# Walsh-based surrogate-assisted multi-objective combinatorial optimization: A fine-grained analysis for pseudo-boolean functions

Bilel Derbel, Geoffrey Pruvost, Arnaud Liefooghe, Sébastien Verel, Qingfu Zhang

# Walsh-based Surrogate-assisted Multi-objective Combinatorial Optimization: A Fine-grained Analysis for Pseudo-boolean Functions*

Bilel Derbel[a,*], Geoffrey Pruvost[a], Arnaud Liefooghe[a], Sébastien Verel[b], Qingfu Zhang[c]

[a] *Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France*
[b] *Univ. Littoral Côte d'Opale, UR 4491, LISIC, F-62100 Calais, France*
[c] *City University of Hong Kong, Kowloon Tong, Hong Kong*

**Abstract**

The aim of this paper is to study surrogate-assisted algorithms for expensive multiobjective combinatorial optimization problems. Targeting pseudo-boolean domains, we provide a fine-grained analysis of an optimization framework using the Walsh basis as a core surrogate model. The considered framework uses decomposition in the objective space, and integrates three different components, namely, (i) an inner optimizer for searching promising solutions with respect to the so-constructed surrogate, (ii) a selection strategy to decide which solution is to be evaluated by the expensive objectives, and (iii) the strategy used to setup the Walsh order hyper-parameter. Based on extensive experiments using two benchmark problems, namely bi-objective NK-landscapes and unconstrained binary quadratic programming problems (UBQP), we conduct a comprehensive in-depth analysis of the combined effects of the considered components on search performance, and provide evidence on the effectiveness of the proposed search strategies. As a by-product, our work shed more light on the key challenges for designing a successful surrogate-assisted multi-objective combinatorial search process.

*Keywords:* Multi-objective optimization, discrete surrogates, decomposition

## 1. Introduction

Surrogate-assisted algorithms have been recognized as a mainstream methodology for expensive optimization where objective function evaluation is economically costly and/or time consuming [1–9]. These algorithms build and use surrogate models for estimating objective function values in order to reduce as many as possible the number of true expensive function evaluations. Although

*This work was supported by the French national research agency (ANR-16-CE23-0013-01) and the Research Grants Council of Hong Kong (RGC Project No. A-CityU101/16).

*Corresponding author

*Email addresses:* bilel.derbel@univ-lille.fr (Bilel Derbel), geoffrey.pruvost@univ-lille.fr (Geoffrey Pruvost), arnaud.liefooghe@univ-lille.fr (Arnaud Liefooghe), verel@univ-littoral.fr (Sébastien Verel), qingfu.zhang@cityu.edu.hk (Qingfu Zhang)

much effort has been made to develop surrogate-assisted algorithms, a number of difficult issues still need to be investigated [1]. In this paper, we are specifically concerned with the design of both *multi-objective* and *discrete* surrogate-assisted optimization algorithms. Interestingly, these two aspects, taken separately, have been identified in [1] as among of the five major challenges that require further in-depth investigations from the community.

Multi-objective optimization problems (MOP) [10, 11] are a natural outcome in different application fields where a number of conflicting objectives are to be optimized in a simultaneous manner. Our goal is to compute a representative set of solutions exposing different trade-offs in the objective space, i.e., a Pareto front approximation. Among the different techniques to tackle MOPs, evolutionary multi-objective algorithms (EMOA) [12, 13] have been proved to be extremely effective, especially due to their ability to evolve a whole set of solutions mapping the target approximation set. In an expensive optimization setting, EMOAs have been leveraged with surrogates in a number of different ways, and for different problems and application domains. Generally speaking, two major inter-dependent issues are considered: (i) the meta-model being used as a surrogate, and (ii) the class of the multi-objective technique used at the core of the underlying surrogate-assisted optimization process. On the one hand, standard statistical and machine-learning meta-models, ranging from kriging [14] and Gaussian process (GP) [15], to support vector regressions (SVR) [16], radial basis functions (RBF) [17], artificial neural networks (ANN) [18], decision trees and random forest (RF) [19, 20], etc, can be considered to learn a blackbox objective function. On the other hand, these models can be coupled with different evolutionary multi-objective search paradigms, such as decomposition [21, 22], dominance [23–25], and indicators [26, 27]; hence, ending-up with a variety of surrogate-assisted algorithms, such as Par-EGO [28], MOEA/D-EGO [29], MOEA/D-RBF [30], KRVEA [31], Multi-objective EGO [32], SMS-EGO [33], CSA-MOEA [34], CSEA [35], SGMOO [36], to cite a few. The existing literature is too vast to discuss in the scope of this paper. We hence refer the reader to the existing taxonomies [7, 8, 37] and surveys [2, 3, 6, 9] for a more thorough overview.

The multi-objective algorithms cited before were mainly intended to work with continuous decision variables. However, expensive optimization problems are not specific to continuous domains; in fact, many real-world applications are known to have discrete variables, e.g., strings, integers, graphs, permutations, etc. More importantly, continuous evolutionary approaches cannot be transferred as such to combinatorial domains. Firstly, given the difficulty of modeling the landscape of combinatorial problems, it is still not clear how to derive a discrete surrogate model that can accurately learn the structure of a combinatorial (single-objective) problem. This is to contrast with the fact that there exist well-established continuous surrogates with well-studied properties. However, one can find a number of investigations for designing discrete surrogates, ranging from naive approaches ignoring the discrete structure of the search space, to more domain-specific modeling techniques [3, 38–42]. In particular, some of the previously mentioned meta-models, such as GP, RBF, RF, ANN, can be adapted to support mixed (continuous and discrete) variables.

Secondly, assuming an accurate discrete surrogate can be used, it is not clear how to finely couple the surrogate model with the optimization process for an optimal performance. Such an issue is even more pressing for multi-objective combinatorial problems, given the variety of discrete spaces that can be considered, and the specific challenges underlying a multi-objective search. Finally, benchmarking surrogate-assisted combinatorial multi-objective algorithms is a challenging research question. Benchmarking is highlighted in [1] as *the* most important challenge, which is (quoting the authors in [1]) "an open issue throughout the whole field of surrogate-assisted optimization and a mandatory step towards identifying strengths and weaknesses of approaches". In this respect, the systematic analysis of discrete surrogate-assisted approaches is even less developed compared to continuous domains, and a lot of effort has to be performed before ending up with a unified view of what makes an approach successful as a function of the target search space.

Our work aims at contributing to the development of new efficient combinatorial surrogate-assisted evolutionary multi-objective algorithms (CS-EMOA), while providing a rigorous analysis of the impact of the underlying design components through extensive benchmarking. Looking at the specialized literature, one can find a number of approaches targeting different discrete search spaces, and/or designed within a number of application-specific contexts, e.g., [43–49] to cite a few. For example, in [50], a data-driven multi-objective approach is designed specifically to constrained discrete problems. Random forests are employed as surrogates, together with logistic regression models to rectify the non-dominated ranking for the constraint functions. The derived approach is experimented on constrained multi-objective knapsack problems, and applied to trauma systems. In [51], a multi-objective task-oriented pattern mining problem is tackled using an ensemble of surrogates derived from a discrete hamming-distance based RBF network. In contrast to previous work, we focus on designing and analyzing CS-EMOAs for pseudo-boolean MOPs, viewed from a high-level perspective as a fundamental general-purpose class of optimization problems.

In a pseudo-boolean optimization problem, solutions are binary strings, and every solution is mapped to a real value for each objective. We do not make any further assumptions on the given problem, which is considered as blackbox. The blackbox problem is assumed to be expensive in the sense that the CPU time of evaluating the quality of one solution can range from at least some minutes, to hours. Pseudo-boolean optimization problems can arise in a number of real-world applications where decision variables typically model the presence/absence of design components. This is for instance the case in different engineering fields, e.g., bus stop design in public transportation systems [52, 53], bike sharing [54], drug discovery [55]. Independently of a particular real-world application, a number of relatively recent studies have investigated the design of discrete surrogate models, e.g., [39–42, 56–66]. Reviewing the theory behind existing models is out of the scope of this paper. We refer the reader to [3], and the reference therein, for a more detailed overview. In this paper, we rely on the so-called Walsh functions [67], which were shown to constitute a well-suited theoretical tool to derive custom surrogates for arbitrary pseudo-boolean functions [56, 57, 62–64]. In fact, discrete Walsh functions form an orthogonal set of functions allowing to uniquely represent

any blackbox pseudo-boolean function in an additive linear form. A first step towards integrating Walsh-based surrogates for expensive multi-objective optimization has been performed in our previous work presented in [68]. At this stage of the presentation, it is important to recall that besides a particular choice of the surrogate model, the successful design of a CS-EMOA depends on different other design choices and algorithmic components; especially, with respect to the smart configuration and training of the surrogate model, as well as, its combination with a multi-objective search process.

In this paper, we leverage our previous work from different perspectives. As in [68], we consider a decomposition-based paradigm [22] to handle the approximation set maintained by the (expensive) search process. However, we propose a number of novel design components and strategies to deal with three critical design aspects, namely, (i) the inner evolutionary optimizer used for searching promising offspring solutions with respect to the Walsh surrogate, (ii) the selection strategy allowing to decide which solution is to be evaluated by the expensive objectives, and (iii) the strategy used to setup the Walsh order hyper-parameter. Compared to the performance results presented in [68], we show that substantial improvements in terms of approximation quality can be obtained by the new proposed multi-objective search strategies. It is worth noticing that there exist no rule-of-thumb to configure the previously mentioned design components, which is a challenging research question. Therefore, besides being able to significantly improve over our previous work, we report our findings in light of a plug-and-play surrogate assisted framework which is benchmarked on the basis of a full factorial experimental design, and using a relatively broad range of benchmark functions. Hence, our results come with an in-depth comprehensive analysis on the combined effect of the different design choices, and sheds more light on the key ingredients for a successful surrogate-assisted multi-objective approach.

The rest of the paper is organized as follows. In Section 2, we recall the necessary background behind multi-objective decomposition and discrete Walsh surrogates. In Section 3, we describe the proposed approach in a step-by-step manner. In Section 4, we give our experimental setup. In Section 5, we report our empirical findings. In Section 6, we conclude the paper and we discuss future research directions.

## 2. Background

### *2.1. Multi-objective Combinatorial Optimization*

We assume that we are given a black-box objective vector function $F = (f_1, f_2, \ldots, f_m)$ to maximize, and a set $X$ of solutions in the *variable space*. When $X$ is a discrete set, we face a *multi-objective combinatorial optimization problem* (MCOP). In particular, we consider unconstrained pseudo-boolean multi-objective optimization problems, such that $F\colon \{0,1\}^n \mapsto \mathbb{R}^m$, where $n$ is the problem size. Let $Z = f(X) \subseteq \mathbb{R}^m$ be the set of feasible outcome vectors in the *objective space*. An objective vector $z \in Z$ is *dominated* by a vector $z' \in Z$ iff $\forall i \in [\![1, m]\!]$, $z_i \leqslant z'_i$, and

$\exists j \in [\![1, m]\!]$ s.t. $z_j < z'_j$. A solution $x \in X$ is dominated by a solution $x' \in X$ iff $F(x)$ is dominated by $F(x')$. A solution is *Pareto optimal*, or *non-dominated*, if there does not exist any other solution that dominates it. The set of all Pareto optimal solutions is the *Pareto set*. Its mapping in the objective space is the *Pareto front*. Identifying the Pareto set is known to be an NP-hard task for a wide range of MCOPs, and the Pareto set typically contains an exponential number of solutions [11]. As such, we often have to rely on a Pareto set *approximation*, for which a large number of multi-objective evolutionary algorithms have been proposed since the early nineties [12, 13].

### *2.2. Decomposition-based Multi-objective Optimization*

In the broad range of multi-objective evolutionary algorithms, approaches based on *decomposition* are amongst the state-of-the-art [21]. In particular, MOEA/D [22] decomposes the multi-objective optimization problem into a set of single-objective sub-problems that target different regions of the Pareto front. The population of solutions $\mathcal{P}_\mu = \{x^1, \ldots, x^\mu\}$ is evolved such that each solution $x^i$, $i \in [\![1, \mu]\!]$, is assigned to a weight vector $\omega^i$ corresponding to a given sub-problem. A sub-problem then seeks for a high-quality solution with respect to an aggregation function $\mathsf{g}(x \mid \omega^i, F)$, parameterized by the weight vector $\omega^i$. The population $\mathcal{P}_\mu$ is evolved following conventional evolutionary mechanisms, such as selection and variation, in order to optimize the different sub-problems.

Given a weight vector $\omega^i$, the aggregation function transforms an objective vector into a scalar value. One recommended function that we consider in this paper is the Chebyshev function (to be minimized):

$$\mathsf{g}(x \mid \omega, F) = \max_{j \in [\![1,m]\!]} \omega_j \cdot \left| z^\star_j - f_j(x) \right|$$

where $x \in X$, $\omega = (\omega_1, \ldots, \omega_m)$ is a positive weight vector, and $z^\star$ is a reference point s.t. $z^\star_j > f_j(x)$, $\forall x \in X$.

In MOEA/D, the population is evolved in a cooperative manner. A solution that is currently assigned to a given sub-problem can become parent for an offspring generated by another sub-problem. A newly-generated offspring is compared to solutions assigned to other sub-problems by means of their corresponding aggregation function. At a given iteration, offspring can replace multiple solutions from the population, assuming that they improve multiple sub-problems. This cooperation may be limited by a neighborhood relation among sub-problems. Different variants [21, 69] of MOEA/D consider different ways of managing the neighborhood. Following [68], we consider a setting of MOEA/D where the whole population is used in the selection and replacements steps. This is clearly motivated by the fact that allowing an offspring to enter the population as soon as it improves any sub-problem privileges convergence, which is to be encouraged under a limited (expensive) budget setting.

### 2.3. Walsh Surrogates

The development of accurate surrogate models for black-box *combinatorial* problems is a difficult challenge [1]. With respect to a pseudo-boolean optimization problem, a surrogate model using the Walsh functions [67] was proved to be particularly accurate [56]. For completeness, we recall the background behind such a model.

#### 2.3.1. Walsh Basis

Walsh functions [67] constitute an enumerable set of functions $\phi_\ell : [0,1] \to \{-1,1\}$ which composes a normal and orthogonal basis of the Hilbert space $L^2([0,1])$. Like the trigonometric functions of the Fourier basis, they can be used to decompose any function from the Hilbert space [67], and have been used since the late seventies in the theory of evolutionary computation [70]. Given a pseudo-boolean function $f\colon \{0,1\}^n \mapsto \mathbb{R}$, Walsh functions are defined as follows [56]. For any integer $\ell \in [\![0, 2^n - 1]\!]$, following the binary representation $\ell = \sum_i \ell_i 2^i$ with $\ell_i \in \{0,1\}$, the Walsh function $\phi_\ell : \{0,1\}^n \to \{-1,1\}$ is defined for any binary string $x = (x_1, \ldots, x_i, \ldots, x_n) \in \{0,1\}^n$ by:

$$\phi_\ell(x) = (-1)^{\sum_{i=1}^{n} \ell_i x_i} \tag{1}$$

The *order* of a Walsh function $\phi_\ell$, denoted by $o(\phi_\ell)$, is defined by the number of binary digits equal to 1 in the binary representation of $\ell$. For example, the Walsh function of order 0 is $\phi_0$, the Walsh functions of order 1 are $\phi_{2^p}$ for all integers $p \geqslant 0$, the Walsh functions of order 2 are $\phi_{2^p+2^{p'}}$ for all pairs of integers $p \neq p' \geqslant 0$, and so on.

#### 2.3.2. Exact Walsh Transform

The so-defined (finite) set of discrete functions is a normal orthogonal basis for the space of pseudo-boolean functions, i.e., $\forall \ell, \ell' \in [\![0, 2^n - 1]\!]$, $\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \phi_\ell(x) \cdot \phi_{\ell'}(x) = \delta_{\ell\ell'}$. As such, *any* pseudo-boolean function $f$ can be written as:

$$f(x) = \sum_{\ell=0}^{2^n-1} w_\ell \cdot \phi_\ell(x) \tag{2}$$

$$\text{s.t.} \quad w_\ell = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \cdot \phi_\ell(x) \tag{3}$$

#### 2.3.3. Approximated Walsh Decomposition

On one hand, the Walsh functions $\phi_\ell$ are uniquely defined and do not depend on the problem at hand; see Eq. (1). On the other hand, the values of the coefficients $w_\ell$ do depend on the considered function $f$, as given in Eq. (3). On top of that, there might exist an exponential number of non-zero coefficients in the exact Walsh transform as given in Eq. (2). Roughly speaking, the coefficients corresponding to some Walsh functions of a given order $\mathsf{O}$ capture the interaction among a set of $\mathsf{O}$ variables in function $f$; i.e., they render how the function value is affected when

the variables change. This means that, unless a large number of variables interact with each other, many coefficients are expected to be zero. This is the case for multiple combinatorial problems with a quadratic or cubic number of variable interactions, and hence a reasonable assumption in practice for a wide range of problem classes. Therefore, the idea developed initially in [56, 57] is to approximate $f$ using solely the Walsh functions up to a (small) constant order $\mathsf{O} \ll n$, and to use an estimate $\widehat{w}_\ell$ of the (unknown) coefficient $w_\ell$. More formally, given a *constant* order $\mathsf{O}$, a function $f$ can be approximated by the following model:

$$\tilde{f}(x \mid d) = \sum_{\ell \,:\, o(\phi_\ell) \leqslant \mathsf{O}} \widetilde{w}_\ell \cdot \phi_\ell(x) \tag{4}$$

For example, a second-order approximation can be rewritten:

$$\tilde{f}(x \mid 2) = \widetilde{w}_0 + \sum_{i=1}^{n} \widetilde{w}_i \cdot (-1)^{x_i} + \sum_{1 \leq i < j \leq n} \widetilde{w}_{ij} \cdot (-1)^{x_i + x_j}$$

where $\widetilde{w}_0$ is the zero-order estimated coefficient, and $\widetilde{w}_i$ and $\widetilde{w}_{ij}$ are first- and second-order estimated coefficients. Intuitively, the larger the order $d$, the more accurate the expansion to approximate the original function.

#### *2.3.4. Walsh Surrogate Model*

Constructing a discrete Walsh surrogate up to constant order $\mathsf{O}$ consists of computing the approximate coefficients $\widetilde{w}_\ell$. This can be done by estimating the value of $\widetilde{w}_\ell$ as in standard supervised machine-learning approaches. In fact, this turns out to be a standard linear regression problem, since Eq. (4) can be interpreted as a linear model whose predictors are the Walsh function values. In particular, sparse techniques can be used to minimize the number of non-zero coefficients when the number of predictors is large [71]. Following [56, 57, 62], we use the Lasso algorithm [72] to fit the approximate model, which is of particular interest given that the number of Walsh functions of up to a given constant order $\mathsf{O}$ might be greater than the number of solutions used for training.

## 3. A Surrogate-assisted Evolutionary Multi-objective Framework based on Walsh Functions

In this section, we describe a surrogate-assisted multi-objective combinatorial optimization framework using the Walsh functions as surrogate (SMCO/W). The proposed framework integrates three main design components; (i) the inner optimizer of the substitute Walsh surrogate, (ii) the selection of the solution to be evaluated at each iteration, and (iii) the setting of the Walsh order at the training phase. These design components can be set interchangeably following different strategies which are described in a step-by-step detailed manner in the following.

### 3.1. General Description

The high-level pseudo-code of the proposed SMCO/W framework is depicted in the template of Algorithm 1. It follows the general computational flow of the (surrogate-less) MOEA/D algorithm, as described in Section 2.2, with a few exceptions. The original problem is decomposed into $\mu$ single-objective sub-problems using the Chebychev scalarizing function. A population $\mathcal{P}_\mu$ is initialized with $\mu$ randomly-generated solutions. Each solution is evaluated using the expensive objectives before being assigned to a unique sub-problem. This population is also used as the initial dataset $\mathcal{D}$ for training the surrogates. A generation of the algorithm (line 6) consists of iterating once over all the sub-problems. The algorithm stops when a given termination condition is satisfied, which renders the computing budget one can afford. In an expensive setting, where the evaluation function is assumed to be very time consuming, the budget is expressed as a maximum number of calls to the objective vector function. In our work, we consider a range of values to better render the anytime behavior of the considered algorithms. Furthermore, as will be detailed later in our experimental study, we shall discuss the implication of using the Walsh surrogate on the computational time.

At the beginning of every iteration $i$ dealing with a sub-problem $i \in [\![1, \mu]\!]$, a Walsh order $\mathsf{O}$ is chosen following the order selection component in line 7. The order selection component takes as input a *History* variable, which is an artifact indicating that some information about the search status may be used at this step. We then consider one surrogate for each of the $m$ objectives. These surrogates $\widetilde{\mathcal{F}} = (\widetilde{f}_1, \ldots, \widetilde{f}_m)$ are trained (line 8) using the (training) dataset $\mathcal{D}$, which contains *all* solutions evaluated so far. The estimates of the Walsh coefficients (Eq. (4)) for each surrogate $\widetilde{f}_i$ are computed following a sparse linear regression methodology, as discussed in Section 2.3.

In contrast to surrogate-less evolutionary algorithms, SMCO/W relies temporarily on the so-trained surrogates to intensively search for high-quality solutions. In other words, after the training step, a pool of offspring solutions $\mathcal{S}$ is generated by a *surrogate optimizer* (line 10), without calling the true objective functions. This component takes the current population as input, together with the current surrogates $\widetilde{\mathcal{F}}$ and a copy $z^{\star\star}$ of the reference point. It is important to use a copy of $z^\star$ in the surrogate optimizer because the reference point will likely be updated on the basis of the (unreliable) estimated objective values, whereas $z^\star$ is only updated according to the true objective values evaluated so far. The different design choices for this inner discrete surrogate optimization component are discussed in detail in Section 3.2.

The next step (line 11) selects one candidate solution from $\mathcal{S}$ in order to be evaluated using the expensive objectives $F$. We call this component the *selection strategy*, and we describe it in Section 3.3. It takes the pool of solutions $\mathcal{S}$ generated by the optimizer as input, together with the surrogate model $\widetilde{\mathcal{F}}$, the reference point $z^{\star\star}$ as updated by the optimizer, and the current weight vector $\omega_i$. As output, it returns the solution to be evaluated at the current iteration $i$. The weight vector $\omega_i$ given as input of the selection component is only an artifact indicating that this component may depend on iteration $i$.

**Algorithm 1:** Surrogate-assisted Multi-objective Combinatorial Optimization based on Walsh basis (SMCO/W)

**Input:** $\mathcal{W}_\mu := \{\omega^1, \ldots, \omega^\mu\}$: weight vectors; $\mathsf{g}(\cdot \mid \omega, \cdot)$: a scalarizing function to be minimized; $\mathsf{O}_{\max}$: maximum order of Walsh functions;

1 $\mathcal{P}_\mu \leftarrow \{x^1, \ldots, x^\mu\}$ : initial population of size $\mu$;
2 $\mathcal{D} \leftarrow \{(x^1, F(x^1)), \ldots, (x^\mu, F(x^\mu))\}$: training set;
3 $\mathcal{EP} \leftarrow$ initialize external archive (optional) ;
4 $z^\star \leftarrow$ initialize reference point;
5 **while** *the termination condition is not satisfied* **do**
6  **for** $i \in \{1, \ldots, \mu\}$ **do**
  `// Choose the Walsh order`
7  $\mathsf{O} \leftarrow \textsc{WalshOrder}(History, \mathsf{O}_{\max})$;
  `// Train Walsh models`
8  $\widetilde{\mathcal{F}} := (\widetilde{f}_1, \ldots, \widetilde{f}_m) \leftarrow \textsc{TrainWalsh}(\mathcal{D}, \mathsf{O})$;
  `// Copy the reference point`
9  $z^{\star\star} \leftarrow z^\star$ ;
  `// Optimize the Walsh surrogate`
10  $\mathcal{S} \leftarrow \textsc{RunOptimizer}(\mathcal{P}_\mu, \widetilde{\mathcal{F}}, z^{\star\star})$ ;
  `// Selection for true evaluation`
11  $x' \leftarrow \textsc{SelectEval}(\mathcal{S}, \omega_i, \widetilde{\mathcal{F}}, z^{\star\star})$ ;
12  $F(x') \leftarrow$ evaluate $x'$ ;
13  $\mathcal{EP} \leftarrow$ (optional) update external archive with $x'$ ;
14  $z^\star \leftarrow$ update reference point using $F(x')$ ;
  `// Replacement`
15  **for** $j \in \{1, \ldots, \mu\}$ **do**
16  **if** $g(x'|\omega^j, F) < g(x^j|\omega^j, F)$ **then**
17  $x^j \leftarrow x'$ ;
  `// