Applied Soft Computing Journal 74 (2019) 368–384

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Dynamic parameter adaptation in metaheuristics using gradient approximation and line search

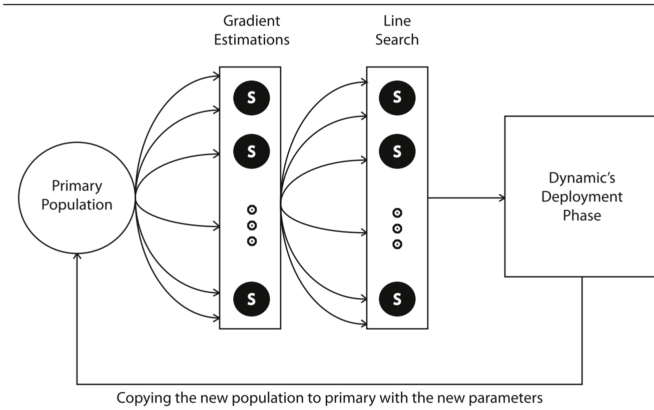
Vasileios A. Tatsis *, Konstantinos E. Parsopoulos

Department of Computer Science and Engineering, University of Ioannina, GR-45110 Ioannina, Greece

HIGHLIGHTS

- Metaheuristics can be effective solvers under proper parameterization.
- We propose GPALS, a general on-line parameter adaptation method for metaheurtistics.
- GPALS is based on approximate gradient search and line search in parameter domain.
- The method is demonstrated on Differential Evolution algorithm for two test suites.
- Significant benefits for the user and the algorithm are gained.

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 12 January 2018
Received in revised form 14 September 2018
Accepted 25 September 2018
Available online 12 October 2018

Keywords:
Metaheuristics
Optimization
Parameter adaptation
Parameter control
Differential evolution

ABSTRACT

Metaheuristics have been established as effective solvers for challenging optimization problems. However, their performance is highly dependent on their parameter settings. For this reason, various parameter tuning techniques have been developed, spanning two major categories: online and offline techniques. Online techniques are based on performance data of the algorithm during its run, while offline techniques are based on preprocessing or historical performance data of the algorithm. Alternatively to these techniques, we propose a general online parameter adaptation method based on estimations of the algorithm's performance and gradient search in the parameter domain. The proposed method is demonstrated on Differential Evolution, a state-of-the-art metaheuristic for continuous optimization. Our experimental validation includes problems of low and high dimension as well as comparisons with distinguished adaptive algorithms. The obtained results suggest that the proposed approach is beneficial, relieving the user from the burden of proper parameterization.

1. Introduction

The increasing complexity of real-world problems has created ongoing demand for efficient optimization algorithms. Metaheuristics are placed among the most popular approaches for solving challenging optimization problems in diverse scientific fields [1]. Despite the lack of guaranteed solution optimality, metaheuristics can provide satisfactory near-optimal solutions in reasonable time. However, their performance is heavily dependent on their proper parameterization, which is far from being an easy task.

There are two major categories of parameter setting techniques that dominate the relevant literature [2–4]:

(a) *Offline* parameter tuning prior to the application of the algorithm.
(b) *Online* parameter adaptation (control) during the algorithm's run.

* Corresponding author.
E-mail addresses: vtatsis@cse.uoi.gr (V.A. Tatsis), kostasp@cse.uoi.gr (K.E. Parsopoulos).

https://doi.org/10.1016/j.asoc.2018.09.034
1568-4946/

Table 1
Statistical comparisons between the three GPALS-DE parameterizations and the base algorithms for the high-dimensional test suite. The symbols "+", "−", and "=", denote wins, losses, and ties of GPALS-DE against the competitor algorithms.

Dim.	Algorithm	GPALS-DE$_{0.5}$			GPALS-DE$_{0.2}$			GPALS-DE$_{0.8}$		
		+	−	=	+	−	=	+	−	=
50	DE$_{bin}$	15	3	1	14	3	2	13	3	3
	DE$_{exp}$	10	5	4	11	6	2	4	8	7
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	16	3	0	16	3	0	16	3	0
100	DE$_{bin}$	17	1	1	17	1	1	17	1	1
	DE$_{exp}$	13	4	2	12	4	3	10	6	3
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	16	3	0	16	3	0	16	3	0
200	DE$_{bin}$	17	1	1	17	1	1	17	1	1
	DE$_{exp}$	13	4	2	14	4	1	13	6	0
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	16	3	0	17	2	0	16	3	0
500	DE$_{bin}$	19	0	0	19	0	0	19	0	0
	DE$_{exp}$	13	5	1	11	4	4	13	6	0
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a

Table 2
Number of wins, losses, and ties of the GPALS-DE$_{0.5}$ algorithm against GaDE, SHADE, and L-SHADE for the high-dimensional test problems.

	Dimension											
	50			100			200			500		
GPALS-DE$_{0.5}$ vs.	+	−	=	+	−	=	+	−	=	+	−	=
GaDE	7	3	9	8	3	8	8	4	7	7	4	8
SHADE$_{60}$	7	1	11	13	1	5	15	1	3	17	1	1
SHADE$_{100}$	5	3	11	13	1	5	13	1	5	16	1	2
mSHADE$_{60}$	6	2	11	6	1	12	8	1	10	16	1	2
mSHADE$_{100}$	13	1	5	13	1	5	14	1	4	14	1	4
L-SHADE	9	2	8	13	2	4	14	2	3	n/a	n/a	n/a
mL-SHADE	18	1	0	18	0	1	19	0	0	n/a	n/a	n/a

Table 3
Statistical comparisons between the GPALS-DE$_{0.5}$ algorithms with different population size and the base algorithms for the high-dimensional test suite.

Dim.	Algorithm	GPALS-DE$^{40}_{0.5}$			GPALS-DE$^{60}_{0.5}$			GPALS-DE$^{80}_{0.5}$		
		+	−	=	+	−	=	+	−	=
50	DE$_{bin}$	14	4	1	15	3	1	11	5	3
	DE$_{exp}$	11	5	3	10	5	4	3	15	1
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	16	3	0	16	3	0	16	3	0
100	DE$_{bin}$	17	1	1	17	1	1	16	1	1
	DE$_{exp}$	13	5	1	13	4	2	6	8	5
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	16	3	0	16	3	0	16	3	0
200	DE$_{bin}$	17	1	1	17	1	1	17	1	1
	DE$_{exp}$	12	4	3	13	4	2	6	9	4
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	16	3	0	16	3	0	16	3	0
500	DE$_{bin}$	19	0	0	19	0	0	19	0	0
	DE$_{exp}$	11	4	4	13	5	1	11	6	2
	CHC	19	0	0	19	0	0	19	0	0
	GCMAES	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a

Table 4
Number of wins, losses, and ties of GPALS-DE against SHADE for different population sizes in the high-dimensional test problems.

	Dimension											
	50			100			200			500		
GPALS-DE$^{40}_{0.5}$ vs.	+	−	=	+	−	=	+	−	=	+	−	=
SHADE$_{40}$	13	2	4	16	1	2	18	1	0	18	1	0
mSHADE$_{40}$	3	8	8	5	5	9	13	5	1	18	1	0
GPALS-DE$^{80}_{0.5}$ vs.	+	−	=	+	−	=	+	−	=	+	−	=
SHADE$_{80}$	5	13	1	12	6	1	13	5	1	18	1	0
mSHADE$_{80}$	15	3	1	14	4	1	14	4	1	18	1	0

Table 5
Statistical comparisons of running times between the serial version of GPALS-DE$_{0.5}$ and SHADE for the high-dimensional test problems.

Dim.	GPALS-DE vs.	+	−	=
50	SHADE	16	2	1
100	SHADE	17	0	2
200	SHADE	19	0	0
500	SHADE	18	1	0

Table 6
Statistical comparisons between GPALS-DE, SHADE, and L-SHADE on the CEC 2013 test suite.

	Dimension					
	30			50		
	+	−	=	+	−	=
mGPALS-DE vs SHADE	3	21	4	3	19	6
GPALS-DE vs mSHADE	8	9	11	12	9	7
mGPALS-DE vs L-SHADE	1	23	4	1	22	5
GPALS-DE vs mL-SHADE	5	12	11	5	18	5

Offline parameter tuning is based on existing performance data of the algorithm from previous applications on similar problems. If such data is unavailable, it is collected through a preprocessing phase based on preliminary experimentation with different parameter settings on the problem at hand. The best-performing parameter values are then adopted in the algorithm. Typically, this approach requires significant computational effort. Our experience with numerous metaheuristics suggests that this effort can be comparable to the effort needed for solving the problem itself. Different parameter tuning approaches are based on statistical methodologies. Typical examples are the Design of Experiments [5], F-Race [6], Sequential Model-Based Optimization [7], and ParamILS [8]. Such approaches can offer promising results at the cost of additional implementation and experimentation effort. Nevertheless, their outcome is often reusable in a wide range of similar problems.

In contrast to the offline tuning approaches, online methods aim at dynamically adapting the parameters of the algorithm, based on feedback during its run [2,8]. The popularity of such methods can be attributed to the lack of preliminary experimentation and the limited user intervention. On the other hand, online methods often exhibit two major weaknesses:

1. *Overspecialization*: they are usually based on *ad hoc* procedures especially developed for a specific algorithm (or problem type).
2. *New parameters*: the number of new parameters introduced by the tuning method may significantly expand the parameter domain.

The overspecialization issue renders the outcome of online methods hardly reusable even in different runs of the algorithm on a specific problem. Moreover, increasing significantly the number of parameters by introducing new ones may increase the sensitivity of the algorithm and, concurrently, impose the necessity for tuning the tuning procedure itself. These weaknesses have offered motivation for the development of new online methods in the past decade. Brief presentation of algorithms equipped with online parameter tuning procedures is postponed until Section 2.2.

Recently, a grid-based parameter adaptation method was proposed and validated on two population-based algorithms, namely Differential Evolution and Particle Swarm Optimization [9,10]. The motivation for its development stemmed from the aforementioned weaknesses of the online parameter tuning methods. Thus, the

Table 7
Statistical comparisons between GPALS-DE and other adaptive DE-based algorithms on the CEC 2013 test suite.

	Dimension					
	30			50		
GPALS-DE vs.	+	−	=	+	−	=
DEcfbLS	10	12	6	11	11	6
jande	12	7	9	14	12	2
DE_APC	12	7	9	11	10	7
PVADE	14	8	6	18	4	6

Table 8
Statistical comparisons of the serial execution times between GPALS-DE and the SHADE algorithm for the CEC 2013 test suite.

Dim.	GPALS-DE vs.	+	−	=
30	SHADE	20	7	1
50	SHADE	20	7	1

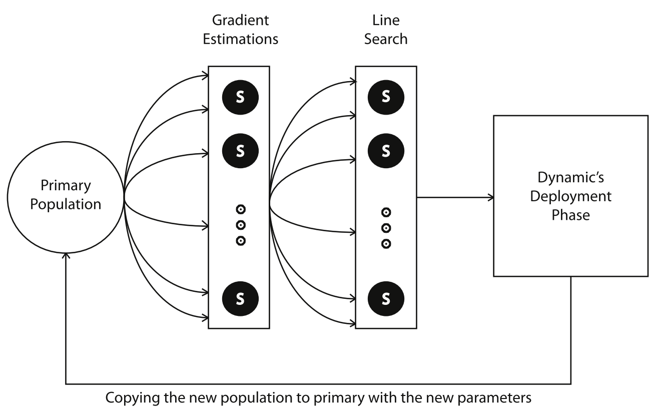


Fig. 1. Graphical illustration of the GPALS method.

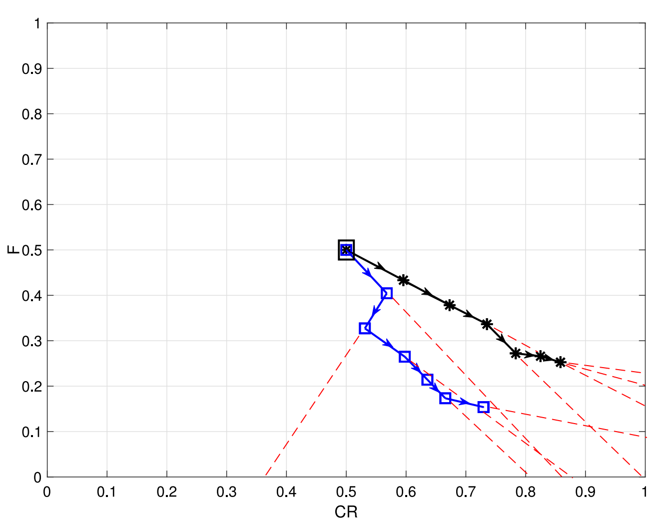


Fig. 2. Sample trajectories of the DE scalar parameters for two test functions.

main goal was the development of a general online parameter tuner with least possible number of new parameters. That method (denoted as DEGPA) performs online search in a discretized parameter domain (grid). The search is based on short-run estimations of the algorithm's performance for neighboring parameter values, starting from an arbitrary setting. The selected parameter values are then used for some iterations before another tuning cycle takes place. Thus, the parameters are periodically adapted during the run

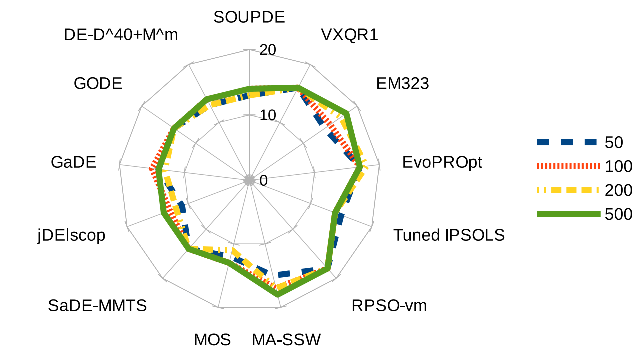


Fig. 3. Number of test problems where GPALS-DE$_{0.5}$ achieved equal or better average error than the competitor algorithms.

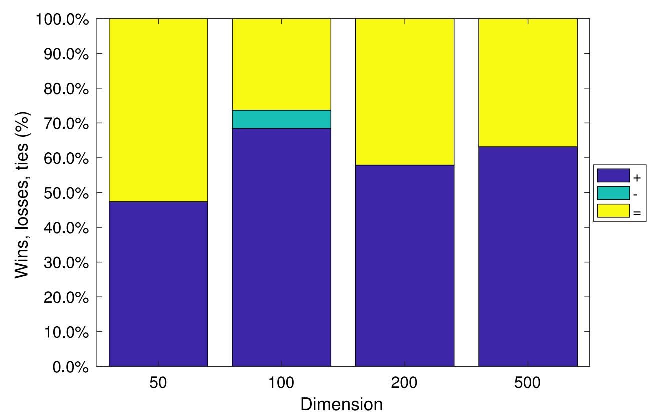


Fig. 4. Wins, losses, and ties of GPALS-DE$_{0.5}$ against the DEGPA approach.

according to the recorded performance of the algorithm. The grid-based parameter adaptation method achieved promising results in tuning mutation operators and scalar parameters of the tested evolutionary algorithms on high-dimensional test problems [9–12].

The present work introduces a new general online parameter tuning method that refines and improves the grid-based approach. Its core mechanism lies in the replacement of the grid search with an approximate gradient search with line search in the parameter domain. This approach offers two significant advantages against the grid. First, the tuner is allowed to perform informative large steps in the parameter domain based on the line search procedure. Second, the scalar parameters are not confined in a discretized subset of their domain, but they can assume continuous values. Moreover, this approach has inherent parallelization properties, taking full advantage of modern computer systems. For consistency reasons, the proposed method is demonstrated on Differential Evolution [13] on two established test suites that consist of low-dimensional and high-dimensional test problems.

The rest of the paper is organized as follows: Section 2 offers the necessary background information. The proposed approach is analyzed in Section 3. In Section 4, the experimental results are reported and discussed. The paper concludes in Section 5.

2. Background information

For completeness reasons, the following paragraphs are devoted to brief presentation of the employed Differential Evolution algorithm and related parameter adaptation methods. Henceforth, we consider the general form of the n-dimensional bound-constrained continuous optimization problem

$$\min_{x \in X \subset \mathbb{R}^n} f(x), \tag{1}$$

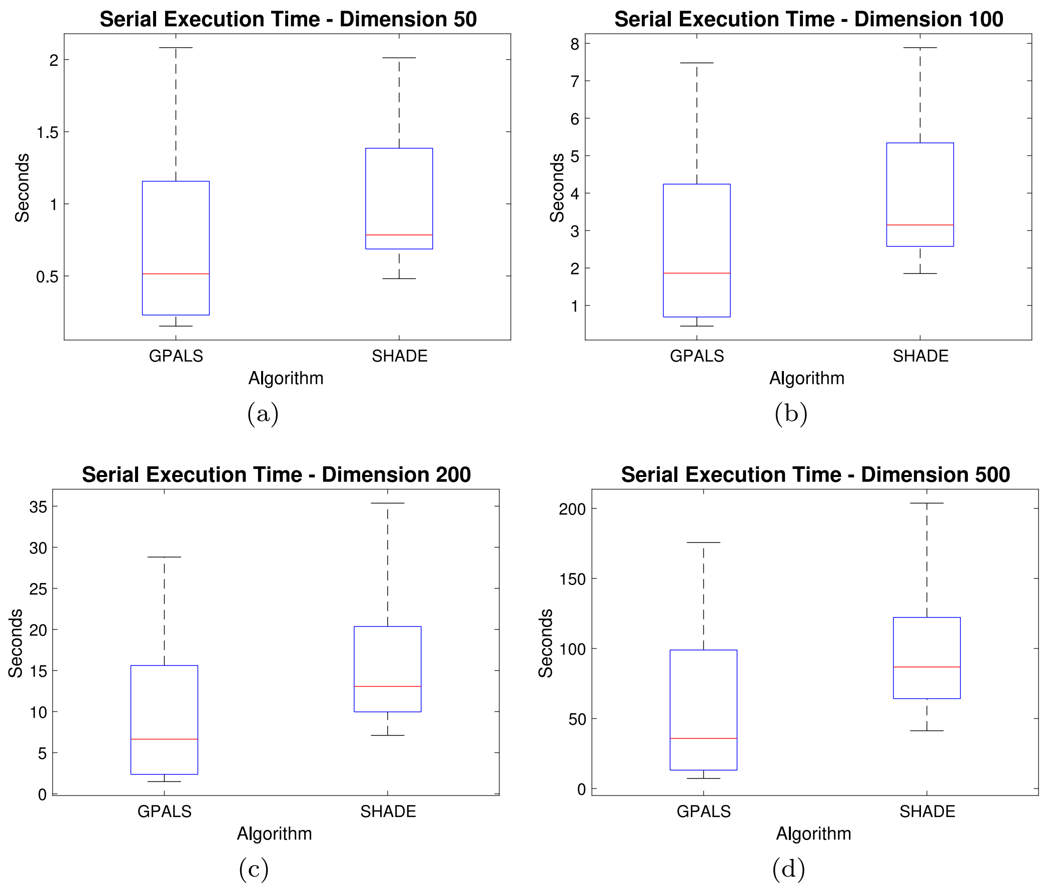


Fig. 5. Serial running time for the proposed GPALS-DE$_{0.5}$ approach and the SHADE algorithm in the high-dimensional test suite for dimension (a) 50, (b) 100, (c) 200, and (d) 500.

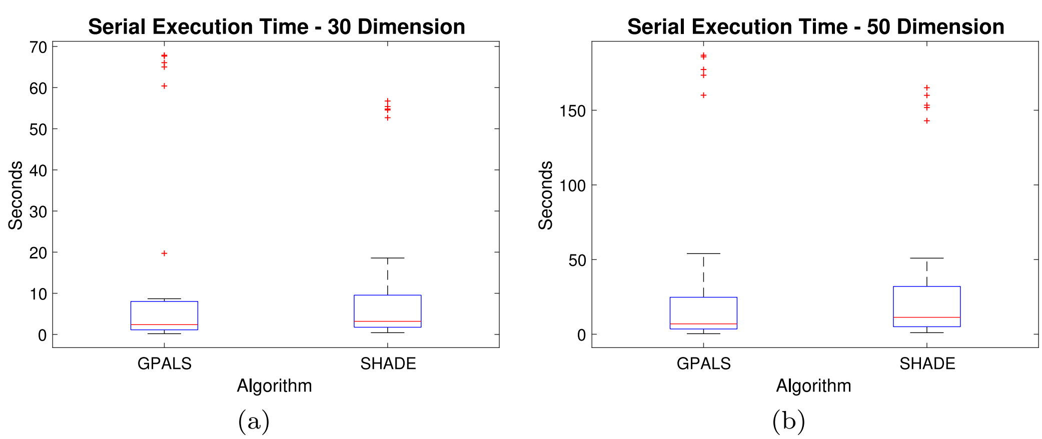


Fig. 6. Serial execution time for the proposed GPALS approach and the baseline SHADE algorithm in the CEC 13 test suite, for dimension (a) 30, and (b) 50.

where the search space X is defined as a hypercube,

$$X = [l_1, u_1] \times \cdots \times [l_n, u_n],$$

with l_i and u_i denoting the lower bound and the upper bound of the ith direction component, respectively. We also consider the sets of indices

$$D \triangleq \{1, 2, \ldots, n\}, \qquad I \triangleq \{1, 2, \ldots, N\}, \tag{2}$$

where D refers to direction components and I refers to population members, respectively. The objective function value of a vector $x_i \in X$, $i \in I$, is simply denoted as

$$f_i = f\left(x_i\right).$$

Finally, the function `rand()` denotes a pseudo-random number generator that produces uniformly distributed real numbers in the range [0, 1].

2.1. Differential evolution

Differential Evolution (DE) [14] is a popular metaheuristic for solving numerical optimization problems. It is a stochastic population-based optimization algorithm with flexible operators but sensitive dynamics with respect to its control parameters [15]. Nevertheless, its adaptability, simplicity, and efficiency has placed it among the most popular metaheuristics, counting a significant number of relevant works [16].

Table 9
Average errors and standard deviations of the three GPALS-DE parameterizations and the base algorithms for dimension $n = 50$.

Problem	GPALS-DE$_{0.5}$		GPALS-DE$_{0.2}$		GPALS-DE$_{0.8}$		DE$_{bin}$		DE$_{exp}$		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
f_1	6.59e−14	2.13e−14	1.34e−13	3.23e−14	6.59e−14	2.13e−14	3.00e−17	7.69e−18	2.78e−17	6.29e−33	2.90e+02	5.69e+02	2.78e−17	6.29e−33
f_2	2.35e+00	2.49e+00	6.79e+00	2.78e+00	3.44e+00	2.55e+00	3.87e+01	8.90e+00	3.31e−01	5.90e−02	7.72e+01	1.23e+01	7.69e−11	4.83e−11
f_3	4.54e+01	1.18e+01	1.80e+01	2.49e+01	4.34e+01	1.89e+01	6.99e+01	3.58e+01	3.10e+01	8.65e+00	5.64e+07	1.42e+08	6.38e−01	1.49e+00
f_4	9.78e−14	3.08e−14	2.39e−01	5.94e−01	1.98e−13	3.07e−13	3.21e+01	1.38e+01	4.79e−02	2.01e−01	1.12e+02	2.74e+01	3.72e+02	8.68e+01
f_5	2.96e−14	5.68e−15	5.12e−14	2.59e−14	2.96e−14	5.68e−15	9.86e−04	2.76e−03	0.00e+00	0.00e+00	9.02e−01	1.82e+00	2.16e−01	5.64e−01
f_6	1.32e−13	4.70e−14	4.38e−09	2.19e−08	7.37e−13	1.76e−12	7.16e−14	1.86e−14	1.39e−13	9.43e−15	3.23e+00	2.44e+00	1.90e+01	1.02e+00
f_7	0.00e+00	0.00e+00	2.06e−15	9.49e−15	6.29e−14	2.76e−13	2.22e−15	1.17e−15	8.88e−17	1.96e−16	1.23e−09	1.45e−09	2.10e+01	1.38e+01
f_8	3.06e+02	6.25e+02	4.79e+02	4.43e+02	7.78e+01	1.10e+02	9.02e+10	0.00e+00	9.02e+10	0.00e+00	9.02e+10	9.02e+06	9.03e+10	9.39e+07
f_9	1.94e−07	6.58e−07	9.12e−07	2.48e−06	5.83e−06	9.41e−06	2.85e+02	5.30e+00	2.73e+02	7.40e−01	3.11e+02	4.98e+00	3.16e+02	7.03e+00
f_{10}	1.01e−31	4.73e−31	2.01e−30	1.01e−29	9.83e−27	3.82e−26	1.53e+00	1.29e+00	6.50e−29	3.60e−29	7.72e+00	2.93e+00	9.25e+00	2.82e+00
f_{11}	1.10e−06	5.01e−06	7.38e−07	1.74e−06	1.51e−05	2.78e−05	9.65e−01	2.02e+00	6.26e−05	1.30e−05	1.01e−02	1.26e−02	1.95e+02	3.65e+01
f_{12}	8.23e−10	3.95e−09	1.30e−10	4.05e−10	2.27e−07	9.21e−07	5.82e+00	1.03e+01	5.26e−13	1.64e−13	8.23e+01	1.53e+02	1.14e+02	1.01e+01
f_{13}	3.32e+01	1.65e+01	4.46e+00	1.39e+01	4.04e+01	2.03e+01	5.97e+01	2.22e+01	2.48e+01	1.31e+00	1.43e+07	3.29e+07	1.16e+02	1.43e+01
f_{14}	7.19e−09	2.01e−08	1.76e−01	3.74e−01	2.32e−05	1.08e−04	3.35e+01	1.86e+01	3.55e−08	2.26e−08	6.76e+01	1.30e+01	2.71e+02	7.30e+01
f_{15}	1.17e−28	5.75e−28	6.37e−15	2.55e−14	7.47e−14	2.95e−13	2.29e−01	6.07e−01	1.99e−24	3.22e−24	3.07e+00	5.32e+00	3.94e+01	1.25e+02
f_{16}	1.80e−10	6.47e−10	1.54e−08	4.67e−08	4.12e−06	1.71e−05	5.64e+00	8.47e+00	1.56e−09	2.81e−10	5.60e+01	5.16e+01	2.23e+02	1.50e+01
f_{17}	1.83e+00	4.64e+00	9.21e+00	1.33e+01	8.36e+00	4.81e+00	1.51e+01	1.43e+01	8.52e−01	4.92e−01	7.61e+06	2.44e+07	3.47e+02	2.18e+01
f_{18}	9.71e−08	1.97e−07	1.20e−01	3.30e−01	1.15e−03	2.81e−03	5.73e+00	5.26e+00	1.28e−04	4.63e−05	6.76e+01	3.46e+01	3.59e+02	8.45e+01
f_{19}	1.07e−29	3.64e−29	1.38e−26	6.87e−26	5.09e−18	1.51e−17	1.23e+00	9.26e−01	2.00e−24	1.50e−24	1.95e+02	5.01e+02	1.71e+03	5.84e+03

Table 10
Average errors and standard deviations of the three GPALS-DE parameterizations and the base algorithms for dimension $n = 100$.

Problem	GPALS-DE$_{0.5}$		GPALS-DE$_{0.2}$		GPALS-DE$_{0.8}$		DE$_{bin}$		DE$_{exp}$		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
f_1	1.55e−13	3.86e−14	2.96e−13	6.14e−14	1.21e−13	2.99e−14	1.12e−16	4.28e−17	7.77e−17	1.13e−17	4.67e+02	7.02e+02	5.55e−17	1.26e−32
f_2	1.32e+01	7.03e+00	1.95e+01	3.63e+00	1.84e+01	8.97e+00	7.74e+01	7.77e+00	4.60e+00	4.24e−01	9.96e+01	1.16e+01	2.61e−03	1.30e−02
f_3	1.06e+02	2.51e+01	3.24e+01	3.93e+01	9.18e+01	1.91e+01	4.43e+02	3.63e+02	8.01e+01	1.03e+01	1.52e+08	2.69e+08	1.23e+01	1.80e+01
f_4	2.46e−13	5.37e−14	2.53e−01	6.61e−01	3.16e−13	1.20e−13	1.01e+02	2.25e+01	9.53e−03	4.76e−02	2.92e+02	5.16e+01	8.38e+02	1.39e+02
f_5	8.30e−14	2.71e−14	1.48e−13	4.42e−14	6.25e−14	2.17e−14	2.93e−02	5.32e−02	2.55e−17	5.19e−18	5.95e+00	1.29e+01	2.68e+00	1.05e+01
f_6	2.39e−13	8.99e−14	1.10e−10	5.43e−10	5.07e−13	1.11e−12	1.55e+00	3.88e−01	3.10e−13	1.62e−14	4.79e+00	1.87e+00	1.86e+01	2.45e+00
f_7	0.00e+00	0.00e+00	5.92e−14	2.86e−13	1.34e−15	6.72e−15	1.39e−14	7.12e−15	3.80e−17	5.29e−17	8.67e−02	3.70e−01	6.35e+01	2.36e+01
f_8	2.33e+03	2.44e+03	3.99e+03	2.92e+03	3.94e+03	3.51e+03	1.79e+11	0.00e+00	1.79e+11	0.00e+00	1.79e+11	1.92e+07	1.80e+11	3.54e+08
f_9	9.76e−09	4.88e−08	4.31e−04	2.15e−03	1.67e−05	7.42e−05	5.43e+02	1.36e+01	5.06e+02	9.16e−01	5.87e+02	1.01e+01	6.08e+02	1.07e+01
f_{10}	1.60e−27	8.00e−27	2.36e−21	1.12e−20	2.19e−30	1.09e−29	1.54e+01	3.31e+00	1.35e−28	3.86e−29	2.89e+01	1.01e+01	1.93e+01	5.10e+00
f_{11}	2.05e−08	1.02e−07	1.60e−06	3.14e−06	2.84e−05	1.07e−04	4.31e+01	2.09e+01	1.25e−04	1.43e−05	2.80e+01	3.02e+01	4.82e+02	4.27e+01
f_{12}	6.31e−11	3.03e−10	1.52e−08	7.21e−08	6.74e−09	2.01e−08	7.21e+01	3.21e+01	6.44e−11	1.52e−11	8.72e+02	2.55e+03	2.41e+02	1.23e+01
f_{13}	6.54e+01	1.73e+01	7.91e+00	1.27e+01	7.35e+01	1.58e+01	2.76e+02	6.18e+01	6.13e+01	1.00e+00	9.37e+07	4.02e+08	2.59e+02	2.16e+01
f_{14}	5.15e−09	2.47e−08	1.59e−01	3.72e−01	1.14e−05	3.21e−05	9.37e+01	1.56e+01	4.48e−02	2.24e−01	2.25e+02	4.59e+01	6.19e+02	9.25e+01
f_{15}	7.26e−23	3.63e−22	2.98e−15	1.16e−14	3.11e−14	1.43e−13	3.67e+00	1.76e+00	7.10e−23	7.00e−23	5.99e+00	1.19e+01	5.57e+01	5.22e+01
f_{16}	4.48e−12	9.43e−12	4.60e−07	2.26e−06	6.60e−08	2.06e−07	1.10e+02	3.80e+01	1.94e−02	9.70e−02	2.08e+02	1.49e+02	4.84e+02	2.08e+01
f_{17}	1.52e+01	1.94e+01	1.28e+01	1.45e+01	3.82e+01	3.40e+01	1.78e+02	5.49e+01	1.19e+01	2.62e+00	4.36e+07	7.09e+07	7.04e+02	3.92e+01
f_{18}	9.68e−09	1.96e−08	4.28e−04	1.43e−03	1.24e−04	2.54e−04	1.04e+02	4.39e+01	2.92e−04	6.77e−05	2.37e+02	7.02e+01	1.09e+03	4.15e+02
f_{19}	1.40e−30	6.96e−30	8.44e−20	4.12e−19	6.86e−24	2.54e−23	1.17e+01	2.61e+00	4.79e−23	2.65e−23	4.70e+02	1.84e+03	5.83e+03	9.85e+03

Table 11
Average errors and standard deviations of the three GPALS-DE parameterizations and the base algorithms for dimension $n = 200$.

Problem	GPALS-DE$_{0.5}$		GPALS-DE$_{0.2}$		GPALS-DE$_{0.8}$		DE$_{bin}$		DE$_{exp}$		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
f_1	3.87e−13	1.10e−13	6.00e−13	1.98e−13	3.18e−13	8.84e−14	6.39e−16	5.32e−16	1.78e−16	1.60e−17	9.61e+02	1.65e+03	1.17e−16	1.13e−17
f_2	3.37e+01	8.45e+00	4.40e+01	6.26e+00	4.42e+01	1.19e+01	1.01e+02	5.90e+00	1.89e+01	1.05e+00	1.17e+02	7.60e+00	7.47e−02	2.44e−01
f_3	2.11e+02	3.13e+01	6.72e+01	7.09e+01	1.93e+02	2.66e+01	6.38e+02	3.80e+02	1.79e+02	8.89e+00	2.54e+08	3.97e+08	1.24e+02	8.85e+01
f_4	5.66e−13	1.76e−13	7.74e−01	8.91e−01	6.82e−13	1.93e−13	4.21e+02	5.63e+01	8.52e−02	3.98e−01	6.32e+02	8.43e+01	1.57e+03	1.54e+02
f_5	1.97e−13	4.99e−14	3.63e−13	1.23e−13	1.75e−13	6.75e−14	3.00e−01	7.73e−01	7.49e−17	6.94e−18	1.02e+01	1.59e+01	1.13e+00	2.84e+00
f_6	5.18e−13	9.65e−14	2.02e−03	1.01e−02	1.73e−12	6.46e−12	5.28e+00	9.65e−01	6.46e−13	2.53e−14	8.14e+00	2.26e+00	1.93e+01	7.39e−01
f_7	0.00e+00	0.00e+00	3.09e−16	1.13e−15	5.20e−15	2.60e−14	1.78e−11	4.65e−11	2.25e−16	1.92e−16	3.95e−01	1.21e+00	1.25e+02	1.92e+01
f_8	1.15e+04	1.19e+04	2.18e+04	1.33e+04	3.36e+04	1.72e+04	8.33e+11	0.00e+00	8.33e+11	0.00e+00	8.33e+11	3.09e+08	8.56e+11	3.36e+09
f_9	2.04e−08	7.48e−08	9.41e−05	2.69e−04	2.79e−05	1.30e−04	1.13e+03	1.87e+01	1.01e+03	1.30e+00	1.18e+03	8.29e+00	1.22e+03	1.79e+01
f_{10}	2.13e−31	6.55e−31	1.53e−13	7.65e−13	2.67e−28	1.33e−27	5.52e+01	1.02e+01	2.77e−28	5.31e−29	7.34e+01	6.25e+01	3.76e+01	2.78e+01
f_{11}	1.40e−06	6.89e−06	1.08e−05	1.84e−05	2.21e−05	8.45e−05	3.95e+02	6.11e+01	2.55e−04	3.20e−05	4.03e+02	8.45e+01	1.08e+03	8.25e+01
f_{12}	2.83e−09	1.42e−08	4.66e−06	2.33e−05	3.22e−08	1.52e−07	2.84e+02	4.97e+01	9.97e−10	2.01e−10	8.11e+02	1.58e+03	5.87e+02	4.52e+02
f_{13}	1.41e+02	1.95e+01	4.51e+01	4.37e+01	1.46e+02	1.20e+01	7.52e+02	2.71e+02	1.40e+02	1.26e+01	2.06e+08	3.51e+08	5.92e+02	1.08e+02
f_{14}	1.53e−09	6.89e−09	2.14e−01	4.20e−01	1.54e−05	5.74e−05	3.11e+02	3.66e+01	8.08e−03	4.04e−02	4.90e+02	5.23e+01	1.26e+03	1.81e+02
f_{15}	2.12e−25	1.05e−24	9.41e−15	3.35e−14	4.90e−27	1.67e−26	1.17e+01	2.85e+00	3.71e−24	2.32e−24	1.40e+01	9.80e+00	1.95e+02	1.66e+02
f_{16}	2.97e−11	8.57e−11	2.50e−08	1.20e−07	1.19e−07	3.50e−07	5.58e+02	7.96e+01	7.85e−09	1.11e−09	6.77e+02	6.04e+02	9.56e+02	3.33e+01
f_{17}	4.97e+01	3.50e+01	8.93e+00	1.01e+01	5.68e+01	2.39e+01	1.03e+03	1.11e+02	3.71e+01	8.30e−01	1.17e+07	1.70e+07	1.49e+03	8.00e+01
f_{18}	1.21e−08	3.69e−08	8.93e−02	2.75e−01	2.20e−04	4.28e−04	7.53e+02	6.55e+01	5.10e−04	9.97e−05	7.67e+02	2.14e+02	3.94e+03	3.91e+03
f_{19}	0.00e+00	0.00e+00	5.05e−26	2.49e−25	6.41e−24	3.12e−23	4.04e+01	7.71e+00	1.67e−22	7.58e−23	7.51e+02	1.76e+03	2.53e+04	2.45e+04

Table 12
Average errors and standard deviations of the three GPALS-DE parameterizations and the base algorithms for dimension $n = 500$.

Problem	GPALS-DE$_{0.5}$		GPALS-DE$_{0.2}$		GPALS-DE$_{0.8}$		DE$_{bin}$		DE$_{exp}$		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
f_1	1.04e−12	2.66e−13	1.86e−12	4.97e−13	9.75e−13	2.34e−13	3.88e−05	7.93e−05	5.17e−16	1.36e−17	9.25e+02	1.27e+03	n/a	n/a
f_2	7.40e+01	8.28e+00	8.20e+01	9.53e+00	8.31e+01	1.04e+01	1.25e+02	5.37e+00	5.38e+01	1.21e+00	1.35e+02	5.52e+00	n/a	n/a
f_3	4.99e+02	2.34e+01	1.92e+02	2.17e+02	4.88e+02	1.95e+01	3.44e+04	1.62e+05	4.74e+02	1.48e+00	6.93e+08	1.72e+09	n/a	n/a
f_4	1.70e−12	4.47e−13	6.98e−01	8.85e−01	1.65e−12	3.37e−13	2.35e+03	1.60e+02	7.12e−01	9.64e−01	2.11e+03	1.66e+02	n/a	n/a
f_5	5.35e−13	8.47e−14	1.01e−12	8.62e−13	5.89e−13	2.40e−13	3.11e−01	5.07e−01	2.38e−16	1.18e−17	1.45e+01	2.52e+01	n/a	n/a
f_6	1.24e−12	1.93e−13	1.09e−02	3.51e−02	1.17e−12	3.40e−13	1.49e+01	8.38e−01	1.64e−12	4.85e−14	1.27e+01	1.26e+00	n/a	n/a
f_7	0.00e+00	0.00e+00	6.31e−12	3.15e−11	0.00e+00	0.00e+00	2.74e−03	6.49e−03	7.29e−16	3.58e−16	3.33e−05	1.12e−04	n/a	n/a
f_8	1.07e+05	7.42e+04	2.38e+05	1.36e+05	2.00e+05	7.54e+04	4.94e+12	0.00e+00	4.94e+12	0.00e+00	4.94e+12	1.42e+08	n/a	n/a
f_9	1.64e−08	4.60e−08	2.65e−04	4.69e−04	1.54e−04	7.10e−04	2.97e+03	3.17e+01	2.52e+03	2.10e+00	3.00e+03	1.64e+01	n/a	n/a
f_{10}	1.42e−31	4.91e−31	4.77e−17	2.38e−16	4.01e−30	1.99e−29	1.36e+02	2.08e+01	9.79e−28	1.43e−28	1.64e+02	5.62e+01	n/a	n/a
f_{11}	0.00e+00	0.00e+00	3.04e−03	1.26e−02	9.77e−07	3.47e−06	2.34e+03	9.22e+01	6.78e−04	3.60e−05	1.67e+03	1.44e+02	n/a	n/a
f_{12}	9.80e−12	4.33e−11	1.76e−06	8.81e−06	4.32e−07	1.83e−06	1.02e+03	6.68e+01	6.80e−09	8.58e−10	1.62e+03	1.83e+03	n/a	n/a
f_{13}	3.79e+02	3.79e+01	8.69e+01	1.00e+02	3.75e+02	2.01e+01	2.49e+03	3.13e+02	3.60e+02	9.23e+00	3.41e+08	4.29e+08	n/a	n/a
f_{14}	9.10e−11	1.83e−10	8.93e−01	1.11e+00	1.29e−04	3.51e−04	1.67e+03	1.51e+02	3.93e−01	1.05e+00	1.59e+03	1.57e+02	n/a	n/a
f_{15}	0.00e+00	0.00e+00	5.53e−16	2.58e−15	3.82e−25	1.89e−24	4.44e+01	5.59e+00	2.93e−18	7.16e−18	3.50e+01	1.20e+01	n/a	n/a
f_{16}	1.46e−09	6.41e−09	6.06e−07	2.67e−06	2.17e−07	7.81e−07	2.02e+03	8.60e+01	2.05e−08	1.64e−09	1.92e+03	1.44e+03	n/a	n/a
f_{17}	1.37e+02	2.99e+01	3.81e+00	4.97e+00	1.24e+02	1.31e+01	3.83e+03	1.41e+02	1.12e+02	1.02e+00	6.64e+08	1.64e+09	n/a	n/a
f_{18}	4.89e−09	1.71e−08	2.92e−01	6.07e−01	7.81e−04	2.27e−03	3.37e+03	4.34e+02	1.25e−03	1.87e−04	2.74e+03	3.59e+02	n/a	n/a
f_{19}	1.04e−12	2.66e−13	4.86e−08	2.43e−07	9.47e−22	4.74e−21	1.29e+02	2.34e+01	3.35e−21	2.15e−21	2.05e+03	4.03e+03	n/a	n/a

The standard DE algorithm assumes a fixed-size population,

$$P = \{x_1, x_2, \ldots, x_N\},$$

of search points (agents) called *individuals*. Each individual defines a position in the search space,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in X, \qquad i \in I,$$

and it is randomly and uniformly initialized in X. At each iteration, *mutation* and *crossover* operators are applied on each individual to sample new points in the search space. This is followed by a *selection* phase where the generated new points compete against the original ones to form the population of the next iteration.

Let t denote the iteration counter. During mutation, a new vector $u_i^{(t+1)}$ is produced for each individual $x_i^{(t)}$ by combining randomly selected and mutually different members of the current population $P^{(t)}$. A variety of mutation operators have been proposed in the relevant literature. The most common ones are the following:

DE/best/1:

$$u_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{3}$$

DE/rand/1:

$$u_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)}\right), \tag{4}$$

DE/current-to-best/2:

$$u_i^{(t+1)} = x_i^{(t)} + F\left(x_g^{(t)} - x_i^{(t)} + x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{5}$$

DE/best/2:

$$u_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)} + x_{r_3}^{(t)} - x_{r_4}^{(t)}\right), \tag{6}$$

DE/rand/2:

$$u_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)} + x_{r_4}^{(t)} - x_{r_5}^{(t)}\right). \tag{7}$$

The user-defined parameter F, also called the *scale factor* [16], determines the step size of the algorithm and affects its exploration/exploitation trade-off. The index g denotes the best quality individual, i.e.,

$$g = \arg\min_{i \in I} \left\{f_i^{(t)}\right\}.$$

The indices $r_1, r_2, \ldots, r_5 \in I$, are randomly selected such that

$$r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i.$$

After mutation, crossover is applied to produce a new *trial vector*

$$v_i^{(t+1)} = \left(v_{i1}^{(t)}, v_{i2}^{(t)}, \ldots, v_{in}^{(t)}\right),$$

for each individual $x_i^{(t)} \in P^{(t)}$. A common crossover operator is the *binomial crossover*, where

$$v_{ij}^{(t+1)} = \begin{cases} u_{ij}^{(t+1)}, & \text{if } \mathtt{rand()} \leqslant CR \text{ or } j = R_i, \\ x_{ij}^{(t)}, & \text{otherwise,} \end{cases} \tag{8}$$

where $j \in D$ (see Eq. (2)); $CR \in (0, 1]$ is a user-defined *crossover rate*; and R_i is a random integer uniformly sampled from D. The use of R_i ensures that at least one component of the trial vector is inherited from the mutated vector.

Alternatively, the *exponential crossover* operator can be used. Specifically, $x_i^{(t)}$ is initially copied into the trial vector $v_i^{(t+1)}$. Then, starting from a randomly selected component $k \in D$, all components $u_{ik}^{(t+1)}$ are copied in $v_{ik}^{(t+1)}$ until a stochastic termination condition is satisfied. The rest of the components in $v_i^{(t+1)}$ retain the values initially copied from $x_i^{(t)}$ [13].

Eventually, the selection phase takes place, where $x_i^{(t)}$ competes against $v_i^{(t+1)}$ for inclusion in the new population,

$$x_i^{(t+1)} = \begin{cases} v_i^{(t+1)}, & \text{if } f\left(v_i^{(t+1)}\right) \leqslant f_i^{(t)}, \\ x_i^{(t)}, & \text{otherwise.} \end{cases} \tag{9}$$

The selection phase completes a full iteration. The algorithm continues with new iterations until a termination condition is met. Usually, this condition involves the desirable solution quality or the maximum computational budget. The best detected individual is eventually reported as the final solution. For a comprehensive presentation of the DE algorithm, the reader is referred to [13].

2.2. Online parameter adaptation approaches

During the past decades, various approaches for dynamic parameter adaptation have been proposed. In the following paragraphs we focus on approaches related to Differential Evolution, which was used to demonstrate our proposed method. Most of the presented approaches are particularly designed for the specific algorithm and specific test suites. As mentioned earlier, this overspecialization is a typical weakness of various tuning methods, thereby offering motivation for the development of alternative, more general methods.

The scale factor and crossover rate are the typically adapted DE parameters. An adaptive scale factor inheritance scheme where subpopulations are connected in a ring topology was proposed for a distributed DE in [17]. A different approach using multi-trajectory search was proposed in [18]. A self-adaptive algorithm that determines DE's control parameters through a probabilistic procedure was proposed in [19]. In [20–22], the reader can review further parameter adaptation schemes.

The JADE [23] algorithm uses an external archive to adapt the control parameters, while SHADE [24] uses historical data stored in memory in order to guide the parameter adaptation. Similarly, the enhanced L-SHADE performs parameter adaptation based on the sequence of successes [25]. In [26] a restarting DE variant was proposed, using the maximum distance among the individuals as the restarting condition.

Competition-based approaches for DE were proposed in [27, 28]. A self-adaptive DE with small population size was proposed in [29]. An adaptive memetic DE approach was proposed in [30]. Moreover, the jDElscop variant was introduced in [31], combining three strategies and a population-reduction mechanism. A general scheme was also proposed in [32]. Significant attributes of the DE algorithm were investigated in [33], where the relation between exploration and exploitation was studied. In [34] the two DE crossover variants were thoroughly analyzed.

Recently, a grid-based method [9] was proposed for dynamic parameter adaptation, and it was demonstrated on the DE algorithm. The so called DEGPA method adapts the scalar parameters and the mutation operator, based on short-run performance estimations under neighboring parameter values in a discretized parameter domain. The parameters are periodically updated, followed by a dynamic-deployment phase where the population is evolved with the selected parameters. Promising results for two evolutionary algorithms have verified the potential of this approach in low-dimensional and high-dimensional test problems [9–12].

The grid-based method consists of a rather general procedure applicable also to other metaheuristics, avoiding the overspecialization issue, while requiring only a small number of additional control parameters for the tuning method [10].

Table 13
Average errors of the GPALS-$DE_{0.5}$ approach against other algorithms for the high-dimensional test problems $f_1 - f_9$.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
GPALS-$DE_{0.5}$									
50	0.00e+00	2.35e+00	4.54e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.06e+02	1.94e−07
100	0.00e+00	1.32e+01	1.06e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.33e+03	0.00e+00
200	0.00e+00	3.37e+01	2.11e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.15e+04	2.04e−08
500	0.00e+00	7.40e+01	4.99e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.07e+05	1.64e−08
SOUPDE									
50	0.00e+00	1.18e+00	3.10e+01	3.98e−02	0.00e+00	1.47e−14	2.28e−14	9.69e−02	3.75e−06
100	0.00e+00	7.47e+00	7.92e+01	3.98e−02	0.00e+00	3.03e−14	3.88e−14	6.55e+01	7.82e−06
200	0.00e+00	2.38e+01	1.80e+02	1.19e−01	0.00e+00	6.40e−14	7.46e−14	2.46e+03	1.51e−05
500	0.00e+00	6.50e+01	4.71e+02	7.96e−02	0.00e+00	1.67e−13	1.78e−13	4.36e+04	3.59e−05
DE-$D^{40}+M^m$									
50	3.33e−18	1.67e−01	1.34e+01	1.99e−01	0.00e+00	4.55e−14	0.00e+00	6.11e−01	0.00e+00
100	2.78e−17	2.24e+00	7.61e+01	1.99e−01	1.39e−17	1.01e−13	5.33e−17	4.75e+05	4.29e−04
200	6.66e−17	9.58e+00	1.69e+02	2.39e−01	2.78e−17	2.51e−13	0.00e+00	2.19e+08	0.00e+00
500	2.23e−16	3.72e+01	4.54e+02	9.15e−01	1.03e−16	7.14e−13	0.00e+00	1.41e+10	6.78e−09
GaDE									
50	0.00e+00	1.46e+01	1.18e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.08e−08	6.24e−07
100	0.00e+00	3.88e+01	5.89e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.23e−03	3.87e−07
200	0.00e+00	5.76e+01	1.61e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.02e+00	4.53e−09
500	0.00e+00	7.42e+01	4.40e+02	0.00e+00	0.00e+00	1.46e−14	0.00e+00	1.33e+03	0.00e+00
jDElscop									
50	0.00e+00	3.15e−02	2.28e+01	0.00e+00	0.00e+00	9.55e−14	0.00e+00	9.97e−03	0.00e+00
100	0.00e+00	1.21e+00	6.13e+01	0.00e+00	0.00e+00	2.00e−13	0.00e+00	5.57e+00	7.18e−09
200	0.00e+00	7.54e+00	1.40e+02	0.00e+00	0.00e+00	4.52e−13	0.00e+00	2.52e+02	4.30e−08
500	0.00e+00	3.06e+01	4.06e+02	1.59e−01	0.00e+00	1.18e−12	0.00e+00	5.66e+03	6.10e−08
SaDE-MMTS									
50	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	4.13e−09	1.35e−01
100	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.05e−04	3.18e−01
200	0.00e+00	1.34e+00	0.00e+00	8.08e−02	0.00e+00	0.00e+00	0.00e+00	2.67e+01	1.24e+00
500	0.00e+00	1.25e+01	0.00e+00	3.85e+00	0.00e+00	0.00e+00	0.00e+00	3.01e+02	2.81e+01
MOS									
50	0.00e+00	4.64e−13	9.61e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.54e−08	0.00e+00
100	0.00e+00	2.94e−12	2.03e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	9.17e−02	0.00e+00
200	0.00e+00	1.24e−11	4.01e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.16e+02	0.00e+00
500	0.00e+00	5.51e−04	4.57e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.28e+04	0.00e+00
MA-SSW-Chains									
50	1.67e−17	7.61e−02	4.79e+01	1.19e−01	0.00e+00	4.89e−14	9.33e−17	3.06e−01	2.94e+02
100	2.78e−17	7.01e+00	1.38e+02	1.19e−01	1.39e−17	6.03e−14	8.17e−16	3.48e+01	5.63e+02
200	5.33e−17	3.36e+01	2.50e+02	4.43e+00	2.72e−17	1.19e−13	6.96e−15	7.23e+02	1.17e+03
500	1.01e−16	7.86e+01	6.07e+02	1.78e+02	7.70e−17	2.63e−13	4.69e−14	1.32e+04	2.53e+03

3. Proposed parameter adaptation method

The proposed approach, henceforth called *Gradient-based Parameter Adaptation with Line Search* (GPALS), draws inspiration from the grid-based DEGPA approach [9]. In DEGPA the parameter domain is discretized, hence forming a grid. For example, in DE the grid consists of all scalar parameter pairs (F, CR) in the domain $[0, 1] \times [0, 1]$, using a small fixed discretization step size $\lambda > 0$. DEGPA starts with an initial parameter pair (F_0, CR_0) and an initial *primary population*, which is evolved for T_p iterations using the parameters (F_0, CR_0). Then, the primary population is copied into nine *secondary populations*, each of which is evolved using one of the neighboring parameter pairs

$$(F_0 \pm i\,\lambda, CR_0 \pm j\,\lambda), \qquad i, j \in \{0, 1\}.$$

The secondary populations are independently evolved using their assigned parameter pairs for a small number of iterations, $T_s \ll T_p$, in order to locally estimate the performance of the current population with the alternative parameter pairs. The best-performing secondary population is then adopted as the new primary population along with its parameter pair. The considered performance measure for assessing the quality of a parameter pair (F, CR) is the *average objective function value* of its corresponding secondary population P_s after the T_s iterations, i.e.,

$$H\left(P_s, (F, CR)\right) = \frac{1}{N} \sum_{x \in P_s} f(x). \qquad (10)$$

This performance measure is preferred against the *best objective function value* because it is less sensitive to temporary quality surges of the best individual (e.g., due to rapidly detected local minimizers). Moreover, the best solutions of all unselected secondary populations are inherited in the new primary population, replacing equal number of worst individuals.

The procedure above constitutes a full cycle of the algorithm, which continues with new cycles until the available computational budget (function evaluations) is exceeded. Details and extensions of DEGPA can be found in [9,11]. Besides its simplicity, DEGPA exhibited remarkable competitiveness against various fine-tuned algorithms in high-dimensional problems [9]. This was achieved even without taking into account the demanding preliminary experimentation needed for the competitor approaches, thereby offering motivation for further investigation.

An obvious limitation of DEGPA is the constant step size λ used for the discretization of the parameter values in the grid. If superior parameter values exist in a specific direction in the grid and λ is too small, DEGPA may need a large number of steps to reach them. Conversely, if λ is too large then promising values may be overshot because the grid search directions are restricted to horizontal, vertical, and diagonal moves. These issues motivated the development of the proposed GPALS approach, where the grid is replaced by a dense parameter search space. In this framework, trajectories of parameter pairs are produced by following approximate gradient directions in the parameter domain, while line search is used to determine the appropriate step size.

Table 14
Average errors of the GPALS-DE$_{0.5}$ approach against other algorithms for the high-dimensional test problems $f_{10}-f_{19}$.

	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}
GPALS-DE$_{0.5}$										
50	0.00e+00	1.10e−06	0.00e+00	3.32e+01	0.00e+00	0.00e+00	0.00e+00	1.83e+00	9.71e−08	0.00e+00
100	0.00e+00	2.05e−08	0.00e+00	6.54e+01	0.00e+00	0.00e+00	0.00e+00	1.52e+01	0.00e+00	0.00e+00
200	0.00e+00	1.40e−06	0.00e+00	1.41e+02	0.00e+00	0.00e+00	0.00e+00	4.97e+01	1.21e−08	0.00e+00
500	0.00e+00	0.00e+00	0.00e+00	3.79e+02	0.00e+00	0.00e+00	0.00e+00	1.37e+02	0.00e+00	0.00e+00
SOUPDE										
50	0.00e+00	3.09e−06	0.00e+00	2.06e+01	0.00e+00	1.38e−14	0.00e+00	2.53e−01	0.00e+00	0.00e+00
100	0.00e+00	6.75e−06	0.00e+00	5.85e+01	9.09e−15	2.79e−14	0.00e+00	8.55e+00	0.00e+00	0.00e+00
200	0.00e+00	1.43e−05	0.00e+00	1.35e+02	3.98e−02	5.79e−14	0.00e+00	3.31e+01	0.00e+00	1.91e−14
500	0.00e+00	4.66e−04	0.00e+00	3.58e+02	1.31e−12	1.39e−13	0.00e+00	1.09e+02	2.82e−13	4.95e−14
DE-$D^{40}+M^{m}$										
50	1.89e−31	6.06e−04	1.58e−21	1.39e+01	1.19e−01	0.00e+00	1.76e−16	4.31e−02	3.98e−02	0.00e+00
100	0.00e+00	0.00e+00	4.62e−17	5.33e+01	1.19e−01	0.00e+00	8.99e−15	3.22e+00	8.11e−10	0.00e+00
200	3.51e+01	0.00e+00	5.45e−15	1.23e+02	3.98e−02	0.00e+00	2.26e−13	2.72e+01	3.98e−02	9.47e−32
500	2.43e−31	0.00e+00	5.05e−13	3.48e+02	2.39e−01	0.00e+00	4.17e−12	1.02e+02	9.97e−08	0.00e+00
GaDE										
50	0.00e+00	1.31e−06	0.00e+00	1.19e+01	9.78e−13	0.00e+00	4.78e−12	4.97e−01	4.82e−08	0.00e+00
100	0.00e+00	4.34e−07	0.00e+00	4.99e+01	7.90e−13	0.00e+00	2.45e−12	3.28e+00	1.96e−08	0.00e+00
200	4.20e−02	1.85e−07	4.92e−14	1.24e+02	2.87e−12	0.00e+00	1.58e−12	2.45e+01	2.53e−08	0.00e+00
500	3.78e−01	0.00e+00	1.07e−12	3.34e+02	2.79e−11	0.00e+00	1.67e−12	9.26e+01	5.59e−08	4.20e−02
jDElscop										
50	0.00e+00	0.00e+00	0.00e+00	1.36e+01	0.00e+00	0.00e+00	0.00e+00	7.43e−03	2.41e−14	0.00e+00
100	0.00e+00	8.17e−09	0.00e+00	5.11e+01	0.00e+00	0.00e+00	0.00e+00	3.21e−01	6.33e−14	0.00e+00
200	0.00e+00	9.58e−09	0.00e+00	1.10e+02	4.11e−16	0.00e+00	0.00e+00	2.39e+01	2.04e−13	0.00e+00
500	0.00e+00	4.40e−08	0.00e+00	3.14e+02	8.00e−02	0.00e+00	0.00e+00	7.65e+01	1.11e−12	0.00e+00
SaDE-MMTS										
50	0.00e+00	5.19e−05	0.00e+00	4.23e+00	3.93e−08	0.00e+00	0.00e+00	4.78e−01	9.38e−03	0.00e+00
100	0.00e+00	2.00e−04	0.00e+00	3.30e+01	1.02e−02	0.00e+00	0.00e+00	1.17e+01	4.70e−02	0.00e+00
200	0.00e+00	2.39e−04	0.00e+00	8.89e+01	1.57e−02	0.00e+00	0.00e+00	3.50e+01	3.35e−01	0.00e+00
500	0.00e+00	2.53e+01	0.00e+00	3.27e+02	4.01e−01	0.00e+00	0.00e+00	9.80e+01	1.18e+00	0.00e+00
MOS										
50	0.00e+00	0.00e+00	0.00e+00	4.55e−01	0.00e+00	0.00e+00	0.00e+00	1.40e+01	0.00e+00	0.00e+00
100	0.00e+00	0.00e+00	0.00e+00	1.75e+01	1.68e−11	0.00e+00	0.00e+00	1.43e+01	0.00e+00	0.00e+00
200	0.00e+00	0.00e+00	0.00e+00	9.03e+00	0.00e+00	0.00e+00	0.00e+00	5.03e+00	0.00e+00	0.00e+00
500	0.00e+00	0.00e+00	0.00e+00	3.78e+01	0.00e+00	0.00e+00	0.00e+00	1.21e+01	0.00e+00	0.00e+00
MA-SSW-Chains										
50	1.67e−30	4.49e−03	6.27e−41	3.02e+01	1.37e−17	3.91e−16	4.06e−03	2.60e+01	3.88e−19	4.02e−31
100	1.05e−29	1.09e−01	3.28e−03	8.35e+01	2.21e−16	1.59e−15	1.61e−02	9.92e+01	2.71e−18	3.15e−30
200	5.41e−29	3.50e−01	1.75e−02	1.68e+02	9.76e−01	5.32e−15	6.02e−02	7.55e+01	4.29e−04	1.51e−16
500	2.80e−01	4.21e+01	2.55e+01	4.00e+02	5.65e+01	5.53e+00	1.08e−01	1.38e+02	2.41e−03	7.84e−17

These procedures require estimations of the algorithm's performance under different parameter settings. Similarly to DEGPA, the estimations are based on short runs of the algorithm using the corresponding parameter pairs. The runs can be conducted either serially or in parallel. Naturally, the main concept of GPALS can be generalized to any algorithm and arbitrary number of parameters.

Let us use DE as an example to describe a complete cycle of the proposed GPALS method. Consider the general optimization problem of Eq. (1), let N be the selected population size, and $[F_{\min}, F_{\max}]$, $[CR_{\min}, CR_{\max}]$, be prescribed ranges for the parameters F and CR, respectively. Then, the parameter domain is defined as

$$G = [F_{\min}, F_{\max}] \times [CR_{\min}, CR_{\max}] .$$

GPALS assumes a *primary population* P_{pri}, sampled in the search space X of the problem at hand. The primary population is assigned a *current parameter pair* $(F_c, CR_c) \in G$. Initially, this pair can be either the central point of G or a randomly selected point in G. Naturally, if additional information is available regarding the most promising parameter values (e.g., due to previous experimentation), the initial pair can be properly adjusted.

After initialization, the primary population is evolved for T_p iterations. According to previous works [9,11], we suggest the value

$$T_p = 10 \times n,$$

where n is the dimension of the considered optimization problem. In general, T_p shall be adequate for the algorithm to deploy its dynamics, but not so large that would rapidly consume the available computational budget.

For the evolved primary population with the current parameter pair, the performance measure $H\left(P_{\text{pri}}, (F_c, CR_c)\right)$ is computed according to Eq. (10). Then, three main procedures take place, namely the *performance gradient estimation*, the *line search*, and the *dynamic deployment*, which are described below.

3.1. Performance gradient estimation

The gradient estimation phase aims at detecting a direction in G at which promising parameters of the algorithm exist. Such parameters shall improve the current population in terms of the performance criterion of Eq. (10). More specifically, we estimate the negative of the approximate gradient of H at (F_c, CR_c) as

$$-\nabla H\left(P_{\text{pri}}, (F_c, CR_c)\right) = -\begin{pmatrix} \frac{\partial H\left(P_{\text{pri}}, (F_c, CR_c)\right)}{\partial F_c} \\[2ex] \frac{\partial H\left(P_{\text{pri}}, (F_c, CR_c)\right)}{\partial CR_c} \end{pmatrix}, \tag{11}$$

where the partial derivatives are computed using the symmetric difference quotient,

$$\frac{\partial H\left(P_{\text{pri}}, (F_c, CR_c)\right)}{\partial F_c} =$$

Table 15
Average error of the SHADE variants for the high-dimensional problems $f_1 - f_9$.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
SHADE$_{60}$									
50	0.00e+00	1.22e+01	7.81e+02	3.98e−02	4.13e−03	2.46e−01	0.00e+00	0.00e+00	9.00e−01
100	0.00e+00	5.86e+01	8.27e+02	0.00e+00	1.70e−02	2.49e+00	0.00e+00	9.06e−05	2.81e+01
200	0.00e+00	8.33e+01	1.07e+03	1.43e+00	5.44e−02	5.51e+00	0.00e+00	2.41e+00	4.68e+01
500	2.01e−09	1.05e+02	1.79e+03	1.06e+02	4.90e−01	1.37e+01	6.05e−05	2.38e+03	3.73e+01
SHADE$_{100}$									
50	0.00e+00	2.24e−01	7.81e+02	0.00e+00	6.90e−04	0.00e+00	0.00e+00	4.22e−09	9.85e−02
100	0.00e+00	4.41e+01	8.36e+02	0.00e+00	3.83e−03	1.45e+00	0.00e+00	5.40e−04	1.42e+01
200	0.00e+00	7.13e+01	1.07e+03	3.98e−02	2.47e−02	3.43e+00	0.00e+00	1.85e+00	1.43e+02
500	0.00e+00	9.55e+01	1.79e+03	1.48e+01	2.56e−01	1.09e+01	1.10e−05	1.66e+03	2.56e+02
mSHADE$_{60}$									
50	0.00e+00	2.15e+01	7.98e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	9.02e−07	4.30e−03
100	0.00e+00	6.82e+01	8.48e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.56e−03	1.10e−02
200	0.00e+00	9.51e+01	9.90e+02	0.00e+00	6.90e−04	0.00e+00	0.00e+00	1.06e+01	3.08e−02
500	4.93e−06	1.17e+02	1.97e+03	5.66e−09	1.09e−01	9.75e−01	1.26e−03	3.81e+03	7.49e−02
mSHADE$_{100}$									
50	0.00e+00	5.60e+00	8.13e+02	0.00e+00	0.00e+00	2.63e−06	5.75e−07	6.45e−03	6.98e−01
100	0.00e+00	4.37e+01	8.57e+02	0.00e+00	0.00e+00	3.23e−06	1.55e−06	1.90e−01	1.70e+00
200	0.00e+00	8.06e+01	9.71e+02	0.00e+00	0.00e+00	3.05e−06	3.55e−06	3.04e+01	4.07e+00
500	0.00e+00	1.07e+02	1.76e+03	0.00e+00	5.39e−03	5.93e−07	3.22e−04	3.64e+03	1.22e+01
L-SHADE									
50	0.00e+00	5.32e−07	8.03e+02	1.03e−01	0.00e+00	0.00e+00	0.00e+00	5.41e−10	1.39e−01
100	0.00e+00	2.40e−03	8.62e+02	2.81e+01	2.96e−04	0.00e+00	0.00e+00	5.43e−02	4.09e+00
200	0.00e+00	3.64e−01	9.71e+02	2.26e+02	1.09e−03	0.00e+00	1.59e−06	9.52e+01	5.55e+01
500	–	–	–	–	–	–	–	–	–
mL-SHADE									
50	6.45e−05	5.91e+00	8.27e+02	1.75e−02	6.43e−05	2.34e−03	1.41e−03	4.59e+00	4.16e+00
100	3.89e−01	2.55e+01	1.17e+03	1.52e+01	2.36e−01	1.87e−01	3.43e−01	1.01e+03	6.02e+01
200	1.20e+02	5.24e+01	7.93e+04	2.91e+02	1.83e+00	1.87e+00	9.68e+00	1.47e+04	4.04e+02
500	–	–	–	–	–	–	–	–	–

$$\frac{H\left(P_{\text{pri}},(F_c+\lambda,CR_c)\right)-H\left(P_{\text{pri}},(F_c-\lambda,CR_c)\right)}{2\,\lambda}, \qquad (12)$$

and

$$\frac{\partial H\left(P_{\text{pri}},(F_c,CR_c)\right)}{\partial CR_c} = \frac{H\left(P_{\text{pri}},(F_c,CR_c+\lambda)\right)-H\left(P_{\text{pri}},(F_c,CR_c-\lambda)\right)}{2\,\lambda}. \qquad (13)$$

This derivative estimation is preferred against the simple forward difference formula due to its higher accuracy, despite its requirement for two function evaluations at each estimation. In practice, the estimated quantities in the nominators are computed by copying the primary population P_{pri} into four secondary populations, each one assigned one of the four parameter pairs

$$(F_c+i\,\lambda,CR_c), \qquad (F_c,CR_c+i\,\lambda), \qquad i\in\{-1,+1\}.$$

Each secondary population is evolved for a small number of iterations, T_s, using its assigned parameter pair. Based on our previous experience with DEGPA, we suggest $T_s \in [5, 10]$. This is reasonable since we need to estimate only the local performance trend of the algorithm. In our implementation, we used $T_s = 10$, while $\lambda = 0.1$ was used as the standard step size for the gradient estimation. This choice is based on our observations suggesting that smaller step sizes produce marginal performance differences in DE [9]. Also, recent sensitivity analysis on the precursor grid-based approach supports this claim [35]. Naturally, different algorithms with higher parameter sensitivity may require smaller step sizes. Also, the computed gradient vectors are also normalized to become unit vectors, in order to alleviate scaling issues in the forthcoming steps.

The gradient estimation phase can be conducted either serially or in parallel. In the serial case, the four secondary populations are sequentially evolved for T_s iterations and their final performance values are recorded. In the parallel case, the procedure becomes more efficient by evolving each secondary population on a different CPU. Obviously, further parallelization within each secondary population (e.g., through fork-join procedures) can be used if additional CPUs are available. Note that modern desktop systems usually offer at least four CPUs.

An arguable issue in the above procedure is the comparability of the estimated performance values of the secondary populations, due to the different sequences of random numbers used in each. Indeed, it is hard to distinguish whether the observed performance differences are the outcome of the different parameters or random fluctuations due to the different sequences of random numbers. Such performance fluctuations may distort the computed gradient directions in the parameter domain. In order to ameliorate this effect, the four secondary populations assume the same sequence of random numbers by seeding their random number generators with the same random seed prior to each performance estimation phase. Thus, the parameter pairs become the sole source of variability among the subpopulations in the short runs, thereby providing comparable estimations.

Naturally, even in this case the performance estimations do not provide exact gradient directions. However, it shall be taken into consideration that DE is a stochastic algorithm. Hence, possible advantages of using exact gradient directions will be most-probably absorbed in the long-run due to the algorithm's stochasticity. Instead, the main purpose of using the gradient estimations is the identification of directions (or trends) in the parameter domain that seem to locally improve the algorithm's performance. This is evidently achieved by using the proposed procedure.

3.2. Line search

The gradient estimation phase determines a direction in the parameter domain that locally improves the algorithm's performance. Given this direction, a mechanism is needed to determine the corresponding step size. In mathematical optimization this is typically addressed through line search. An estimation-based

Table 16
Average error of the SHADE variants for the high-dimensional problems $f_{10} - f_{19}$.

	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}
SHADE$_{60}$										
50	3.47e+00	4.52e−01	4.05e+00	4.20e+01	0.00e+00	1.00e−01	7.18e−01	4.49e+00	5.79e−09	1.45e+00
100	3.10e+01	1.87e+01	7.11e+01	1.72e+02	0.00e+00	2.88e+00	5.65e+01	1.52e+02	2.11e−06	1.88e+01
200	8.45e+01	4.46e+01	2.85e+02	6.35e+02	5.33e−01	1.26e+01	5.49e+02	1.05e+03	4.61e−02	5.88e+01
500	2.26e+02	4.51e+01	1.00e+03	2.65e+03	4.84e+01	5.68e+01	1.84e+03	3.50e+03	8.13e+00	1.76e+02
SHADE$_{100}$										
50	2.81e−01	5.99e−02	0.00e+00	1.31e+01	2.93e−06	0.00e+00	8.58e−04	1.19e+00	2.61e−01	0.00e+00
100	2.45e+01	1.62e+01	3.37e+01	1.16e+02	0.00e+00	8.81e−01	3.38e+01	5.35e+01	1.07e+00	1.05e+01
200	8.14e+01	1.24e+02	2.28e+02	4.83e+02	4.29e−04	1.05e+01	3.89e+02	8.32e+02	5.43e+00	5.75e+01
500	2.47e+02	2.54e+02	7.90e+02	1.96e+03	8.24e+00	5.10e+01	1.61e+03	3.03e+03	3.36e+01	1.91e+02
mSHADE$_{60}$										
50	0.00e+00	4.56e−03	0.00e+00	1.41e+01	0.00e+00	0.00e+00	3.09e−06	4.82e−01	9.20e−05	0.00e+00
100	0.00e+00	1.26e−02	0.00e+00	6.59e+01	0.00e+00	0.00e+00	5.93e−06	7.60e+00	1.96e−04	0.00e+00
200	0.00e+00	2.96e−02	6.16e−01	1.98e+02	4.79e−09	1.41e+00	4.50e−06	8.98e+01	4.66e−04	3.36e+00
500	3.27e+01	7.59e−02	3.07e+02	1.69e+03	8.51e−03	3.02e+01	9.76e+01	7.09e+02	1.19e−02	9.95e+01
mSHADE$_{100}$										
50	0.00e+00	7.06e−01	8.99e−03	3.10e+01	8.83e−03	2.98e−07	6.48e−02	9.18e+00	1.19e−01	1.82e−09
100	0.00e+00	1.70e+00	2.75e−02	6.31e+01	2.17e−02	8.88e−07	1.35e−01	2.92e+01	2.59e−01	3.15e−08
200	0.00e+00	4.12e+00	7.41e−02	1.73e+02	4.20e−02	2.12e−06	2.83e−01	6.58e+01	5.48e−01	2.10e−01
500	7.99e+00	1.23e+01	2.04e+01	1.07e+03	5.78e−02	2.08e+01	3.31e+00	3.81e+02	1.41e+00	1.91e+01
L-SHADE										
50	1.65e−02	1.20e−01	4.29e−04	5.61e+01	2.68e+00	0.00e+00	2.54e−01	1.04e+01	4.49e+00	0.00e+00
100	8.21e−01	7.03e+00	2.34e+01	1.52e+02	3.49e+01	7.46e−09	1.41e+01	4.80e+01	3.97e+01	2.92e−01
200	1.47e+01	5.74e+01	2.05e+02	3.32e+02	2.01e+02	8.27e−02	1.84e+02	4.66e+02	1.36e+02	4.74e+00
500	–	–	–	–	–	–	–	–	–	
mL-SHADE										
50	2.17e−04	4.30e+00	2.18e−01	3.61e+01	3.27e−01	1.54e−03	9.16e−01	1.46e+01	9.88e−01	3.40e−04
100	1.27e+00	5.86e+01	8.04e+00	9.37e+01	1.83e+01	6.79e−01	2.61e+01	8.54e+01	1.74e+01	1.03e+00
200	3.17e+01	3.98e+02	8.83e+01	6.68e+03	2.34e+02	1.45e+01	2.06e+02	3.17e+02	1.40e+02	2.80e+01
500	–	–	–	–	–	–	–	–	–	

Algorithm 1: Bracketing and Bisection.

Input: x (current point); l, u (bracketing scalars); g (gradient vector); G (parameters domain);

```
/* Bracketing */
l ← 0, u ← 0.5
y ← x − u g
while (y ∈ G) do
    l ← u, u ← u + 0.5
    y ← x − u g
end while
/* Bisection */
while (u > l) do
    μ ← 0.5 (l + u)
    y ← x − μ g
    if (y ∉ G) then
        u ← μ
    else
        l ← μ
    end if
y ← x − μ g
end while
s4 ← 0.5 (l + u) g
return s4
```

analogue of line search is adopted in the proposed GPALS approach to refine the outcome of the procedures described in the previous section.

Specifically, line search is used to determine the appropriate step size $s > 0$, in order to generate the new parameter pair as follows,

$$(F, CR) = (F_c, CR_c) - s \nabla H \left(P_{\text{pri}}, (F_c, CR_c)\right). \tag{14}$$

For this purpose, the derivative-free line search algorithm proposed in [36] is adopted. The specific approach is based on the well known golden section method and has rapid convergence properties. For its application, four step size values are required,

$$s_1 < s_2 < s_3 < s_4,$$

each one defining a different point in the parameter domain through Eq. (14). The first value is taken as $s_1 = 0$ and corresponds to the current parameter pair (F_c, CR_c). The rest are defined according to the golden section approach as

$$s_2 = s_4 - \gamma\,\Delta, \qquad s_3 = s_1 + \gamma\,\Delta,$$

where $\Delta = s_4 - s_1$, and $\gamma = (\sqrt{5} - 1)/2$. The step size s_4 corresponds to the intersection point of the estimated gradient direction with the boundary of the parameter domain G. Thus, it is determined through a bracketing and bisection procedure, which is reported in Algorithm 1 [36]. Note that this procedure does not add any computational overhead in terms of function evaluations.

The line search iterates on the step sizes until a termination condition is met. The condition is related to the distance $0.5\,(s_4 - s_1)$. A reasonable choice is to terminate line search when this quantity becomes less than the reference step size λ that is used for the gradient estimation. For a more thorough description of the line search procedure, the reader is referred to [36].

The parameter pairs that correspond to the four step sizes in the line search procedure are evaluated through short runs of secondary populations, following the same procedure as in the gradient estimation phase. Thus, four secondary populations, $P_{s_1}, \ldots, P_{s_4}$, are initiated as copies of the primary population P_{pri}. Then, each step size s_i is evaluated by performing T_s iterations on P_{s_i}, using the parameter pair

$$(F_{s_i}, CR_{s_i}) = (F_c, CR_c) - s_i \nabla H \left(P_{s_i}, (F_c, CR_c)\right).$$

Eventually, line search provides a step size s^* that corresponds to the improving parameter pair

$$\left(F^*, CR^*\right) = (F_c, CR_c) - s^* \nabla H \left(P_{s^*}, (F_c, CR_c)\right). \tag{15}$$

Algorithm 2: Workflow of the proposed GPALS method.

```
Initialize( )
while (not termination) do
    /* Performance Gradient Estimation */
    Gradient-Estimation( )
    Normalization( )
    /* Line Search */
    Bracketing( )
    Golden-Section( )
    /* Dynamic Deployment */
    Update-Population( )
    Evolve-Population( )
end while
```

This parameter pair and its corresponding secondary population P_{s^*} are used in the next phase of the algorithm.

If all the estimated partial derivatives are close to zero within a prescribed tolerance $\delta > 0$ (10^{-8} was used in our implementation) for a parameter pair, then a local minimizer in the parameter domain may have been reached. In this case, line search is temporarily abandoned until at least one partial derivative becomes higher than the prescribed tolerance δ. However, the gradients are still computed in every cycle of the algorithm because the evolved primary population may perform better with different parameter values after some iterations. Alternatively, the parameter search procedure can be restarted if adequate computational budget is still available.

3.3. Dynamic deployment phase

In this phase, the best-performing secondary population P_{s^*} becomes the primary population, and its parameter pair (F^*, CR^*) replaces the current parameter pair (F_c, CR_c) if adequate performance improvement is achieved, i.e.,

$$H\left(P_{\text{pri}}, (F_c, CR_c)\right) - H\left(P_{s^*}, \left(F^*, CR^*\right)\right) > \theta_{\min} \geqslant 0,$$

where $\theta_{\min}$ is the smallest acceptable improvement (this is an optional parameter that is not used in our experiments). This step completes the GPALS cycle, and a new cycle begins by performing T_p iterations with the new primary population and the new parameter pair. In case of insufficient improvement from the secondary populations, the new cycle retains the previous primary population and parameter pair. In any case, the new primary population inherits the best individuals of all subpopulations. Thus, good solutions that are sampled during the performance estimation procedure are not neglected.

The GPALS workflow is outlined in Algorithm 2 and graphically illustrated in Fig. 1. Sample trajectories of DE parameters for two test problems used in our experimental evaluation (see next section) are illustrated in Fig. 2. The estimated gradient directions are shown in red dashed lines, while two parameter trajectories appear in darker colors with different markers denoting different problems. The memory complexity of GPALS is linearly increased compared to the plain DE algorithm due to the use of the subpopulations. If N denotes the population size and n the problem dimension, DE requires Nn memory positions while GPALS require $5Nn$. For the typical population sizes of hundreds (or even thousands) of individuals, this increase is definitely bearable in any modern desktop system. Moreover, the proposed approach does not add in terms of computational burden since it is supposed to use exactly the same budget of function evaluations as the plain DE. The gradient estimation and line search procedures add a fixed number of mathematical operations, while the ratio of calls of the random number generator to the number of function evaluations is the same as for DE. Thus, there are no time-consuming procedures introduced by GPALS.

Perhaps the most critical parts of the method is the estimation phase, especially the length of the short runs, followed by the dynamic deployment phase. Both these procedures are strongly based on the available computational budget. Subsequently follows the gradient step size λ, which is related to the algorithm and the desirable level of search granularity in the parameter domain. Although our presentation is tailored to the DE case, all procedures can be straightforwardly modified to fit the framework of other metaheuristics.

4. Experimental evaluation

The proposed GPALS method is demonstrated on the state-of-the-art DE algorithm. The derived scheme is henceforth denoted as GPALS-DE. Our experiments were conducted using two established test suites that comprise low-dimensional and high-dimensional problems. According to our analysis in the previous sections, the parameters of GPALS assumed the following values in all experiments:

$$T_p = 10n, \quad T_s = 10, \quad \lambda = 0.1, \quad \delta = 10^{-8}.$$

In the following paragraphs, the experimental setting is analyzed and the obtained results are reported and discussed for each test suite.

4.1. Test suites

It is reasonable to expect that parameter adaptation methods such as GPALS can be beneficial for an algorithm particularly in cases of computationally demanding problems. Undoubtedly, large-scale problems consisting of hundreds of decision variables constitute the appropriate testbed for investigating such potential benefits.

For this reason, we considered the test suite provided in the *special issue on large-scale continuous optimization problems* of the Soft Computing journal [37]. This test suite consist of 19 large-scale continuous optimization problems, henceforth denoted as f_1–f_{19}, of dimension $n = 50, 100, 200, 500$. Among them, f_1–f_6 come from the CEC 2008 test suite, accompanied by 13 shifted hybrid test problems of high complexity. Both separable and non-separable problems are included. The main goal determined by the test suite is the detection of the known global minimizers of the test problems within the tight limit of $T_{\max} = 5000n$ function evaluations (FEs).

The test suite provides an internet repository[1] with complementary material. This includes complete results for three base algorithms, namely DE with exponential crossover, CHC [38], and GCMAES [39], along with the average solution values for 13 additional algorithms, namely SOUPDE [40], DE-D^{40}+M^{m} [41], GaDE [42], jDElscop [31], SaDE-MMTS [18], MOS [43], MA-SSW-Chains [44], RPSO-vm [45], Tuned IPSOLS [46], EvoPROpt [47], EM323 [48], VXQR1 [49], and GODE [50]. Note that adaptive and self-adaptive algorithms are included among them. Besides the exponential DE, we additionally considered the most popular binomial DE as a base algorithm. The source code of the test problems is available online [51]. The solution quality criterion for all problems is the objective value error,

$$\varepsilon^* = f\left(x^*\right) - f\left(x_{\text{opt}}\right), \tag{16}$$

where x^* is the solution achieved by the algorithm and x_{opt} is the known globally optimal solution of the problem.

1 http://sci2s.ugr.es.

Table 17
Average errors for the CEC 2013 test problems $f_1 - f_9$.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
mGPALS-DE									
30	0.00e+00	1.40e+05	2.83e+07	2.49e+03	0.00e+00	4.40e+00	1.03e+01	2.09e+01	2.19e+01
50	0.00e+00	2.78e+05	4.89e+08	6.43e+03	0.00e+00	4.36e+01	3.17e+01	2.11e+01	4.10e+01
GPALS-DE									
30	0.00e+00	1.37e+07	1.58e+09	2.42e+04	0.00e+00	2.51e+01	1.08e+02	2.10e+01	2.71e+01
50	0.00e+00	3.28e+07	1.08e+10	4.46e+04	0.00e+00	4.60e+01	1.48e+02	2.11e+01	5.30e+01
mSHADE									
30	0.00e+00	1.97e+07	1.50e+09	2.91e+01	0.00e+00	2.48e+01	6.66e+01	2.08e+01	2.80e+01
50	0.00e+00	5.08e+07	1.59e+10	3.85e+01	0.00e+00	4.49e+01	1.39e+02	2.09e+01	5.56e+01
SHADE									
30	0.00e+00	9.00e+03	4.02e+01	1.92e−04	0.00e+00	5.96e−01	4.60e+00	2.07e+01	2.75e+01
50	0.00e+00	2.66e+04	8.80e+05	1.61e−03	0.00e+00	4.28e+01	2.33e+01	2.09e+01	5.54e+01
mL-SHADE									
30	0.00e+00	1.11e+07	1.17e+08	3.22e+03	0.00e+00	1.69e+01	6.96e+01	2.08e+01	2.65e+01
50	0.00e+00	2.00e+07	1.28e+09	1.00e+05	0.00e+00	4.34e+01	1.11e+02	2.10e+01	5.45e+01
L-SHADE									
30	0.00e+00	0.00e+00	1.64e+00	0.00e+00	0.00e+00	1.17e−06	7.79e−01	2.08e+01	2.64e+01
50	0.00e+00	1.96e+03	1.50e+04	0.00e+00	0.00e+00	4.34e+01	1.93e+00	2.10e+01	5.37e+01
DEcfbLS									
30	0.00e+00	1.99e+05	2.11e+06	3.82e+02	0.00e+00	7.08e+00	5.68e+01	2.09e+01	2.40e+01
50	0.00e+00	6.55e+05	2.20e+08	1.21e+03	0.00e+00	4.34e+01	1.05e+02	2.11e+01	4.71e+01
jande									
30	0.00e+00	1.29e+05	9.84e+06	1.97e+04	1.26e−08	7.93e+00	9.82e+00	2.09e+01	2.10e+01
50	2.76e−08	6.05e+05	4.78e+07	8.34e+04	2.43e−06	4.30e+01	2.94e+01	2.11e+01	5.33e+01
DE_APC									
30	0.00e+00	1.75e+05	3.21e+06	2.20e−01	0.00e+00	9.35e+00	2.18e+01	2.09e+01	3.07e+01
50	0.00e+00	3.60e+05	6.98e+06	1.53e+00	0.00e+00	3.90e+01	3.66e+01	2.11e+01	6.09e+01
PVADE									
30	0.00e+00	2.12e+06	1.65e+03	1.70e+04	1.40e−07	8.29e+00	1.29e+00	2.09e+01	6.30e+00
50	0.00e+00	2.04e+05	7.48e+06	2.20e+02	1.39e−03	7.36e+01	2.07e+01	2.11e+01	2.60e+01

Table 18
Average errors for the CEC 2013 test problems $f_{10} - f_{19}$.

	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}
mGPALS-DE										
30	5.26e−02	8.71e+00	3.46e+01	7.42e+01	4.27e+02	5.38e+03	2.56e+00	3.99e+01	1.38e+02	2.72e+00
50	3.61e−02	4.24e+01	8.39e+01	2.01e+02	8.79e+02	1.04e+04	3.27e+00	9.34e+01	3.10e+02	1.44e+01
GPALS-DE										
30	6.67e+00	0.00e+00	1.14e+02	1.57e+02	2.08e−01	4.28e+03	1.51e+00	3.04e+01	1.65e+02	1.05e+00
50	2.13e+01	0.00e+00	2.97e+02	3.97e+02	2.32e−01	8.52e+03	1.89e+00	5.08e+01	3.68e+02	1.94e+00
mSHADE										
30	1.99e+01	0.00e+00	1.30e+02	1.59e+02	2.86e−03	4.79e+03	1.50e+00	3.04e+01	1.05e+02	1.56e+00
50	6.11e+01	0.00e+00	3.43e+02	3.10e+02	8.08e−03	9.32e+03	1.71e+00	5.08e+01	2.02e+02	2.87e+00
SHADE										
30	7.69e−02	0.00e+00	2.30e+01	5.03e+01	3.18e−02	3.22e+03	9.13e−01	3.04e+01	7.25e+01	1.36e+00
50	7.37e−02	0.00e+00	5.86e+01	1.45e+02	3.45e−02	6.82e+03	1.28e+00	5.08e+01	1.37e+02	2.64e+00
mL-SHADE										
30	2.15e−02	0.00e+00	9.46e+01	1.34e+02	1.27e−02	4.36e+03	1.32e+00	3.04e+01	1.65e+02	1.62e+00
50	7.94e−02	0.00e+00	1.41e+02	2.35e+02	3.45e−02	7.56e+03	1.17e+00	5.08e+01	2.17e+02	1.63e+00
L-SHADE										
30	9.67e−04	0.00e+00	5.48e+00	6.34e+00	7.63e−02	2.68e+03	6.54e−01	3.04e+01	5.23e+01	1.21e+00
50	1.27e−02	4.66e−09	1.41e+01	2.12e+01	4.02e−01	6.36e+03	1.26e+00	5.08e+01	1.04e+02	2.56e+00
DEcfbLS										
30	2.01e−02	5.85e−02	5.42e+01	1.01e+02	3.29e+01	3.43e+03	7.27e−01	3.53e+01	7.94e+01	1.50e+00
50	3.21e−02	4.64e+00	1.34e+02	2.47e+02	2.31e+02	6.25e+03	1.63e+00	6.58e+01	1.57e+02	2.95e+00
jande										
30	7.91e−02	0.00e+00	4.28e+01	7.08e+01	1.33e+00	4.83e+03	2.28e+00	3.04e+01	1.23e+02	1.10e+00
50	1.47e−01	1.95e−02	9.72e+01	1.76e+02	8.01e+00	9.48e+03	3.13e+00	5.08e+01	2.18e+02	2.24e+00
DE_APC										
30	6.42e−02	3.08e+00	3.17e+01	7.55e+01	3.84e+03	4.14e+03	2.46e+00	5.92e+01	6.04e+01	2.30e+00
50	6.71e−02	3.44e+01	5.96e+01	1.55e+02	9.96e+03	9.34e+03	3.24e+00	1.72e+02	1.05e+02	5.08e+00
PVADE										
30	2.16e−02	5.84e+01	1.15e+02	1.31e+02	3.20e+03	5.61e+03	2.39e+00	1.02e+02	1.82e+02	5.40e+00
50	5.99e−01	1.68e+02	2.57e+02	3.06e+02	7.34e+03	1.25e+04	3.39e+00	2.38e+02	3.87e+02	2.12e+01

In addition to the large-scale problems, we also considered the mainstream *CEC 2013* test suite from the special session on real-parameter single-objective optimization [52]. It consists of 28 benchmark problems denoted as f_1–f_{28}, including unimodal,

Table 19
Average errors for the CEC 2013 test problems $f_{20} - f_{28}$.

	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}
mGPALS-DE									
30	1.13e+01	3.15e+02	3.07e+02	4.89e+03	2.15e+02	2.48e+02	2.20e+02	5.19e+02	3.01e+02
50	2.08e+01	8.45e+02	6.13e+02	1.04e+04	2.51e+02	2.92e+02	2.95e+02	9.11e+02	5.90e+02
GPALS-DE									
30	1.32e+01	2.70e+02	4.76e+01	4.87e+03	2.71e+02	2.70e+02	2.01e+02	8.00e+02	3.00e+02
50	2.34e+01	3.65e+02	2.25e+01	9.88e+03	3.41e+02	3.37e+02	2.03e+02	1.59e+03	4.00e+02
mSHADE									
30	1.12e+01	2.26e+02	4.62e+01	5.57e+03	2.75e+02	2.93e+02	2.01e+02	6.07e+02	3.00e+02
50	2.12e+01	2.74e+02	3.83e+01	1.10e+04	3.50e+02	3.88e+02	2.02e+02	1.57e+03	4.00e+02
SHADE									
30	1.05e+01	3.09e+02	9.81e+01	3.51e+03	2.05e+02	2.59e+02	2.02e+02	3.88e+02	3.00e+02
50	1.93e+01	8.45e+02	1.33e+01	7.63e+03	2.34e+02	3.40e+02	2.58e+02	9.36e+02	4.58e+02
mL-SHADE									
30	1.20e+01	4.23e+02	1.11e+02	4.96e+03	2.68e+02	2.87e+02	2.00e+02	1.00e+03	3.00e+02
50	2.24e+01	3.39e+02	1.39e+01	9.31e+03	3.47e+02	3.82e+02	2.06e+02	1.74e+03	4.00e+02
L-SHADE									
30	9.48e+00	2.96e+02	1.09e+02	2.51e+03	2.00e+02	2.40e+02	2.00e+02	3.02e+02	3.00e+02
50	1.78e+01	8.27e+02	1.47e+01	5.60e+03	2.12e+02	2.77e+02	2.33e+02	4.19e+02	4.00e+02
DEcfbLS									
30	1.17e+01	3.36e+02	2.56e+02	3.59e+03	2.64e+02	2.83e+02	2.00e+02	9.38e+02	3.00e+02
50	2.17e+01	5.24e+02	6.89e+02	7.77e+03	3.31e+02	3.60e+02	2.00e+02	1.55e+03	4.00e+02
jande									
30	1.16e+01	2.94e+02	5.16e+01	4.61e+03	2.48e+02	2.60e+02	2.58e+02	7.22e+02	3.00e+02
50	2.15e+01	8.24e+02	3.10e+01	9.48e+03	2.89e+02	3.17e+02	3.97e+02	1.16e+03	9.43e+02
DE_APC									
30	1.26e+01	2.67e+02	4.56e+03	4.18e+03	2.92e+02	2.99e+02	3.28e+02	1.19e+03	3.00e+02
50	2.23e+01	6.81e+02	1.06e+04	9.09e+03	3.84e+02	3.83e+02	4.09e+02	2.14e+03	6.97e+02
PVADE									
30	1.13e+01	3.19e+02	2.50e+03	5.81e+03	2.02e+02	2.30e+02	2.18e+02	3.26e+02	3.00e+02
50	2.07e+01	9.65e+02	7.72e+03	1.18e+04	2.78e+02	3.54e+02	3.47e+02	1.11e+03	4.62e+02

multimodal, and composite functions. We considered only the dimension $n = 30$ and 50, since problems of lower dimension can hardly offer useful information for approaches such as GPALS. According to this test suite, the search space is fixed and equal to $[-100, 100]^n$ for all test problems. Also, the maximum number of function evaluations is equal to $T_{\max} = 10\,000n$. This is in contrast to the previous large-scale test suite where non-symmetrical search spaces and half the number of function evaluations are considered in problems of larger dimension. The optimal solutions for all test functions are known, and the suggested number of runs per problem are set to 51 as dictated by the test suite. The solution quality criterion of Eq. (16) is adopted also here. The SHADE [24] algorithm has been distinguished in the specific test suite and, hence, it was used as the baseline for comparisons in our study.

All implementations of our proposed approach were made in the C programming language. The results of the competitor algorithms were adopted either from published data or from sources referred in the test suites. The OpenMPI library[2] was used for the parallelization of the secondary populations in GPALS, according to a standard master–slave model. All experiments were conducted on Ubuntu Linux powered Intel® i7 machines with 8GB RAM, providing 8 CPUs each. Note that all function evaluations performed by GPALS, including the ones for the gradient estimation, were counted in the total number of evaluations in our experiments.

4.2. Results for high-dimensional problems

The GPALS-DE algorithm was applied following the exact settings of the base DE algorithm in the test suite [51]. Specifically, it was based on the exponential crossover operator with the DE/rand/1 mutation operator (see Eq. (4)), and population size $N = 60$. For each test problem, 25 independent experiments were conducted, recording the best solution that was detected within the available computational budget $T_{\max}$. It is a reasonable choice to initialize the parameters of the algorithms at the center of the corresponding search space [9,11]. Although we primarily adopted this approach, we also considered extreme initial parameter values in order to investigate the impact of their initialization on the results.

It shall be emphasized that the competitor algorithms were already tuned for the test suite, while the proposed GPALS-DE algorithm assumed no prior information. This is clearly an unfair setting for our approach, since it neglects the computationally demanding preprocessing phase of the competitor algorithms for their parameter tuning. Nevertheless, the investigation of the efficiency of GPALS even under such limited resources seemed appealing.

The DE scalar parameters F and CR are usually set in the range $(0, 1]$ [16]. Following this trend, we considered the parameter domain

$$G = (0, 1] \times (0, 1]. \tag{17}$$

GPALS-DE was initially tested using the central initial parameter pair $(F, CR) = (0.5, 0.5)$. The extreme initial pairs $(F, CR) = (0.2, 0.2)$ and $(F, CR) = (0.8, 0.8)$ were subsequently investigated. We henceforth denote these approaches as GPALS-DE$_{0.5}$, GPALS-DE$_{0.2}$, and GPALS-DE$_{0.8}$, respectively.

The average error and the standard deviation per test problem for the three GPALS-DE variants and the base algorithms of the test suite are reported in Tables 9–12 for all dimensions. In the 500-dimensional case, no results were obtained for GCMAES due to excessive computational time demanded by the provided source codes. In order to verify the significance of the observed performance differences and facilitate the extraction of sound conclusions, statistical significance tests were conducted. Thus, the

[2] http://www.open-mpi.org.

GPALS-DE approaches were statistically compared in terms of the achieved errors with each one of the base algorithms by using the Wilcoxon rank-sum test at confidence level 95%. Whenever GPALS-DE exhibited statistically significant difference and better median error value than another algorithm, it was awarded a *win*, while a *loss* was counted for the competitor. The opposite was considered in case of worst median value of GPALS-DE against a competitor. In case of no statistically significant difference, a *tie* was counted for both algorithms.

The obtained results are reported in Table 1 where wins, losses, and ties are denoted as "+ ", "−", and "=", respectively. The GPALS-DE approaches evidently outperformed the base algorithms in all dimensions with one exception. Specifically, GPALS-DE$_{0.2}$ and GPALS-DE$_{0.5}$ exhibited contiguous performance, while GPALS-DE$_{0.8}$ slightly deviated from this performance only for the lowest dimension $n = 50$. In that case, it was outperformed by the fine-tuned DE$_{\text{exp}}$ algorithm, which is reported to be the best-performing algorithm in the test suite repository. This shortcoming of GPALS-DE$_{0.8}$ can be attributed to the specific initial parameter pair, which reduces the convergence speed of DE in the specific test problems. Thus, GPALS needs additional effort to reach appropriate parameter values that lead the algorithm to optimal solutions efficiently. Nevertheless, it offers clear evidence that, even under defective parameter initialization and without any additional experimentation, the proposed GPALS approach can be highly beneficial for the algorithm.

The GPALS-DE$_{0.5}$ approach exhibited better or equivalent performance than the rest of the GPALS-DE variants. Thus, besides the base algorithms, GPALS-DE$_{0.5}$ was further compared against the rest of the algorithms of the test suite in terms of average error [37]. The results are reported in Tables 13 and 14. The data for the rest of the algorithms are reproduced from the original sources [51]. Figure 3 illustrates the number of problems (out of 19 problems) where GPALS-DE$_{0.5}$ was not outperformed by other algorithms. The data suggests that GPALS-DE was highly competitive also against non-DE-based algorithms for all dimensions.

Moreover, Wilcoxon rank-sum tests were conducted between GPALS-DE$_{0.5}$ and the previously proposed DEGPA approach [9]. The percentage of wins, losses, and ties between GPALS-DE$_{0.5}$ and DEGPA are graphically illustrated in Fig. 4. As we can see, GPALS-DE$_{0.5}$ achieved better or equivalent performance in almost all test problems, particularly for higher dimension. This observation verifies the additional benefits of using the approximate performance gradients with line search against the simpler grid-based approach of DEGPA.

Besides the comparisons with the competitor algorithms of the test suite, GPALS-DE$_{0.5}$ was compared with three state-of-the-art adaptive DE algorithms, namely GaDE [32], SHADE [24] and L-SHADE [25], on the high-dimensional test problems. For this purpose, source codes available on the test suites' repositories were used. The provided source code of GaDE was already tuned on the specific test suite, requiring no further modification. On the other hand, SHADE and L-SHADE are based on an adaptive scheme that employs a special mutation operator, which differs from the standard one used in our GPALS-DE approach. Also, their available codes came with proposed parameters that were tuned on the CEC 2013 test suite. Thus, in a head-to-head comparison with the GPALS-DE approaches, it would be difficult to identify whether the observed good or bad performance patterns can be attributed to the main parameter adaptation procedure of these algorithms or to their special mutation scheme, which is irrelevant to the parameter adaptation. For this reason, we decided to consider both the original forms of SHADE and L-SHADE, as well as a modified version of each one, denoted as mSHADE and mL-SHADE, respectively. In the modified versions, the same DE/rand/1 operator with exponential mutation that is used in GPALS-DE was adopted instead of the special mutation operator.

Moreover, the test suite demands populations of size $N = 60$. However, the proposed size for SHADE is $N = 100$. Thus, we decided to consider both population sizes in our comparisons, with the corresponding variant being denoted with a subscript, i.e., SHADE$_{60}$ and SHADE$_{100}$. Moreover, L-SHADE assumes linearly decreasing population size during its run, starting from a fixed value $N = 18n$. This scheme was also retained in our experiments.

The results of the Wilcoxon rank-sum tests of GPALS-DE$_{0.5}$ against GaDE, SHADE and the L-SHADE variants are reported in Table 2. In the 500-dimensional case, the provided source codes of L-SHADE failed to execute due to excessive memory demand. The results show that GPALS-DE$_{0.5}$ is highly competitive to all competitor algorithms, with evident superiority as dimension increases. It is interesting to notice that, despite the significant number of function evaluations spared by L-SHADE due to its decreasing population size, GPALS-DE$_{0.5}$ was superior. For completeness purpose, the corresponding average errors for SHADE and L-SHADE are reported in Tables 15 and 16.

In order to gain further insight on the impact of population size on the performance of GPALS-DE$_{0.5}$, additional experiments were conducted for $N = 40$ and 80 and compared to the base algorithms. The obtained results are reported in Table 3, where the corresponding population size is denoted as superscript. As we see, GPALS-DE$^{40}_{0.5}$ and GPALS-DE$^{60}_{0.5}$ exhibited similar or superior performance than the competitor algorithms, while GPALS-DE$^{80}_{0.5}$ slightly deviated from this performance especially for lower dimensions against the best base algorithm, namely DE$_{\text{exp}}$. Nevertheless, this effect is ameliorated as dimension increases. This is a consequence of using high population sizes in easier problems of lower dimension, which results in futile spending of additional function evaluations in the performance estimation phase. The same effect can also be observed against the SHADE variants in Table 4.

Finally, we recorded the running time of GPALS-DE$_{0.5}$ against SHADE. Running time is not a measure that can clearly offer sound conclusions regarding an algorithm's performance, since it depends on many external factors such as implementation quality, machine configuration, employed compilers, and machine workload at the moment of experimentation. Nevertheless, it is an indication that the nice performance of an algorithm is not achieved in excessively higher time than another algorithm. In our case, the available SHADE source code was serial, so we conducted 25 serial experiments on the same machine for each algorithm. The obtained times are illustrated in Fig. 5. Obviously, the proposed approach achieves lower execution time than the competitor SHADE algorithm in all dimensions. In addition, the running times were evaluated with Wilcoxon rank-sum tests, and wins, losses, and ties are reported in Table 5, verifying our observations.

4.3. Results for low-dimensional problems

GPALS has proved to be effective for high-dimensional problems, as we have shown so far. However, our study would be incomplete without reference to a mainstream test suite, even of lower problem dimension. For this reason, we investigated the performance of GPALS-DE$_{0.5}$ on the CEC 2013 real-parameter single-objective optimization benchmark suite [52]. Again, for comparison purposes, we considered SHADE and L-SHADE as the baseline algorithms. Both have been shown to be among the most efficient adaptive DE algorithms for the specific test suite. For simplicity reason, we will henceforth denote GPALS-DE$_{0.5}$ as GPALS-DE.

Our main goal remained the evaluation of GPALS as a general parameter adaptation method, using DE for demonstration purposes. Thus, in order to achieve a fair comparison between GPALS and the parameter adaptation scheme of SHADE, we also considered GPALS-DE with the special mutation operator of SHADE with population archive. Note that the corresponding modification was made for SHADE and L-SHADE previously in the high-dimensional test suite, resulting in mSHADE and mL-SHADE. Thus, we considered two GPALS-DE approaches, namely GPALS-DE that uses the original DE/rand/1 mutation operator with exponential crossover, and mGPALS-DE that uses SHADE's mutation operator with population archive. Whenever a mutant vector component of the GPALS approach was violating the boundary of the search space, we applied the same correction as in SHADE [24]. Moreover, besides the original SHADE and L-SHADE algorithm for the CEC 2013 test suite, we considered also the mSHADE and mL-SHADE defined in the previous section. This was motivated by the necessity to investigate whether the efficiency of the parameter adaptation scheme of SHADE and L-SHADE is intimately related to its special mutation operator or it can work equally good with the plain DE mutation operator as well. The population size for all our algorithms was equal to $N = 60$. All GPALS approaches assumed as initial parameter pair the central point of the search space, namely $(F, CR) = (0.5, 0.5)$. The rest of the settings were as dictated by the test suite.

Table 6 reports the results of the Wilcoxon rank-sum tests between GPALS-DE, SHADE, and L-SHADE. We can observe that SHADE and L-SHADE demonstrated clear superiority, dominating the mGPALS-DE algorithm in most of the test problems. However, this was not the case when the standard DE operator was used in SHADE, as revealed by the comparisons against GPALS-DE. Especially when dimension increases to 50, GPALS-DE shows superior performance compared to SHADE. On the other hand, L-SHADE achieves better performance compared to GPALS-DE, which obviously stems from the population size reduction, that spares computational budget. These results suggest that the parameter adaptation scheme of SHADE and L-SHADE, although very effective in the low-dimensional CEC 2013 test suite, seems to be intimately related to its special mutation operator.

Further comparisons of GPALS-DE were conducted with various other adaptive DE algorithms. Table 7 reports the results for four adaptive DE-based algorithms, which can be found in [53–57]. The average errors for the proposed and competitor algorithms are reported in Tables 17–19. We can clearly see that GPALS-DE exhibited competitive performance even for the low-dimensional problems.

Finally, running time analysis was conducted also for this test suite, following the same methodology as in the previous section. The corresponding results are illustrated in Fig. 6. Again,the GPALS-DE approach achieved better running times than SHADE. These findings are confirmed also in the statistical tests reported in Table 8.

5. Conclusions

The laborious task of parameter setting in metaheuristics typically requires significant computational resources, especially in demanding high-dimensional problems. Choosing appropriate parameters for an algorithm has been shown to be a problem-dependent task. The online parameter adaptation methods that are usually incorporated in specific algorithms suffer from two common weaknesses, namely overspecialization on the specific algorithms and inclusion of critical new parameters. This leaves space for further research developments.

Recently, a grid-based online parameter adaptation method was introduced as a simple and straightforward method for tuning an algorithm during its run. That method defines a rather general framework that can be used on various algorithms. The results obtained through demonstrations on state-of-the-art metaheuristics showed that the grid-based approach is competitive on both high-dimensional and low-dimensional problems despite its minimal number of (mostly operational) new parameters. However, it has two weak points that admit further improvement. The first one is the restricted number of search directions in the parameter domain, and the second one is the limited step size allowed on these directions. These weaknesses may result in rather slow adaptation of the algorithm's parameters.

The present work treats the weaknesses of the grid-based method by extending the parameter search procedure from the simplistic grid-based search to the more sophisticated gradient search with line search. While gradient estimations enhance the number of search directions, line search tackles the step size problem. All estimations of the underlying mathematical quantities are based on short-runs of the algorithm.

The proposed GPALS method was demonstrated on the state-of-the-art DE algorithm. The effectiveness of the corresponding GPALS-DE approach was shown on two test suites that include a variety of high-dimensional and low-dimensional test problems. The results suggested that GPALS can disburden the user from the need of finding proper parameter settings, while it can significantly improve the algorithm's performance. Very competitive performance was observed also against other already fine-tuned algorithms from the relevant literature, despite the fact that GPALS neither underwent fine-tuning nor was assigned the additional computational budget spent by the rest of the algorithms for their optimal tuning. Moreover, compared to other methods, GPALS has shown satisfactory running time performance on the specific test problems.

These promising results cultivate the ground for further inquiry, including benchmarking with different problem types and algorithms, as well as alternative parameter search methods.

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive comments and suggestions.

References

[1] G. Zäpfel, R. Braune, M. Bögl, Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics, Springer, Berlin, 2010.
[2] A.E. Eiben, S.K. Smit, Evolutionary algorithm parameters and methods to tune them, in: Y. Hamadi, E. Monfroy, F. Saubion (Eds.), Autonomous Search, Springer, Berlin, Heidelberg, 2011, pp. 15–36 (Chapter 2).
[3] G. Karafotias, M. Hoogendoorn, A.E. Eiben, Parameter control in evolutionary algorithms: Trends and challenges, IEEE Trans. Evol. Comput. 19 (2) (2015) 167–187.
[4] S.W. Leung, S.Y. Yuen, C.K. Chow, Parameter control system of evolutionary algorithm that is aided by the entire search history, Appl. Soft Comput. 12 (9) (2012) 3063–3078.
[5] T. Bartz-Beielstein, Experimental Research in Evolutionary Computation, Springer-Verlag, Berlin, 2006.
[6] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-race and iterated f-race: An overview, in: T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin, Heidelberg, 2010, pp. 311–336.
[7] F. Hutter, H.H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: A.C. Coello Coello (Ed.), Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy. Selected Papers, Springer, Berlin, Heidelberg, 2011, pp. 507–523.
[8] H.H. Hoos, Automated algorithm configuration and parameter tuning, in: Y. Hamadi, E. Monfroy, F. Saubion (Eds.), Autonomous Search, Springer, Berlin, Heidelberg, 2011, pp. 37–72 (Chapter 3).
[9] V.A. Tatsis, K.E. Parsopoulos, Differential evolution with grid-based parameter adaptation, Soft Comput. 21 (8) (2017) 2105–2127.
[10] V.A. Tatsis, K.E. Parsopoulos, Grid-based parameter adaptation in particle swarm optimization, in: Proceedings of the 12th Metaheuristics International Conference, MIC 2017, 2017.
[11] V.A. Tatsis, K.E. Parsopoulos, Grid search for operator and parameter control in differential evolution, in: Proceedings of the 9th Hellenic Conference on Artificial Intelligence, SETN '16, ACM, New York, NY, USA, 2016, pp. 7:1–7:9.

[12] V.A. Tatsis, K.E. Parsopoulos, Experimental assessment of differential evolution with grid-based parameter adaptation, Int. J. Artif. Intell. Tools 27 (2018).
[13] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, Verlag, Berlin, 2005.
[14] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[15] R. Gämperle, S.D. Müller, P. Koumoutsakos, A parameter study for differential evolution, in: WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, Press, 2002, pp. 293–298.
[16] S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[17] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Comput. 14 (2010) 1187–1207.
[18] S.Z. Zhao, P.N. Suganthan, S. Das, Self-adaptive differential evolution with multi-trajectory search for large-scale optimization, Soft Comput. 15 (11) (2011) 2175–2185.
[19] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[20] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (2) (1999) 124–141.
[21] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[22] D. Zaharie, D. Petcu, Parallel implementation of multi-population differential evolution, Concurr. Inf. Process. Comput. (2005) 223–232.
[23] J. Zhang, A.C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Trans, Evol. Comput. (CEC) 13 (2009) 945–958.
[24] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 71–78.
[25] R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: IEEE Congress on Evolutionary Computation, 2014.
[26] R. Poláková, J. Tvrdík, P. Bujok, Controlled restart in differential evolution applied to CEC2014 benchmark functions, in: IEEE Congress on Evolutionary Computation, CEC, 2014.
[27] J. Tvrdík, Competitive differential evolution, in: 12th International Coference on Soft Computing, 2006.
[28] J. Tvrdík, R. Poláková, Competitive differential evolution applied to CEC problems, in: 2013 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2013, p. 1651 1657.
[29] J. Brest, B. Bošković, A. Zamuda, Self-adaptive differential evolution algorithm with a small and varying population size, in: WCCI 2012 IEEE World Congress on Computational Intelligence, 2012.
[30] A.P. Piotrowski, Adaptive memetic differential evolution with global and local neighborhood-based mutation operators, Inform. Sci. 241 (2013) 164–194.
[31] J. Brest, M.S. Maucec, Self-adaptive differential evolution algorithm using population size reduction and three strategies, Soft Comput. 15 (2011) 2157–2174.
[32] Z. Yang, K. Tang, X. Yao, Scalability of generalized adaptive differential evolution for large-scale continuous optimization, Soft Comput. 15 (11) (2011) 2141–2155.
[33] C. Segura, A.C. Coello Coello, E. Segredo, C. León, On the adaptation of the mutation scale factor in differential evolution, Optim. Lett. 9 (1) (2015) 189–198.
[34] D. Zaharie, Influence of crossover on the behavior of differential evolution algorithms, Appl. Soft Comput. 9 (3) (2009) 1126–1138.
[35] V.A. Tatsis, K.E. Parsopoulos, On the Sensitivity of Grid-Based Parameter Adaptation Method, in: 7th International Conference on Metaheuristics and Nature Inspired Computing (META 2018), 2018, accepted for publication.
[36] D.A.G. Vieira, A.C. Lisboa, Line search methods with guaranteed asymptotical convergence to an improving local optimum of multimodal functions, European J. Oper. Res. 235 (1) (2014) 38–46.
[37] M. Lozano, F. Herrera, D. Molina, Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, Soft Comput. 15 (2011) 2085–2087.
[38] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval-schemata, Found. Genet. Algorithms 2 (1993) 187–202.
[39] A. Auger, N. Hansen, A restart cma evolution strategy with increasing population size, in: Proc. of the 2005 IEEE Congress on Evolutionary Computation, CEC, 2005, pp. 769–1776.
[40] M. Weber, F. Neri, V. Tirronen, Shuffle or update parallel differential evolution for large scale optimization, Soft Comput. 15 (2011) 2089–2107.
[41] C. García-Martínez, F.J. Rodríguez, M. Lozano, Role differentiation and malleable mating for differential evolution: An analysis on large scale optimisation, Soft Comput. 15 (2011) 2109–2126.
[42] Z. Yang, K. Tang, X. Yao, Scalability of generalized adaptive differential evolution for large-scale continuous optimization, Soft Comput. 15 (2011) 2141–2155.
[43] A. LaTorre, S. Muelas, J. Peña, A MOS-based dynamic memetic differential evolution algorithm for continuous optimization a scalability test, Soft Comput. 15 (2011) 2187–2199.
[44] D. Molina, M. Lozano, A.M. Sánchez, F. Herrera, Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-Chains, Soft Comput. 15 (2011) 2201–2220.
[45] J. García-Nieto, E. Alba, Restart particle swarm optimization with velocity modulation: A scalability test, Soft Comput. 15 (2011) 2221–2232.
[46] M.A. Montes de Oca, D. Aydin, T. Stützle, An incremental particle swarm for large-scale optimization problems: An example of tuning-in-the-loop (re)design of optimization algorithms, Soft Comput. 15 (2011) 2233–2255.
[47] A. Duarte, R. Martí, F. Gortazar, Path relinking for large scale global optimization, Soft Comput. 15 (2011) 2257–2273.
[48] V. Gardeux, R. Chelouah, P. Siarry, F. Glover, EM323: A line search based algorithm for solving high-dimensional continuous non-linear optimization problems, Soft Comput. 15 (2011) 2275–2285.
[49] A. Neumaier, H. Fendl, H. Schilly, T. Leitner, VXQR: Derivative-free unconstrained optimization based on qr factorizations, Soft Comput. 15 (2011) 2287–2298.
[50] H. Wang, Z. Wu, S. Rahnamayan, Role differentiation and malleable mating for differential evolution: An analysis on large scale optimisation, Soft Comput. 15 (2011) 2127–2140.
[51] M. Lozano, F. Herrera, D. Molina, Evolutionary algorithms and other metaheuristics for continuous optimization problems, 2010, URL http://sci2s.ugr.es/eamhco/.
[52] J.J. Liang, B.Y. Qu, P.N. Suganthan, A.G. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the Cec 2013 Special Session on Real-Parameter Optimization, Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore.
[53] Complementary material: Special session & competition on real-parameter single objective optimization at cec-2013, http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm.
[54] I. Poikolainen, F. Neri, Differential evolution with concurrent fitness based local search, in: IEEE Congress on Evolutionary Computation, CEC, IEEE, 2013.
[55] J. Brest, B. Bošković, A. Zamuda, I. Fister, E. Mezura-Montes, Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies, in: IEEE Congress on Evolutionary Computation, CEC, IEEE, 2013, pp. 377–383.
[56] S.M.M. Elsayed, R.A. Sarker, T. Ray, Differential evolution with automatic parameter configuration for solving the cec2013 competition on real-parameter optimization, in: 2013 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2013, pp. 1932–1937.
[57] L. dos Santos Coelho, H.V.H. Ayala, R.Z. Freire, Population's variance-based adaptive differential evolution for real parameter optimization, in: 2013 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2013, pp. 1672–1677.