is marked by a *black circle*. Moreover, all graphs contain the contour lines of the average normalized hypervolume obtained by parameter tuning.

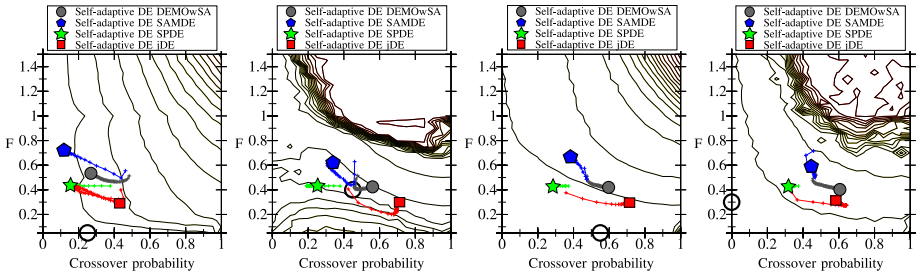
It is immediately clear that all trajectories have different starting points. This is because each PC mechanism has its own way of initialization. Next, we see that each *adaptive* method seems to behave the same way across all observed problems. That is, both JADE2 and OW-MOSaDE aim for the lower values of Cr, while not adapting F much and PDCaDE seems to always converge to the same point, regardless of where the optimal parameter combination is. Conversely, the self-adaptive methods behave differently on each problem.

The situation is very similar for 3 objectives. The trajectories for the adaptive and deterministic methods in Figure 5 seem to behave indifferently to the



(a) WFG4 (S-MM) (b) WFG6 (S-UM) (c) WFG7 (NS-UM) (d) WFG9 (NS-MM)

Fig. 5: [Cr; F] trajectories of adaptive methods for 3 objectives



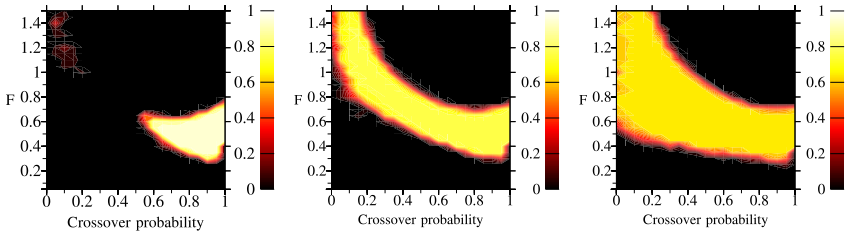
(a) WFG4 (S-MM) (b) WFG6 (S-UM) (c) WFG7 (NS-UM) (d) WFG9 (NS-MM)

Fig. 6: [Cr; F] trajectories of self-adaptive methods for 3 objectives

problem and to the number of objectives. On the other hand, the behavior of self-adaptive methods in Figure 6 depends on the problem. Looking at the results of parameter tuning in Figures 1 and 2 we see a possible explanation. The heat-maps of normalized hypervolume for problems WFG4 and WFG6 have more structure than those of WFG7 and WFG9. The contour lines are more evenly distributed, which may help the algorithms find favorable parameter values. Conversely, the heat-maps for WFG7 and WFG9 have large plateaus associated with favorable parameters, separated by steep cliffs from plateaus with bad parameters. Consequently we see that on WFG4 and WFG6 problems, the trajectories of self-adaptive methods aim correctly for the more favorable regions, while on the WFG7 and WFG9 problems, the behavior seems more random.

4.3 *Q* problems

The performance heat-maps for the Q problems are in Figure 7. The contrast with the data for WFG in Figures 1 and 2 is immediately visible. The area of favorable parameter combinations is relatively small. Moreover, the favorable area is surrounded by steep cliffs. Even a small change in one parameter may mean the difference between a successful convergence and total failure. On such hard problems, the difference in performance of parameter control methods becomes



(a) 2 objectives (b) 3 objectives (c) 4 objectives

Fig. 7: Average normalized hypervolume for the Q problem

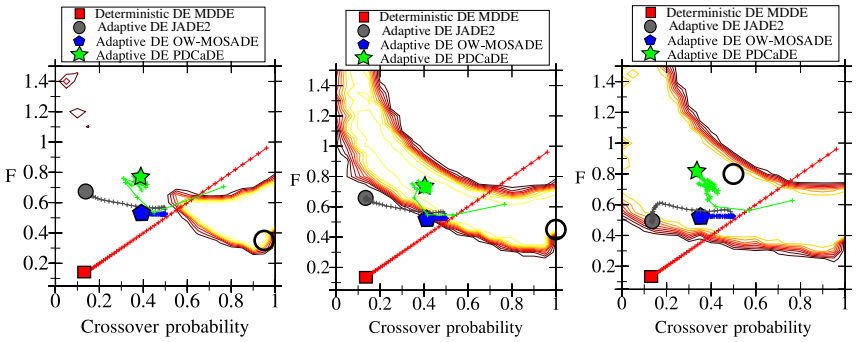
Table 4: Average normalized hypervolume for the Q problem

