Improving LSHADE by means of a pre-screening mechanism

Mateusz Zaborski*
M.Zaborski@mini.pw.edu.pl
Warsaw University of Technology
Warsaw, Poland

Jacek Mańdziuk*
mandziuk@mini.pw.edu.pl
Warsaw University of Technology
Warsaw, Poland

ABSTRACT

Evolutionary algorithms have proven to be highly effective in continuous optimization, especially when numerous fitness function evaluations (FFEs) are possible. In certain cases, however, an expensive optimization approach (i.e. with relatively low number of FFEs) must be taken, and such a setting is considered in this work. The paper introduces an extension to the well-known LSHADE algorithm in the form of a pre-screening mechanism (psLSHADE). The proposed pre-screening relies on the three following components: a specific initial sampling procedure, an archive of samples, and a global linear meta-model of a fitness function that consists of 6 independent transformations of variables. The pre-screening mechanism preliminary assesses the trial vectors and designates the best one of them for further evaluation with the fitness function. The performance of psLSHADE is evaluated using the CEC2021 benchmark in an expensive scenario with an optimization budget of $10^2 - 10^4$ FFEs per dimension. We compare psLSHADE with the baseline LSHADE method and the MadDE algorithm. The results indicate that with restricted optimization budgets psLSHADE visibly outperforms both competitive algorithms. In addition, the use of the pre-screening mechanism results in faster population convergence of psLSHADE compared to LSHADE.

CCS CONCEPTS

• **Theory of computation** → **Continuous optimization**; *Evolutionary algorithms*.

KEYWORDS

Surrogate model, LSHADE, Meta-model

1 INTRODUCTION

Evolutionary algorithms (EAs) are applicable to various global optimization problems [28]. In this work, we focus on single-objective continuous optimization, in which EAs have proven to be particularly useful [6].

The widely known Differential Evolution (DE) algorithm [23] was initially designed as a relatively simple but efficient heuristic. Despite unsophisticated structure, it became the precursor of various more advanced and specialized algorithms. For instance, Adaptive Differential Evolution with Optional External Archive [32] (JADE) introduced parameter adaptation. Then, Success-History Based Parameter Adaptation for Differential Evolution [25] (SHADE) presented a more efficient parameter adaptation mechanism utilizing an external memory. Afterward, the LSHADE [27], that extends SHADE with Linear Population Size Reduction (LPSR), has demonstrated to ameliorate the performance even further. Also, a population restart mechanism appeared to be

beneficial in certain experimental settings [26]. Moreover, several other algorithms enhanced certain DE mechanisms, e.g. AGSK [18], LSHADE_cnEpSin [3], j2020 [8], IMODE [22], jDE [7], or SaDE [21].

Finally, the recently proposed MadDE algorithm [5], presented at the *CEC2021 Special Session and Competition on Single Objective Bound Constrained Optimization*, turned out to be superior to both LSHADE and LSHADE_cnEpSin.

Besides DE-related methods there is also another family of effective global optimization algorithms that refer to the Covariance Matrix Adaptation (CMA) method. These include CMA-ES [11], IPOP-CMA-ES [1], or KL-BIPOP-CMA-ES [30], PSA-CMA-ES [19].

The vast majority of the above-mentioned DE- and CMA-based algorithms are generally effective and the differences between them often boil down to the details of the experiment scenario, e.g. the utilized benchmarks or method parameterization.

Predominantly, the performance of an optimization algorithm is measured with respect to a number of (true) fitness function evaluations (FFEs). In many settings, numerous FFEs are assumed while evaluating the algorithm's performance (e.g. the CEC2021 benchmark assumes $2 \cdot 10^5$ FFEs for 10D problems). In contrast, expensive optimization setups are limited to a relatively small number of evaluations (e.g. CEC2015 benchmark for Computationally Expensive Numerical Optimization assumes $5 \cdot 10^2$ FFEs for 10D problems). Complex surrogate models can be highly supportive in such costly optimization, but their high computational complexity makes them scale poorly (e.g. Efficient Global Optimization [14], Kriging [9], DTS-CMA-ES [4]).

Surrogate-assisted optimization has received considerable attention in recent years [13]. The LS-CMA-ES [2] utilizes quadratic approximation of the fitness function. Also, polynomials can be utilized as local meta-models [16]. M-GAPSO [31] is a hybrid of PSO, DE, and local meta-models: linear and quadratic. SHADE-LM [20] is a new hybrid of SHADE and efficient, in terms of complexity, local meta-models. The lq-CMA-ES [10] incorporates a quadratic meta-model to replace fitness evaluations with surrogate estimates to reduce unnecessary FFEs. The replacement depends on the rank correlation measure (Kendall's τ [15]).

This work presents and evaluates psLSHADE: the LSHADE algorithm extended with a pre-screening mechanism. The pre-screening mechanism includes a modified initial sampling procedure, an archive of samples, and a global meta-model of a fitness function. The meta-model is a linear combination of 6 components (independent variable transformations). It ranks trial vectors and designates the best one of them for further FFE.

The remainder of the paper is structured as follows. Section 2 describes the baseline LSHADE. Section 3 introduces the principles of psLSHADE, in particular the pre-screening mechanism. Section 4 presents experimental evaluation of psLSHADE using the CEC2021

---

*Both authors contributed equally to this research.

benchmark in expensive scenarios. Section 5 analyses the meta-model performance. Finally, conclusions and directions for future work are briefly described in Section 6.

2 LSHADE ALGORITHM

In this section the LSHADE algorithm is introduced based on its original description [27]. LSHADE is a single-objective optimization metaheuristic that extends SHADE [25] with Linear Population Size Reduction. LSHADE includes parameter adaptation, memory, and external archive, similar to the underlying SHADE. LSHADE is a population-based iterative algorithm where each individual i represents D-dimensional point $\boldsymbol{x}_i^g = [x_{i,1}^g, \ldots, x_{i,D}^g]$ in assumed solution space, where g denotes an iteration number. In each iteration g, the population P^g consists of N^g individuals $[\boldsymbol{x}_1^g, \ldots, \boldsymbol{x}_{N^g}^g]$ independently subjected to subsequent phases: mutation, crossover, and selection.

The mutation phase is responsible for random generation of a mutated vector $\boldsymbol{v}_i^g$ based on randomly generated scaling factor parameter F_i^g and three randomly chosen individuals $(\boldsymbol{x}_{pbest_i}^g, \boldsymbol{x}_{r1_i}^g,$ and $\boldsymbol{x}_{r2_i}^g)$. The method of determining F_i^g is described in section 2.2. Index $pbest_i$ indicates an individual randomly selected from the set of the currently best $N^g \cdot p$ individuals, where p is a parameter. $r1_i \in \{1, \ldots, N^g\}$ denotes an individual from population P^g and $r2_i \in \{1, \ldots, N^g + |A|\}, (r2_i \neq r1_i)$ an individual from the union of the population P^g and an external archive A. The rules of constructing the external archive A are described in more detail in section 2.1. The final form of the mutated vector $\boldsymbol{v}_i^g$ can be described as follows:

$$\boldsymbol{v}_i^g = \boldsymbol{x}_i^g + F_i^g(\boldsymbol{x}_{pbest_i}^g - \boldsymbol{x}_i^g) + F_i^g(\boldsymbol{x}_{r1_i}^g - \boldsymbol{x}_{r2_i}^g) \quad (1)$$

Next, the parent vector $\boldsymbol{x}_i^g$ is crossed with the mutated vector $\boldsymbol{v}_i^g$, leading to a trial vector $\boldsymbol{u}_i^g = [u_{i,1}^g, \ldots, u_{i,D}^g]$, in which each element $u_{i,d}^g, d = 1, \ldots, D$ takes either the value $x_{i,d}^g$ (with probability CR_i^g) or $v_{i,d}^g$, otherwise. The way the CR_i^g is determined is described in section 2.2. Furthermore, one randomly chosen element $u_{i,d_{rand}}^g$ is crossed regardless of the probabilistic outcome, i.e.

$$u_{i,d}^g = \begin{cases} v_{i,d}^g, & \text{if } rand(0,1) \leq CR_i^g \text{ or } d = d_i^{rand} \\ x_{i,d}^g, & \text{otherwise} \end{cases} \quad (2)$$

where $rand(0,1)$ denotes a uniformly selected random number from $[0,1)$ and $d_i^{rand} \in \{1, \ldots, D\}$ is a randomly selected index. Both $rand(0,1)$ and d_i^{rand} are generated independently for each i.

Finally, the trial vector $\boldsymbol{u}_i^g$ is subject to the selection phase. Technically, it is evaluated using fitness function f, and its value $f(\boldsymbol{u}_i^g)$ is compared with the value $f(\boldsymbol{x}_i^g)$ of the parent vector $\boldsymbol{x}_i^g$. The better vector is promoted to the next generation population $(g + 1)$. The selection phase can be formally expressed as follows:

$$\boldsymbol{x}_i^{g+1} = \begin{cases} \boldsymbol{u}_i^g, & \text{if } f(\boldsymbol{u}_i^g) < f(\boldsymbol{x}_i^g) \\ \boldsymbol{x}_i^g, & \text{otherwise} \end{cases} \quad (3)$$

LSHADE utilizes a Linear Population Size Reduction (LPSR) mechanism, which results in changing in time population size N^g during an optimization run. The LPSR mechanism requires N_{init} and N_{min} parameters, indicating the initial and minimal (terminal) population sizes, resp. In each iteration g, after the selection

phase, the population size N^{g+1} for the next iteration is obtained as follows:

$$N^{g+1} = round\left(\left(\frac{N_{min} - N_{init}}{MAX\_NFE}\right) \cdot NFE + N_{init}\right) \quad (4)$$

where MAX_NFE indicates the optimization budget and NFE is the number of fitness function evaluations made so-far. If the population size is reduced, the worst individuals, in the sense of fitness function value, are removed.