		2 objectives	3 objectives	4 objectives
	start	0.000 (0.0e+00)	0.000 (0.0e+00)	0.000 (0.0e+00)
	ideal	0.999 (2.1e-05)	0.783 (6.7e-04)	0.673 (2.4e-03)
	MDDE	0.128 (2.8e-01)	0.732 (1.2e-01)	0.653 (5.7e-03)
adaptive	JADE2	0.000 (0.0e+00)	0.175 (2.8e-01)	0.648 (5.8e-03)
adaptive	OW-MOSaDE	0.000 (0.0e+00)	**0.668** (1.1e-01)	0.654 (4.3e-03)
adaptive	PDCaDE	0.000 (0.0e+00)	0.000 (0.0e+00)	**0.659** (8.7e-03)
self-adaptive	DEMOwSA	**0.999** (2.1e-05)	**0.783** (6.8e-04)	**0.652** (5.8e-03)
self-adaptive	jDE	0.745 (4.0e-01)	0.433 (3.7e-01)	0.651 (6.1e-03)
self-adaptive	SAMDE	0.994 (1.5e-02)	0.778 (2.0e-03)	0.638 (7.6e-03)
self-adaptive	SPDE	0.548 (4.6e-01)	0.640 (2.4e-01)	0.643 (6.7e-03)

apparent. The averages and standard deviations of 50 independent runs for 500 generations with a population size of 500 individuals are presented in Table 4.

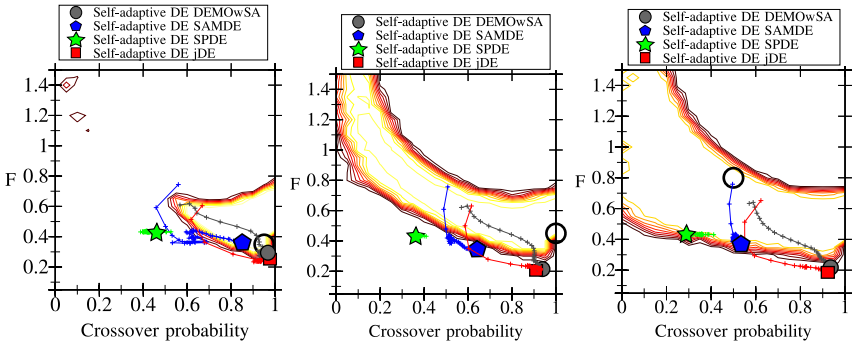
On the 2-objective Q problem *all* the adaptive methods fail completely. Out of 50 runs, not one of them approached close enough to the Pareto front. Some minor success has been achieved by the deterministic MDDE method, but the best performers are the *self-adaptive* methods. On the 3-objective problem, OW-MOSaDE catches up, while the other adaptive methods are lagging. For the 4-objective problem, the performances even out. It may seem counterintuitive that increasing the number of objectives makes the parameter control easier, but looking at Figure 7 we see that the more objectives the Q problem has, the bigger is the set of favorable parameter combinations.

The trajectories of the *adaptive* mechanisms in Figure 8 again seem to be very similar for the 2, 3 and 4 objective Q problems. Disturbingly, they resemble those of the WFG problems. The PDCaDE algorithm seems to always converge to Cr = 0.4 and F = 0.8. The OW-MOSaDE cannot adapt the distribution of



(a) 2 objectives (b) 3 objectives (c) 4 objectives

Fig. 8: [Cr; F] trajectories of adaptive methods for the Q problem.



(a) 2 objectives (b) 3 objectives (c) 4 objectives

Fig. 9: [Cr; F] trajectories of self-adaptive methods for the Q problem.

the F parameter and invariably pushes the value of Cr down. This makes sense for the WFG problems, but it is counterproductive for the 2-objective Q problem. The JADE2 mechanism seems to be lured towards small values of Cr even more. Both JADE2 and OW-MOSaDE try to adapt the parameters by learning which parameters generate individuals which dominate another individual. This suggests that for each problem the parameters with this property are similar and that this property does not guarantee good performance. Of course, a more detailed and rigorous investigation is suggested as future work.

The behavior of self-adaptive mechanisms in Figure 9 is completely different. On the 2-objective problem, all self-adaptive mechanisms achieve at least half of the possible hypervolume. This is even true for the SPDE mechanism, which does *not* find the area of favorable parameter combinations. It seems that since the parameters of SPDE are generated randomly, favorable parameter combinations arise often enough to converge partially. The adaptive algorithms also generate

their parameters randomly, but the centers of the random distributions from which these parameters are generated are shifting in the wrong direction.

5 Conclusion

In this paper we compared various deterministic, adaptive, and self-adaptive mechanisms of parameter control in multi-objective differential evolution. We isolated the mechanisms and applied them to a single multi-objective algorithm. We then tested this algorithm on a set of known benchmark problems as well as one new problem. We measured the performance of these methods as well as their behavior in terms of *which parameters* they found.

We found out that on the usual benchmark problems even the simple mechanisms can lead to results comparable with parameter tuning. On the new problem, which we proposed exactly because it can be optimized only by a small set of parameters, the *self-adaptive* methods were the only ones that managed to find a satisfactory Pareto front for all objective dimensionalities. The deterministic method achieved also some limited success, but it is hard to determine if we can attribute this to luck or to the underlying quality of the method. After examining the progress of the parameters used by the *adaptive* methods we found out that each method evolves its parameters in a more or less problem independent way, which seems undesirable.

For future work the behavior of *adaptive* mechanisms should be first confirmed to exist in other contexts, and if so, to be examined in detail and its cause should be established. It would also be interesting to see if our results hold for more modern DE algorithms.

References

1. H. A. Abbass. The self-adaptive Pareto differential evolution algorithm. In *Evolutionary Computation, 2002. CEC 02. Proceedings of the 2002 Congress on*, volume 1, pages 831–836, May 2002.
2. J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. S. Maučec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing*, 11(7):617–629, feb 2007.
3. J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, Dec 2006.
4. M. Drozdik, K. Tanaka, H. Aguirre, S. Verel, A. Liefooghe, and B. Derbel. An Analysis of Differential Evolution Parameters on Rotated Bi-objective Optimization Functions. In *SEAL 2014 conference*, volume 8886, chapter Lecture Notes in Computer Science, pages 143–154. Springer, Dec 2014.
5. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124–141, 1999.
6. V. L. Huang, S. Z. Zhao, R. Mallipeddi, and P. N. Suganthan. Multi-objective optimization using self-adaptive differential evolution algorithm. In *Evolutionary Computation, 2009. CEC 09. IEEE Congress on*, pages 190–194, May 2009.

7. S. Huband, P. Hingston, L. Barone, and L. While. A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.
8. F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Sttzle. ParamILS: An Automatic Algorithm Configuration Framework. *J. Artif. Int. Res.*, 36(1):267–306, sep 2009.
9. S. Kukkonen and K. Deb. A Fast and Effective Method for Pruning of Non-dominated Solutions in Many-Objective Problems. In *Parallel Problem Solving from Nature - PPSN IX*, volume 4193, chapter Lecture Notes in Computer Science, pages 553–562. Springer Berlin Heidelberg, 2006.
10. S. Kukkonen and J. Lampinen. An Empirical Study of Control Parameters for The Third Version of Generalized Differential Evolution (GDE3). In *IEEE Congress on Evolutionary Computation*, pages 2002–2009, 2006.
11. J. Liu and J. Lampinen. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Computing*, 9(6):448, 2005.
12. R. C. Pedrosa Silva, R. A. Lopes, and F. G. Guimares. Self-adaptive Mutation in the Differential Evolution. In *GECCO*, GECCO 11, pages 1939–1946, New York, NY, USA, 2011. ACM.
13. P. Pellegrini, T. Sttzle, and M. Birattari. A critical analysis of parameter adaptation in ant colony optimization. *Swarm Intelligence*, 6(1):23–48, 2012.
14. K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, New York, NY, 2005.
15. A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. *Evolutionary Computation, IEEE Transactions on*, 13(2):398–417, April 2009.
16. T. Robič and B. Filipič. DEMO: Differential Evolution for Multiobjective Optimization. In *Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410, pages 520–533. Springer Berlin Heidelberg, 2005.
17. J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, 10(8):673–686, 2006.
18. D. Zaharie. Critical Values for the Control Parameters of Differential Evolution Algorithm. In *Proceedings of MENDEL 2002*, 2002.
19. D. Zaharie. Control of Population Diversity and Adaptation in Differential Evolution Algorithms. In R. Matousek and P. Osmera, editors, *Proc. of Mendel 2003, 9th International Conference on Soft Computing*, pages 41–46, Brno, Czech Republic, jun 2003.
20. A. Zamuda, J. Brest, B. Boskovic, and V. Zumer. Differential evolution for multiobjective optimization with self adaptation. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3617–3624, Sept 2007.
21. J. Zhang and A. C. Sanderson. JADE: Adaptive Differential Evolution With Optional External Archive. *Evolutionary Computation, IEEE Transactions on*, 13(5):945–958, Oct 2009.
22. M. Zhang, S. Zhao, and X. Wang. Multi-objective evolutionary algorithm based on adaptive discrete Differential Evolution. In *Evolutionary Computation, 2009. CEC 09. IEEE Congress on*, pages 614–621, May 2009.
23. E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Comput. Eng. Netw. Lab. Swiss Federal Instit. Technol. (ETH), Zurich, Switzerland, 1999.