2.1 External archive

LSHADE utilizes an external archive A that extends the current population with the individuals (parent vectors) that have been replaced with better offsprings (trial vectors) in the selection phase. The archive is of size $|A| = a \cdot N^g$, where a is a parameter. If the archive is full, a randomly selected element is removed to allow an insertion of a new one in its place. Likewise, when the archive size is shrunk due to population size reduction, randomly selected elements are removed.

2.2 Parameter adaptation

The scaling factor F_i^g utilized in the mutation phase (1), and crossover rate CR_i^g utilized in the crossover phase (2) are designated using the memory. The purpose of the memory is to store historical values of F_i^g and CR_i^g that succeeded in the selection phase, i.e. the trial vector $\boldsymbol{u}_i^g$ was better than the parent vector $\boldsymbol{x}_i^g$. The memory consists of H pairs of scaling factor entries $M_{F,k}^g$ and crossover rate entries $M_{CR,k}^g$, $k = 1, \ldots, H$. In each iteration, after the selection phase, all successful values of F_i^g and CR_i^g are recorded in sets S_F and S_{CR}, resp. Then, both sets are transformed using a weighted Lehmer mean so that two values, $mean_{W_L}(S_F)$ and $mean_{W_L}(S_{CR})$, are obtained. The following equation describes the weighted Lehmer transformation:

$$mean_{W_L}(S) = \frac{\sum_{k=1}^{|S|} w_k S_k^2}{\sum_{k=1}^{|S|} w_k S_k}, \qquad w_k = \frac{\Delta f_k}{\sum_{l=1}^{|S|} \Delta f_l} \quad (5)$$

where the improvement Δf_p, $p \in \{k, l\}$ is a difference between the fitness function value of the parent vector $(f(\boldsymbol{x}_i^g))$ and the fitness function value of the trial vector $(f(\boldsymbol{u}_i^g))$.

The memory entries $M_{F,k}^g$ and $M_{CR,k}^g$ are updated sequentially with values $mean_{W_L}(S_F)$ and $mean_{W_L}(S_{CR})$, from $k = 1$ to $k = H$. After updating the last entry $(k = H)$, the updating procedure starts again from $k = 1$.

In addition, if all CR_i^g values in set S_{CR} are equal to 0, the memory updating procedure permanently marks the entry $M_{CR,k}^g$ with the terminal value $\perp$ (instead of $mean_{W_L}(S_{CR})$).

At the beginning of each iteration, a random index r_i of memory entry is determined, independently for each individual. The values F_i^g and CR_i^g are generated randomly using Cauchy distribution and Normal distribution, resp. with M_{F,r_i}^g and M_{CR,r_i}^g (taken from the memory) being the parameters of these distributions. In summary, F_i^g and CR_i^g are generated in the following way:

$$F_i^g = rand_{Cauchy}(M_{F,r_i}^g, 0.1) \quad (6)$$

$$CR_i^g = \begin{cases} 0 & \text{if } M_{CR,r_i}^g = \perp \\ rand_{Normal}(M_{CR,r_i}^g, 0.1) & \text{otherwise} \end{cases} \quad (7)$$

If $F_i^g > 1$, F is truncated to 1, and if $F_i^g \leq 0$, a random generation is repeated. In case the generated CR_i^g value is outside $[0, 1]$, it is truncated to the limit value of 0 or 1, resp.

3 PROPOSED PRE-SCREENING MECHANISM

The core of this paper is the proposal of incorporation of the pre-screening mechanism into LSHADE, which results in the psLSHADE method. The mechanism assumes that each individual i will generate more than one trial vector, but only the most promising one, according to the meta-model evaluation, will be evaluated using the (true) fitness function. The proposed pre-screening mechanism includes a modified initial sampling procedure, an archive of samples that stores already evaluated samples, and a global meta-model re-estimated in each iteration. Moreover, all three phases (mutation, crossover and selection) of the underlying LSHADE algorithm have been modified to generate an increased number of trial vectors.

A pseudocode of the psLSHADE algorithm is presented in Algorithm 1. A source code is available in the public repository [1].

3.1 Initial population

The initial population P^g in psLSHADE is generated using Latin Hypercube Sampling [12] to ensure better coverage of the search space compared to purely uniform random sampling.

3.2 Genetic operators

The modified mutation phase generates N_s mutated vectors $\boldsymbol{v}_i^{g,j}$, $j = 1, \ldots, N_s$, per individual i. All randomly generated values utilized in LSHADE are now designated independently for each of N_s mutated vectors $\boldsymbol{v}_i^{g,j}$. The generation principles, however, remain unchanged, including M_{F,r_i}^g parameter in $F_i^{g,j}$ distribution (9), which is constant in all generation procedures related to individual i in iteration g. In summary, the mutated vector $\boldsymbol{v}_i^{g,j}$ is obtained as follows:

$$\boldsymbol{v}_i^{g,j} = \boldsymbol{x}_i^g + F_i^{g,j}(\boldsymbol{x}_{pbest_i}^{g,j} - \boldsymbol{x}_i^g) + F_i^{g,j}(\boldsymbol{x}_{r1_i}^{g,j} - \boldsymbol{x}_{r2_i}^{g,j}) \quad (8)$$

where $F_i^{g,j}$ is generated using the Cauchy distribution:

$$F_i^{g,j} = rand_{Cauchy}(M_{F,r_i}^g, 0.1) \quad (9)$$

Then, for each individual i, N_s trial vectors $\boldsymbol{u}_i^{g,j} = [u_{i,1}^{g,j}, \ldots, u_{i,D}^{g,j}]$, $j = 1, \ldots, N_s$, are designated using the same CR_i^g and d_i^{rand} values. Also, the value $rand(0, 1)$ for individual i is generated once for all N_s trial vectors. Consequently, each element $u_{i,1}^{g,j}$ is described by the following equation:

$$u_{i,d}^{g,j} = \begin{cases} v_{i,d}^{g,j}, & \text{if } rand(0,1) \leq CR_i^g \text{ or } d = d_i^{rand} \\ x_{i,d}^g, & \text{otherwise} \end{cases} \quad (10)$$

where $\forall_{j \in \{1,\ldots,N_s\}} CR_i^g = const.$, $d_{rand} = const.$, and $rand(0, 1) = const.$

After the crossover phase, the meta-model's coefficients are estimated. The principles of a global meta-model are described in detail in section 3.5. Then, all $N^g \cdot N_s$ trial vectors $\boldsymbol{u}_i^{g,j}$ are evaluated using the meta-model to obtain $N^g \cdot N_s$ surrogate function values $f^{surr}(\boldsymbol{u}_i^{g,j})$. Afterward, the best trial vector $\boldsymbol{u}_i^{g,best}$ is chosen, independently for each individual i, based on the $f^{surr}(\boldsymbol{u}_i^{g,j})$ values. Finally, N^g best trial vectors are evaluated using the fitness function, and each of them undergoes the selection phase:

$$\boldsymbol{x}_i^{g+1} = \begin{cases} \boldsymbol{u}_i^{g,best}, & \text{if } f(\boldsymbol{u}_i^{g,best}) < f(\boldsymbol{x}_i^g) \\ \boldsymbol{x}_i^g, & \text{otherwise} \end{cases} \quad (11)$$

3.3 Archive

The archive of samples stores at most N_a pairs of already evaluated trial vectors $\boldsymbol{u}_i^{g,best}$ (or vectors $\boldsymbol{x}_i^0$ generated by initial sampling) and their respective function values $f(\boldsymbol{u}_i^{g,best})$ ($f(\boldsymbol{x}_i^0)$, resp.). A new pair $(\boldsymbol{u}_i^{g,best}, f(\boldsymbol{u}_i^{g,best}))$ is unconditionally inserted if the archive contains less than N_a elements. If the archive is full, the worst pair in terms of the fitness function value is removed from the archive, and a new pair is inserted. Removing the worst pair and insertion of a new one is not executed if the worst pair from the archive is better than the new pair. In order to ensure numerically correct estimation of meta-model parameters, an additional anti-duplicate condition is met. Removing and inserting is not executed if the archive already contains a pair that is similar to a new pair. The pairs are treated as similar when their trial vectors are equal or fitness function values are equal. The accepted numerical tolerance of equality is 10^{-12}.

3.4 Parameter adaptation

The parameter adaptation procedure remains unchanged compared to LSHADE. Utilization of pre-screening is transparent for the memory. All successful pairs $(F_i^{g,best}, CR_i^{g,best})$ that replaced (F_i^g, CR_i^g) are collected in sets S_F and S_{CR}, resp. The remaining steps are analogous to those employed in LSHADE.

Algorithm 1 Pre-screening LSHADE high-level pseudocode

1: Set all parameters N_{init}, N_{min}, M_F, M_{CR}, p, a, H, N_a, N_s (Section 4.2)
2: Initialize $M_{F,k}^0$ and $M_{CR,k}^0$ memory entries with default values of M_F and M_{CR}
3: Initialize $P^0 = [\boldsymbol{x}_1^0, \ldots, \boldsymbol{x}_N^0]$ with $N = N_{init}$ using Latin Hypercube Sampling
4: Update sequentially the archive of samples with all individuals from P^0
5: $g = 1$
6: **while** evaluation budget left **do**
7:     Generate $N^g \cdot N_s$ mutated vectors $\boldsymbol{v}_i^{g,j}$ using eq. (8)
8:     Generate $N^g \cdot N_s$ trial vectors $\boldsymbol{u}_i^{g,j}$ using eq. (10)
9:     Estimate meta-model parameters (Table 1)
10:    Calculate $N^g \cdot N_s$ surrogate values $f^{surr}(\boldsymbol{u}_i^{g,j})$
11:    For each individual i designate the best trial vector $\boldsymbol{u}_i^{g,best}$
12:    **for** i = 1 to N^g **do**
13:       Do selection of $\boldsymbol{u}_i^{g,best}$ using eq. (11)
14:       Add $\boldsymbol{u}_i^{g,best}$ and $f(\boldsymbol{u}_i^{g,best})$ to the archive of samples
15:    **end for**
16:    Update memory with $M_{F,k}^g$ and $M_{CR,k}^g$ using eq. (5)
17:    Set new population size N^{g+1} using eq. (4)
18:    $g = g + 1$
19: **end while**

---

[1]https://bitbucket.org/mateuszzaborski/pslshade/

3.5 Meta-model characteristic

The meta-model utilized in the pre-screening mechanism is a global linear model that consists of 6 independent transformations of variables: constant, linear, quadratic, modeling interactions, inverse linear, and inverse quadratic. All transformations, including their form and the number of degrees of freedom, are described in Table 1. The final form of the meta-model is a linear combination of 6 transformations. The meta-model coefficients are estimated using Ordinary Least Squares [29]. The meta-model is constructed when the archive of samples contains at least df_{mm} samples. Thus, the archive size should equal at least df_{mm}.

Table 1: A description of transformations and the final meta-model (mm.) The estimated coefficients applied to each variable are omitted for the sake of readability.

Name	Form	DoF
Constant	$X_c = [1]$	$df_c = 1$
Linear	$X_l = [x_1, \ldots, x_D]$	$df_l = D$
Quadratic	$X_q = [x_1^2, \ldots, x_D^2]$	$df_q = D$
Interactions	$X_i = [x_1 x_2, \ldots, x_{D-1} x_D]$	$df_i = \frac{D(D-1)}{2}$
Inv. linear	$X_{il} = [\frac{1}{x_1}, \ldots, \frac{1}{x_D}]$	$df_{il} = D$
Inv. quad.	$X_{iq} = [\frac{1}{x_1^2}, \ldots, \frac{1}{x_D^2}]$	$df_{iq} = D$
Final mm.	$[X_c + X_l + X_q + X_i + X_{il} + X_{iq}]$	$df_{mm} = \frac{D^2+7D}{2} + 1$

4 EXPERIMENTAL EVALUATION

We evaluated psSHADE on the recent popular *CEC2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization benchmark suite*, described in technical report [17], henceforth referred to as CEC2021. CEC2021 consists of 10 bound-constraint functions belonging to the following four categories: unimodal functions (F_1), basic functions (F_2 - F_4), hybrid functions (F_5 - F_7), and composition functions (F_8 - F_{10}). Each function is defined for both 10 and 20 dimensions.

In addition, three function transformations are proposed in CEC2021: bias (B), shift (S), and rotation (R). Besides the baseline case without transformations applied, the following four combinations of transformations (applied to each function) are tested: S, B+S, S+R, and B+S+R.

By default CEC2021 assumes the following optimization budget: $2 \cdot 10^5 \cdot D$ FFEs for 10D problems and $10^6 \cdot D$ FFEs for 20D problems. Please refer to [17] for explicit function definitions. Since surrogate-assisted algorithms, such as psLSHADE, are generally designed for expensive scenarios, i.e. restricted optimization budgets, we present the results based on the smaller budgets: $10^2 \cdot D$, $10^3 \cdot D$, and $10^4 \cdot D$. Employing three different budgets helps to determine when incorporating the pre-screening mechanism is the most effective.

Each experiment is repeated 30 times. A search range is $[-100, 100]^D$ and is the same for all functions, dimensions, and transformations. In summary, for each optimization budget the entire evaluation process included $10 \times 5 \times 30 \times 2 = 3000$ (functions $\times$ transformations $\times$ repetitions $\times$ dimensions) independent runs.

The performance of the two competing algorithms (baseline LSHADE and MadDE) was examined under the same conditions and in the same manner. The implementations of LSHADE and MadDE were taken from [24]. In addition, we made sure that psLSHADE implementation without pre-screening exactly matches the implementation of LSHADE.

4.1 Performance metrics

The evaluation metrics are adopted from the aforementioned technical report [17]. The final score is calculated in the few following steps. First, the *SNE* value is calculated:

$$SNE = 0.5 \sum_{m=1}^{5} \sum_{i=1}^{10} ne_{i,m}^{10D} + 0.5 \sum_{m=1}^{5} \sum_{i=1}^{10} ne_{i,m}^{20D} \quad (12)$$

where $ne_{i,m}^{dim}$ is a normalized error value (13) for function F_i, transformation m, and dimension dim (for the sake of clarity we omit the indices):

$$ne = \frac{f(\boldsymbol{x}_{best}) - f(\boldsymbol{x}^*)}{f(\boldsymbol{x}_{best})_{max} - f(\boldsymbol{x}^*)} \quad (13)$$

where $f(\boldsymbol{x}_{best})$ is the algorithm's best result out of 30 trials (repetitions), $f(\boldsymbol{x}^*)$ is the function's optimal value, and $f(\boldsymbol{x}_{best})_{max}$ is the largest (the worst) $f(\boldsymbol{x}_{best})$ among all algorithms.

The Score1 value is calculated as follows:

$$Score1 = \left(1 - \frac{SNE - SNE_{min}}{SNE}\right) \times 50 \quad (14)$$

where SNE_{min} is the minimal SNE among all algorithms. Then the SR value is obtained:

$$SR = 0.5 \sum_{m=1}^{5} \sum_{i=1}^{10} rank_{i,m}^{10D} + 0.5 \sum_{m=1}^{5} \sum_{i=1}^{10} rank_{i,m}^{20D} \quad (15)$$

where $rank_{i,m}^{dim}$ is the algorithm's rank among all algorithms for a given function i, transformation m, and dimension dim, using the mean error value from all trials.

The Score2 value is computed in the following way:

$$Score2 = \left(1 - \frac{SR - SR_{min}}{SR}\right) \times 50 \quad (16)$$

where SR_{min} is the minimal sum of $rank_{i,m}$ among the compared algorithms.

The final *Score* measure is the sum of (14) and (16):

$$Score = Score1 + Score2 \quad (17)$$

For auxiliary measures *SNE* and *SR*, lower values indicate better performance, whereas for *Score1*, *Score2*, and *Score*, higher values correspond to better results. The maximum value of *Score* is 100.

4.2 Tuning of psLSHADE parameters

4.2.1 Shared psLSHADE/LSHADE parameters. For the sake of fair comparison, all psLSHADE parameters shared with LSHADE have the same values in both algorithms, and follow the parameterization used previously for benchmarking LSHADE on CEC2021 [24]. The following parameters are shared by both algorithms: initial population size $N_{init} = 18 \cdot D$, final population size $N_{min} = 4$, initial value of $M_F = 0.5$, initial value of $M_{CR} = 0.5$, best rate $p = 0.11$, archive rate $a = 1.4$, memory size $H = 5$.

4.2.2 psLSHADE pre-screening component parameters. The two additional psLSHADE parameters are N_a and N_s. The archive size (N_a) was selected arbitrary (with no tuning) following the lq-CMA-ES [10] parameterization. In [10] the archive is of the size $max(\lambda, 2 \cdot df_{max})$, where λ is the population size and df_{max} is the number of degrees of freedom of the most complex meta-model. Due to population size reduction in psLSHADE, we conditioned the archive size on the number of degrees of freedom only.

For the number of trial vectors per individual (N_s) the tuning tests were performed with $10^3 \cdot D$ budget and the following values: $N_s = \{2, 5, 10, 20\}$. psLSHADE results obtained for all four N_s choices are presented in Table 2. The best performing value was $N_s = 5$. $N_s = 2$ was a slightly weaker choice, indicating that increasing the number of trial vectors up to a certain point ($N_s = 5$ in this case) improves performance, rendering the use of pre-screening beneficial. Both $N_s = 10$ and $N_s = 20$ appeared to be inferior selections. Moreover, $N_s = 20$ performed much worse than $N_s = 10$, which clearly outlined a trend of decreasing performance beyond a certain threshold.

At the same time we would like to point out that the above tuning procedure was by no means exhaustive and we would rather advocate for a certain qualitative performance pattern with respect to N_s, than for particular N_s values. In conclusion, $N_s = 5$ was considered sufficient and used in further comparisons.

Table 2: Scores achieved by psLSHADE with $N_s = \{2, 5, 10, 20\}$ for $10^3 \cdot D$ optimization budget.

Algo Score	$N_s = 2$	$N_s = 5$	$N_s = 10$	$N_s = 20$
SNE	34.87	30.80	33.59	41.42
SR	101.25	93.75	133.75	171.25
Score 1	44.17	50.00	45.84	37.18
Score 2	46.30	50.00	35.05	27.37
Score	**90.46**	**100.00**	**80.89**	**64.56**

4.3 Comparison with LSHADE and MadDE

A comparison of psLSHADE with LSHADE and MadDE was carried out with three optimization budgets: $10^2 \cdot D$, $10^3 \cdot D$, and $10^4 \cdot D$. MadDE, utilizing the same CEC2021 benchmark but with a default optimization budget, outperformed several other algorithms (AGSK [18], LSHADE [27], LSHADE_cnEpSin [3], j2020 [8], and IMODE [22]) in a comparison reported by the MadDE authors [5]. Hence, due to its superior performance, MadDE presents itself as a strong competitive method. A comparison with the baseline LSHADE, also being a highly efficient approach, directly verifies the value of the proposed pre-screening enhancement.

The comparison results for all three optimization budgets are shown in Table 3. psLSHADE distinctly outperforms both LSHADE and MadDE with the smallest budget of $10^2 \cdot D$. The differences are significant, especially when looking at the partial measures *SNE* and *SRE*. Moreover, being superior in non-expensive optimization budgets, MadDE is in this case less efficient than baseline LSHADE. A highly restricted optimization budget might induce the mediocre

performance of MadDE due to instability of some of its adaptation procedures.

In the regime of $10^3 \cdot D$ budget the results are qualitatively similar to those obtained for $10^2 \cdot D$ budget. psLSHADE still outperforms both competitors, although its advantage is not so prevailing. MadDE remains the least efficient approach.

The $10^4 \cdot D$ optimization budget can be classified as a borderline between expensive and non-expensive scenarios. In this case, all algorithms perform at a similar level, with MadDE being slightly less effective than the remaining two methods. Considering the *Score* metrics, psLSHADE seems to be marginally less efficient than LSHADE, however, its *SNE* value outperforms that of LSHADE. A relatively weaker psLSHADE performance for larger budgets may be caused by the decreasing model fit, as the number of evaluations increases. This phenomenon is illustrated further in Section 5.

Table 3: Scores achieved by MadDE, LSHADE and psLSHADE with $N_s = 5$ and $10^2 \cdot D$, $10^3 \cdot D$, $10^4 \cdot D$ optimization budgets.

O. b.	Algo Score	MadDE	LSHADE	psLSHADE
$10^2 \cdot D$	SNE	41.72	32.38	19.58
	SR	131.00	110.50	58.50
	Score 1	23.46	30.23	50.00
	Score 2	22.33	26.47	50.00
	Score	**45.79**	**56.70**	**100.00**
$10^3 \cdot D$	SNE	37.92	25.38	20.36
	SR	125.75	104.50	69.75
	Score 1	26.85	40.10	50.00
	Score 2	27.73	33.37	50.00
	Score	**54.58**	**73.48**	**100.00**
$10^4 \cdot D$	SNE	28.50	26.81	26.37
	SR	105.50	94.00	100.50
	Score 1	46.26	49.18	50.00
	Score 2	44.55	50.00	46.77
	Score	**90.81**	**99.18**	**96.77**

To sum up, psLSHADE is highly effective in scenarios with restricted FFE budgets ($10^2 \cdot D$ and $10^3 \cdot D$), leaving both competitors: LSHADE and MadDE behind. In the case of $10^4 \cdot D$ budget, psLSHADE is narrowly worse than LSHADE and slightly more effective than MadDE.

4.4 Computational complexity

All experiments were conducted using the following system setup. OS: Windows 10, CPU: Intel Core i7-4700MQ (2.40Ghz), RAM: 16GB, Language: Matlab R2020a, Compiler: MinGW64 C/C++ Compiler.

psLSHADE and LSHADE complexity was computed according to the CEC2021 technical report [17] and is presented in Table 4. Computational complexity of psLSHADE is clearly higher than that of LSHADE (by a factor of 25.8 and 72.8 for 10D and 20D functions, resp.) which is caused by additional operations stemming from pre-screening utilization.

At the same time, it should be underlined that psLSHADE complexity does not increase with the number of FFEs. Since the meta-model utilizes the archive of samples of limited size of N_a, after the first N_a FFEs, all algebraic operations related to meta-model estimation maintain a constant complexity. Therefore, the requested optimization time, counted in seconds, is linearly dependent on the FFE budget.

On a general note, the *"overhead"* of psLSHADE optimization time (excluding the time of FFEs) with $2 \cdot 10^5$ FFEs, equals approx. $34s/83s$ for a $10D/20D$ function, resp. ($T_2 - T_1$ in Table 4). Considering the perspective of expensive optimization (where FFEs are typically costly), the reported times seem acceptable.

Table 4: Computational complexity of psLSHADE and LSHADE calculated according to [17] (i.e. for one benchmark representative). T_0 **is the time of a test program run.** T_1 **- time of pure** $2 \cdot 10^5$ **FFEs of** F_1 **function.** T_2 **- the average running time of the algorithm for** F_1 **with** $2 \cdot 10^5$ **evaluation budget.** $(T_2 - T_1)/T_0$ **is the final complexity.**

D	Algorithm	$T_0[s]$	$T_1[s]$	$T_2[s]$	$(T_2 - T_1)/T_0$
10	psLSHADE		11.3341	44.8177	6916.6669
	LSHADE	0.004841	0.2027	1.5016	268.3196
20	psLSHADE		10.9902	93.9839	17143.9248
	LSHADE		0.2040	1.3442	235.5063

4.5 Convergence of psLSHADE

The experimental results indicate that the pre-screening mechanism is most advantageous in the case of $10^2 \cdot D$ optimization budget. However, for the sake of generality of the presented observations, in further analysis of psLSHADE performed in the reminder of the paper a bigger (i.e. less favourable) budget of $10^3 \cdot D$ is considered.

Following [17] we recorded 16 error values ($f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$) for each optimization run after certain FFE counts. For each function, we obtained 150 vectors (5 transformations × 30 repetitions) of error values. Figure 1 presents the convergence of psLSHADE and LSHADE, calculated as the average error value after each FFE count point, separately for each 20D function. The respective plots for 10D functions are presented in the supplementary material.

In 77 of 100 test cases (functions × transformations × dimensions), the difference between psLSHADE and LSHADE was significant (Mann–Whitney test, *p-value*=0.05). In none of the cases psLSHADE was significantly worse than LSHADE. The results demonstrate that superior convergence of psLSHADE over LSHADE can be observed for 9 out of 10 functions. It is worth noting that an improved convergence is especially noticeable in the initial optimization phase, depending on the function until 5000-15000 FFEs. In the final optimization phase, the difference starts to diminish, on average.

The most notable increase of convergence occurs for function F_1, whereas for function F_2 the advantage is negligible. The 2D versions of both functions are presented in Figure 2. F_1 is unimodal and smooth, while F_2 is multi-modal with a huge number of local optima. Moreover, F_2 does not possess a visible global structure, making the

global meta-model not so much useful in this case. Nevertheless, the meta-model does not lead to premature convergence, which is always a risk with such ill-conditioned functions.

The convergence based on error values is a key performance indicator, however, it does not explain how the pre-screening mechanism affects the population. Hence, we designed an experiment in which we measured the hyper-volume (h-v) of the population in each generation g. The h-v represents the volume of a hyperrectangle spanning a population and helps to estimate the population dispersion. Formally, the h-v is calculated as follows:

$$h^g = \prod_d^D \left(max(x_{i,d}^g) - min(x_{i,d}^g) \right) \tag{18}$$

Figure 3 demonstrates changes of the h-v sizes of psLSHADE and LSHADE during an optimization run for F_1 and F_2. Measurements were collected in each generation, but for consistency, the figure refers to the number of FFEs made so far in each generation. For each function, in a given generation, the h-v value represents the average of 150 values (5 transformations × 30 repetitions). The figure confirms that significant convergence increase in psLSHADE vs. LSHADE (presented in Figure 1) is strongly correlated with the h-v decrease. For function F_1, with a substantial global structure, the use of pre-screening results in generally faster convergence of each individual (to the global optimum), which results in the h-v reduction. No significant change of h-v is observed for F_2, which confirms the assumption that in this case the pre-screening did not cause premature convergence to a local optimum, i.e. the search ability of psLSHADE has been preserved. The respective figures for the remaining functions in both 10D and 20D versions are included in the supplementary material.

5 META-MODEL PERFORMANCE

Consistent with the experimental evaluation, the usefulness of incorporating the pre-screening mechanism into LSHADE has been demonstrated. Notwithstanding, the convergence plots signify some issues with the meta-model in the final optimization phase. For most functions, the advantage accumulated earlier begins to diminish at the end of $10^3 \cdot D$ optimization budget. Two hypotheses regarding this phenomenon emerge. The first one assumes that the global-meta model is well-fitted at the beginning due to the regular dispersion of the initial population. However, as the population loses regularity during the optimization run, the meta-model becomes not fitted correctly, i.e. its selection of trial vectors behaves more randomly. Please recall that the meta-model utilizes the archive containing N_a best-so-far evaluated samples. Consequently, the global meta-model may tend towards the local model.

The other conjecture is a more pessimistic version of the first one and assumes that the meta-model not only tends to behave randomly, but at some point may even start to select worse-than-average trial vectors.

In order to verify these conjectures, we examined the accuracy of the meta-model during an optimization run. In a dedicated experiment we recorded whether the trial vector $\boldsymbol{u}_i^{g,best}$ selected from all N_s trial vectors $\boldsymbol{u}_i^{g,j}$ for FFE was, indeed, the best one in terms of the fitness function value (not only the meta-model estimation). Each iteration generated N^g logical values: *true* or *false*, referring
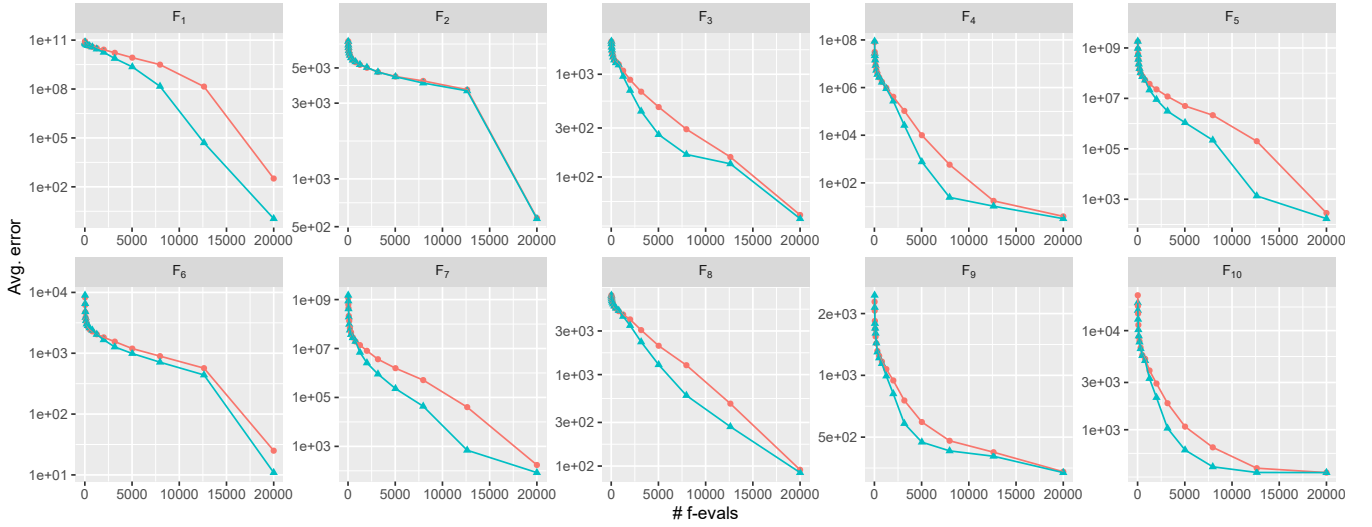
Figure 1: The averaged convergence of psLSHADE (blue line) and LSHADE (red line) by function in 20D with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs, and the y-axis the average error (a difference from the optimum).
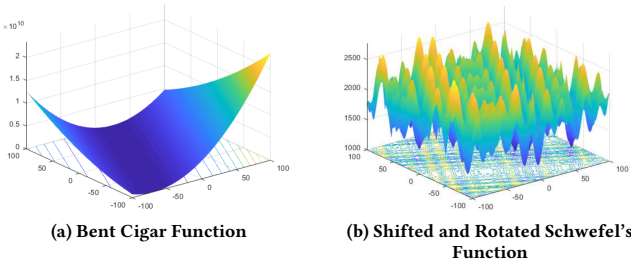

(a) Bent Cigar Function

(b) Shifted and Rotated Schwefel's Function

Figure 2: The 3D maps of $2D$ versions of functions F_1 (a) and F_2 (b). The figures are reprinted from [17].

Figure 3: The hyper-volume (h-v) of psLSHADE (blue line) and LSHADE (red line) for F_1 and F_2 in 20D with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs, and the y-axis the average h-v in each generation.

to N^g individuals. For each function and each iteration, there were $150 \cdot N^g$ (5 transformations × 30 repetitions ×N^g) logical values obtained. Calculating the share of *true* values in all gathered values leads to the average meta-model accuracy per function and iteration. Figuratively speaking, the perfect meta-model will gain the average accuracy of 1 in each iteration. In contrast, an entirely random meta-model will achieve the average accuracy $\frac{1}{N^s}$.

Figure 4 presents the average meta-model accuracy for functions F_1, F_2, F_8 and F_9 in 20D. For F_1, the accuracy is close to 1 in most iterations and never falls below the randomness threshold (0.2). However, a significant drop of nearly 0.7 after $1.2 \cdot 10^4$ evaluations can be observed. This decline may be caused by the convergence of some trials to global optimum earlier than after $10^3 \cdot D$, in which case the meta-model has become irrelevant.

For F_2, the meta-model accuracy is as expected. The proposed global meta-model was not able to estimate the ill-conditioned function properly and pre-screen samples better than randomly. Towards the end, the average accuracy began to increase. The reason for this phenomenon may be convergence to some local optimum.

F_8 and F_9 were chosen because their convergence plots indicated a notable decrease in meta-model accuracy in the final phase of the optimization run. The meta-model accuracy for F_8 remains significantly above the randomness threshold until approx. $1.4 \cdot 10^4$ FEEs. Contrasting this observation with the convergence plot (Figure 1), we observe a similar point ($\approx 1.4 \cdot 10^4$) when the convergence, compared to LSHADE, starts to decline. Regardless, the accuracy of the model after $1.5 \cdot 10^4$ FFEs is concerning. It decreased below the randomness threshold. At the end of the budget, the convergence plots of psLSHADE and LSHADE almost meet.

For F_9, a significant reduction in meta-model accuracy occurred earlier, so the convergence advantage of psLSHADE over LSHADE also started to decline earlier.

The real-time evaluation of the meta-model performance could be beneficial for two reasons. Firstly, deactivation of useless, in particular cases, pre-screening can reduce the computational complexity (see Table 4). Secondly, the F_8 case showed that meta-model predictions might sometimes be worse than random, so deactivation of the meta-model seems even more reasonable in this case.
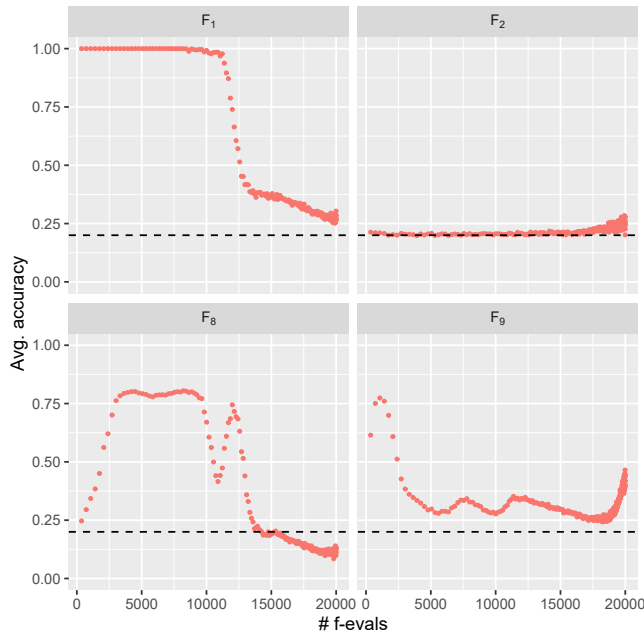
Figure 4: The average accuracy of meta-model selections in each generation for functions F_1, F_2, F_8 and F_9 in 20D with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average accuracy.

Figure 5: The average R^2 and Kendall's τ in each generation for functions F_1, F_2, F_8 and F_9 in 20D with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average values of R^2 and Kendall's τ.

Unfortunately, the *true/false* accuracy measure presented above is not suitable for real-world implementations due to the 5 times higher number of costly FFEs. Therefore, we conducted a related experiment but used other performance metrics. The first criterion was the well-known coefficient of determination $R^2 \in [0, 1]$. In each iteration, its value was designed after the meta-model had been fitted. The second criterion was Kendall's $\tau \in [0, 1]$, measuring the rank correlation between N^g fitness function values $f(\boldsymbol{u}_i^{g,best})$ and their respective meta-models estimates $f^{surr}(\boldsymbol{u}_i^{g,best})$. Kendall's τ, like R^2, was determined in each generation. Importantly, both measures are cost-free in terms of FFEs (no auxiliary evaluations are required).

The results obtained for the same set of 4 functions are presented in Figure 5. The main disadvantage of R^2 is its uncertain impact on the final performance, i.e. the meta-model can be overfitted or inaccurate because of a small number of samples utilized in coefficient estimation. Nonetheless, for both criteria (R^2 and Kendall's τ) the results are highly similar in shape to the accuracy measure. Therefore, we assume they can potentially be helpful in conditional disabling of the meta-model in practical applications.

6 CONCLUSION

In this paper we introduced the psLSHADE algorithm that enhances the well-known LSHADE method with the pre-screening mechanism. On a popular CEC2021 benchmark, the proposed method outperformed LSHADE and MadDE in expensive scenarios (restrictive budget of $10^2 \cdot D$ and $10^3 \cdot D$ FFEs). In $10^4 \cdot D$ budget,
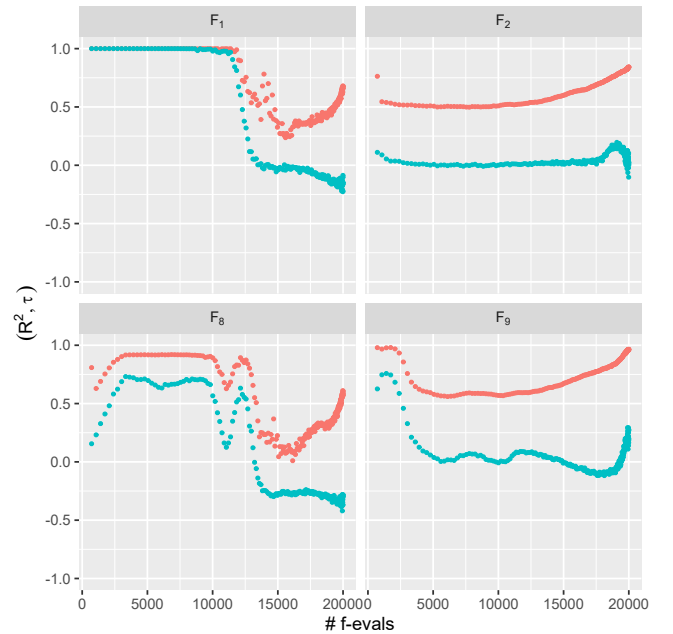
psLSHADE performed on par with LSHADE and continued to outperform MadDE.

Additionally, a comprehensive analysis of the meta-model performance is presented, including accuracy, fit (R^2), and rank correlation (Kendall's τ). We also investigate in which cases the pre-screening is indeed beneficial and how does it affect the population in terms of its h-v. Finally, we demonstrate that the meta-model performance can potentially be monitored in real-time, so as to activate the pre-screening solely when advantageous.

The future work concerns further analysis and development of deactivation conditions of the meta-model (extending the remarks presented at the end of section 5).

REFERENCES

[1] Anne Auger and Nikolaus Hansen. 2005. A restart CMA evolution strategy with increasing population size. In *2005 IEEE congress on evolutionary computation*, Vol. 2. IEEE, 1769–1776.

[2] Anne Auger, Marc Schoenauer, and Nicolas Vanhaecke. 2004. LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In *International Conference on Parallel Problem Solving from Nature*. Springer, 182–191.

[3] Noor H Awad, Mostafa Z Ali, and Ponnuthurai N Suganthan. 2017. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 372–379.

[4] Lukáš Bajer, Zbyněk Pitra, Jakub Repickỳ, and Martin Holeňa. 2019. Gaussian process surrogate models for the CMA evolution strategy. *Evolutionary computation* 27, 4 (2019), 665–697.

[5] Subhodip Biswas, Debanjan Saha, Shuvodeep De, Adam D Cobb, Swagatam Das, and Brian A Jalaian. 2021. Improving differential evolution through bayesian hyperparameter optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 832–840.

[6] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. 2013. A survey on optimization metaheuristics. *Information sciences* 237 (2013), 82–117.

[7] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. 2006. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation* 10, 6 (2006), 646–657.

[8] Janez Brest, Mirjam Sepesy Maučec, and Borko Bošković. 2020. Differential evolution algorithm for single objective bound-constrained optimization: Algorithm j2020. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[9] Noel Cressie. 1990. The origins of kriging. *Mathematical geology* 22, 3 (1990), 239–252.

[10] Nikolaus Hansen. 2019. A global surrogate assisted CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 664–672.

[11] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation* 11, 1 (2003), 1–18.

[12] Jon C Helton and Freddie Joe Davis. 2003. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety* 81, 1 (2003), 23–69.

[13] Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1, 2 (2011), 61–70.

[14] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

[15] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[16] Stefan Kern, Nikolaus Hansen, and Petros Koumoutsakos. 2006. Local meta-models for optimization using evolution strategies. In *Parallel Problem Solving from Nature-PPSN IX*. Springer, 939–948.

[17] Ali Wagdy Mohamed, Anas A Hadi, Ali Khater Mohamed, Prachi Agrawal, Abhishek Kumar, and P.N Suganthan. [n. d.]. Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. https://github.com/P-N-Suganthan/2021-SO-BCO/blob/main/CEC2021%20TR_final%20(1).pdf.

[18] Ali Wagdy Mohamed, Anas A Hadi, Ali Khater Mohamed, and Noor H Awad. 2020. Evaluating the performance of adaptive GainingSharing knowledge based algorithm on CEC 2020 benchmark problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[19] Kouhei Nishida and Youhei Akimoto. 2018. Benchmarking the PSA-CMA-ES on the BBOB noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1529–1536.

[20] Michał Okulewicz and Mateusz Zaborski. 2021. Benchmarking SHADE algorithm enhanced with model based optimization on the BBOB noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1259–1266.

[21] A Kai Qin, Vicky Ling Huang, and Ponnuthurai N Suganthan. 2008. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation* 13, 2 (2008), 398–417.

[22] Karam M Sallam, Saber M Elsayed, Ripon K Chakrabortty, and Michael J Ryan. 2020. Improved multi-operator differential evolution algorithm for solving unconstrained problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[23] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.

[24] P.N Suganthan. [n. d.]. Code of top methods. https://github.com/P-N-Suganthan/2021-SO-BCO/blob/main/Codes-of-top-methods%20(1).zip.

[25] Ryoji Tanabe and Alex Fukunaga. 2013. Success-history based parameter adaptation for differential evolution. In *2013 IEEE congress on evolutionary computation*. IEEE, 71–78.

[26] Ryoji Tanabe and Alex Fukunaga. 2015. Tuning differential evolution for cheap, medium, and expensive computational budgets. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018–2025.

[27] Ryoji Tanabe and Alex S Fukunaga. 2014. Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 1658–1665.

[28] Pradnya A Vikhar. 2016. Evolutionary algorithms: A critical review and its future prospects. In *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)*. IEEE, 261–265.

[29] Sanford Weisberg. 2013. *Applied linear regression*. John Wiley & Sons.

[30] Takahiro Yamaguchi and Youhei Akimoto. 2017. Benchmarking the novel CMA-ES restart strategy using the search history on the BBOB noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1780–1787.

[31] Mateusz Zaborski, Michał Okulewicz, and Jacek Mańdziuk. 2020. Analysis of statistical model-based optimization enhancements in Generalized Self-Adapting Particle Swarm Optimization framework. *Foundations of Computing and Decision Sciences* 45, 3 (2020), 233–254.

[32] Jingqiao Zhang and Arthur C Sanderson. 2009. JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on evolutionary computation* 13, 5 (2009), 945–958.

— *SUPPLEMENTARY MATERIAL* —

In this supplementary material we extend the results presented in the main body of the paper. Figure 6, for each 10D function from CEC2021 [17], illustrates the convergence of psLSHADE and LSHADE calculated as the average error value after each FFE count point. This figure is an analog of Figure 1 in the main text which shows the same plots for 20D functions. Generally speaking, the result for 10D and 20D are similar when comparing the individual functions and the conclusions drawn in the paper with respect to 20D functions are also valid for 10D.

Figures 7 and 8 present the changes of the hyper-volumes of psLSHADE and LSHADE during an optimization run, for 10D and 20D functions, respectively. In the main body of the paper (Figure 3), hyper-volume changes are demonstrated only for F_1 and F_2 in 20D. The impact of pre-screening on the hyper-volume is visible for all functions in both dimensions (10D and 20D), however its scale varies depending on the function.

Figures 9 and 10 present the average accuracy of the meta-model selections in each generation, for 10D and 20D functions, respectively. The figures extend Figure 4 from the main paper which presents the same results for 20D versions of the selected 4 functions: F_1, F_2, F_8, and F_9. The general trend of diminishing accuracy of estimates is apparent for all functions and does not vary significantly between 10D and 20D cases. The only exception is F_8, for which the characteristic for 20D is slightly different than for 10D (one can observe a spike in accuracy around $1.2 \cdot 10^4$ FFEs in 20D). A deeper analysis showed that this spike can be observed in all 4 transformations containing *shift*. The 2D versions of F_8 and its contour plot are presented in Figure 13. As can be seen in the figure, F_8 is a composition multi-modal function, but with a significant smooth area. This smooth area can be relatively well approximated by the meta-model. We hypothesize that the spike occurs when the archive begins to contain mainly samples belonging to the smooth area, which increases the accuracy. This phenomenon is not visible for the 10D case, most likely because the shift transformation in 10D is less severe due to the specific location of the global optimum. A full explanation of this phenomenon is the subject of future research.

Figures 11 and 12 depict the average R^2 and Kendall's τ for functions in 10D and 20D, respectively. The results extend Figure 5 in the main paper that relates exclusively to 20D versions of functions F_1, F_2, F_8 and F_9. Generally, there are no particular differences between the 10D and 20D plots. although function F_8 again behaves a little differently between 10D and 20D cases. The reasons for this phenomenon are most probably the same as for the case of accuracy. The unusual increase of the meta-model fit, defined by R^2, further confirms the above-presented hypothesis.
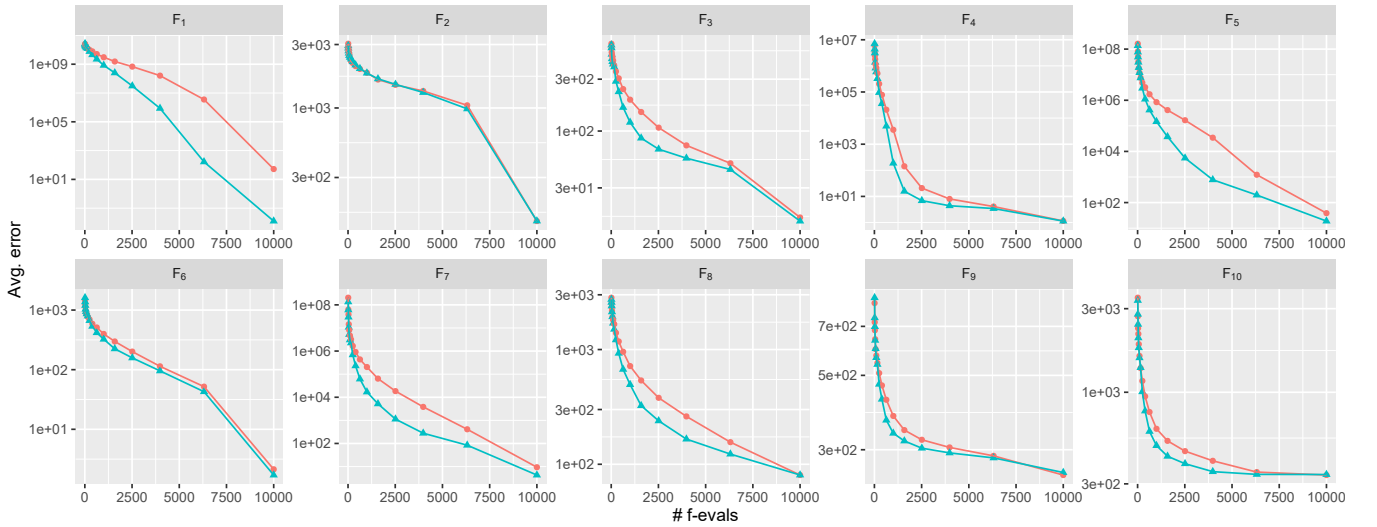

Figure 6: The averaged convergence of psLSHADE (blue line) and LSHADE (red line) for all 10D functions from CEC2021 with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average error.
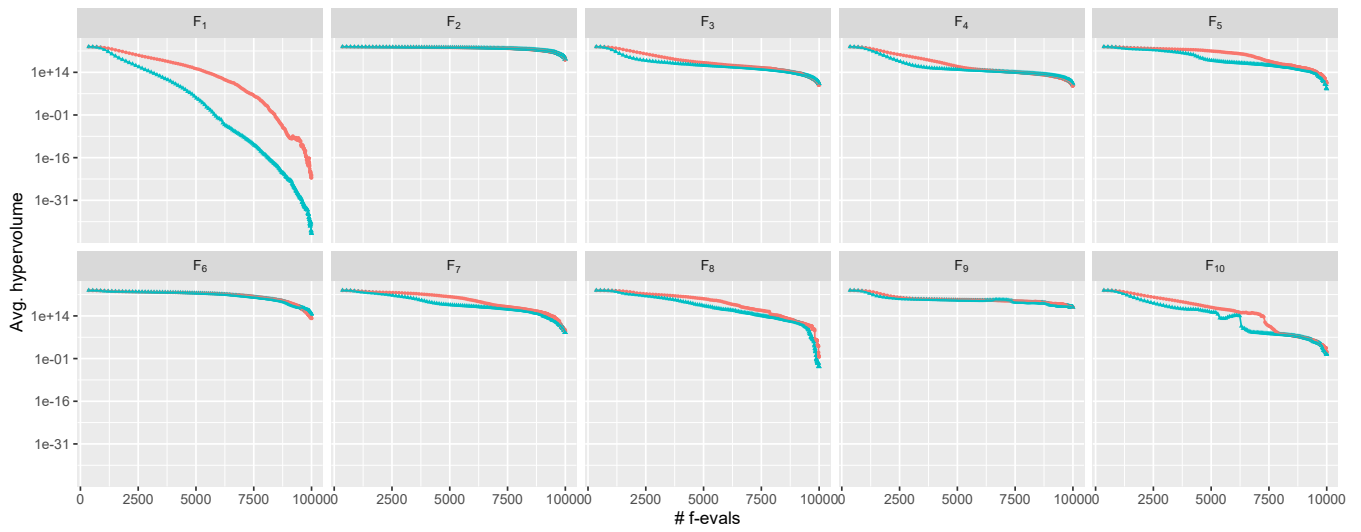
Figure 7: The hyper-volme of psLSHADE (blue line) and LSHADE (red line) for all 10D functions from CEC2021 with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average hypervolume size in each generation.
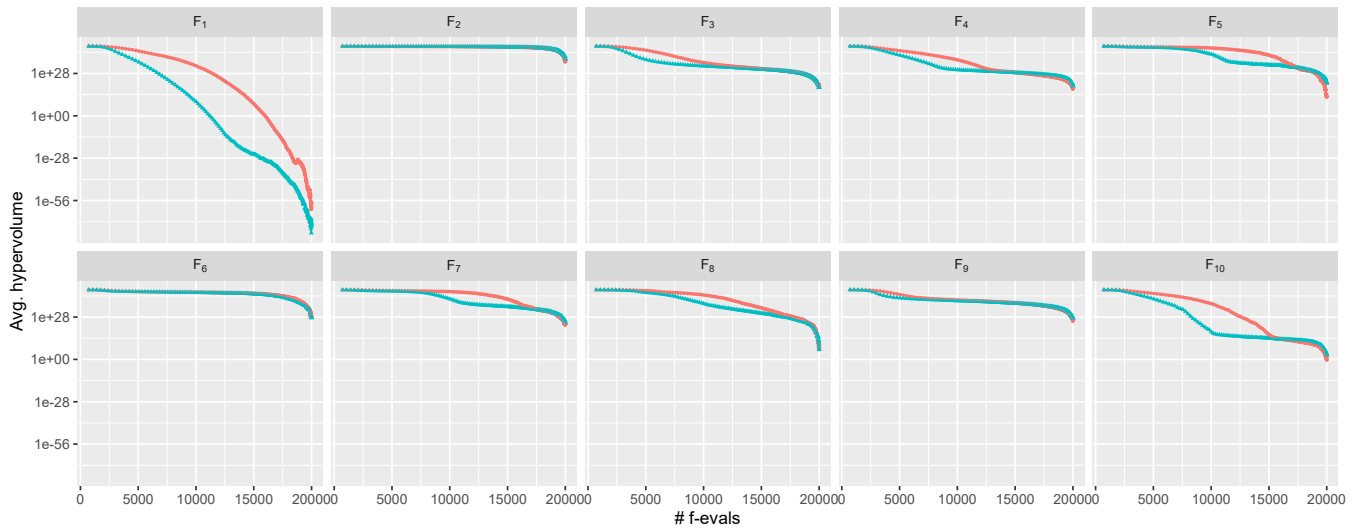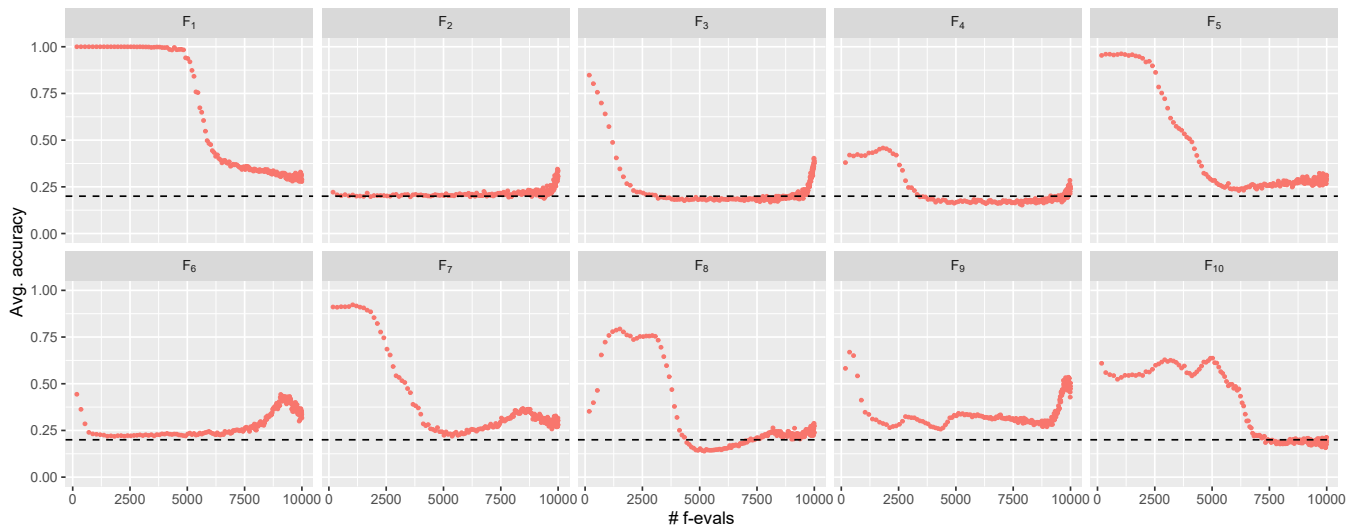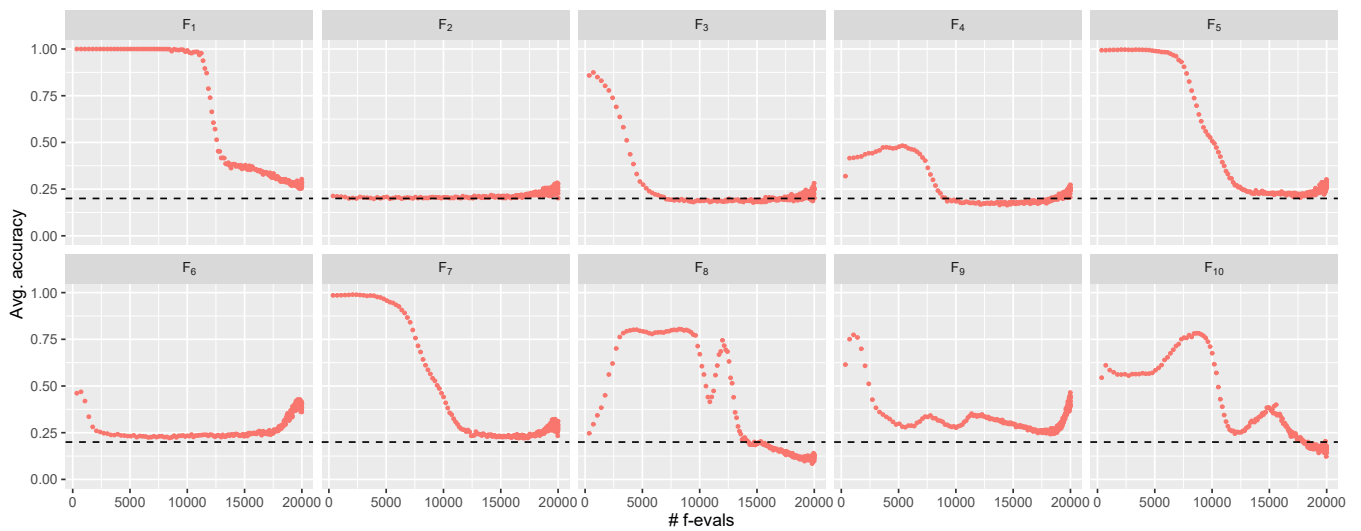

Figure 8: The hyper-volme of psLSHADE (blue line) and LSHADE (red line) for all 20D functions from CEC2021 with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average hypervolume size in each generation.

Figure 9: The averaged accuracy of meta-model selection in each generation for all 10D functions from CEC2021 with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average accuracy.
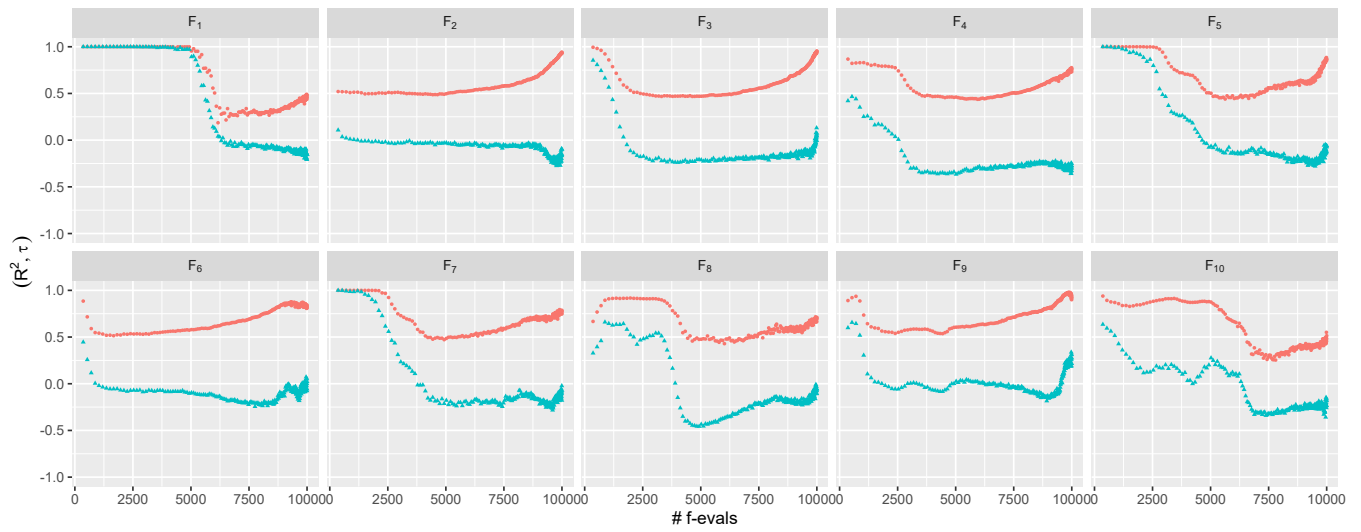

Figure 10: The average accuracy of meta-model selection in each generation for all 20D functions from CEC2021 with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average accuracy.

Figure 11: The average R^2 and Kendall's τ in each generation for all 10D functions from CEC2021 with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average values of R^2 and Kendall's τ.
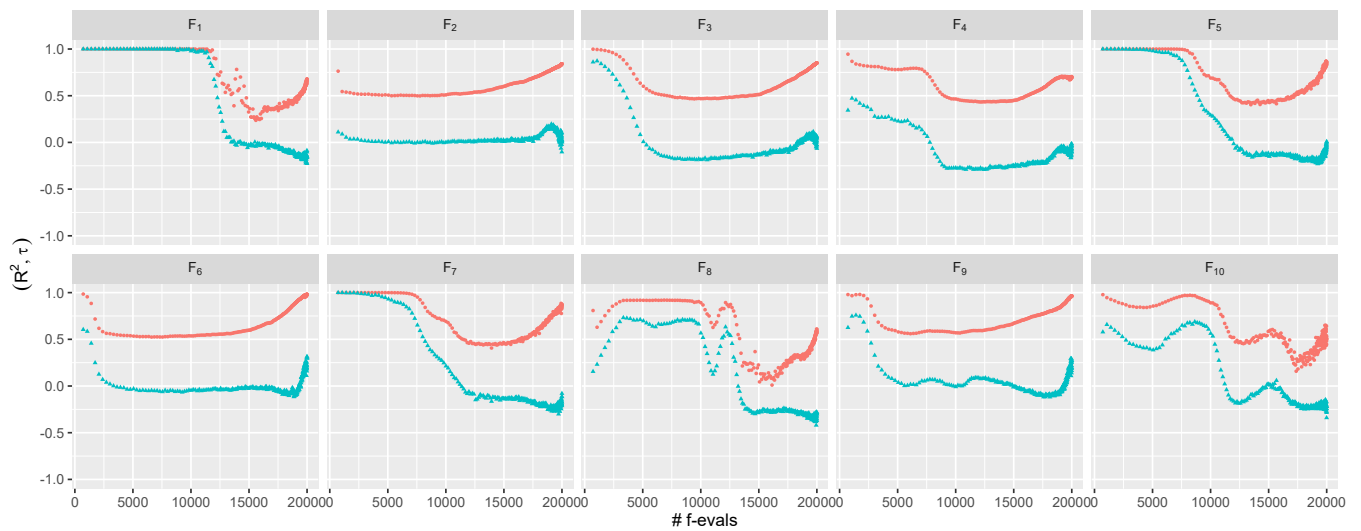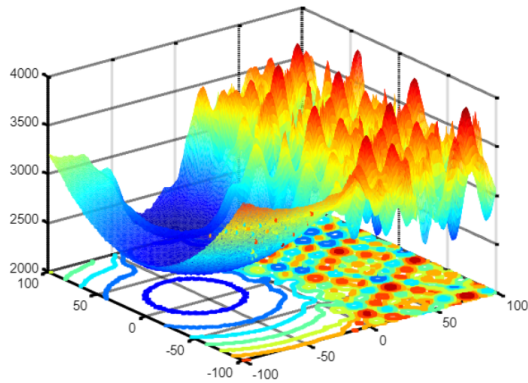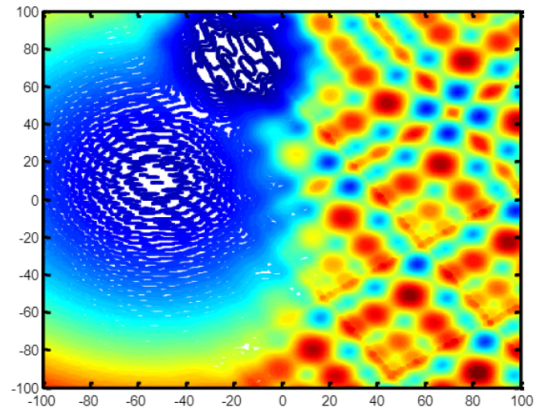

Figure 12: The average R^2 and Kendall's τ in each generation for all 20D functions from CEC2021 with $10^3 \cdot D$ optimization budget. The x-axis represents the number of FFEs and the y-axis the average values of R^2 and Kendall's τ.

(a) The 3D maps of $2D$ version of composition function F_8

(b) The 2D contour map of $2D$ version of composition function F_8

Figure 13: The 3D maps of $2D$ version of composition function F_1 (a) and its contour map (b) The figures are reprinted from [17].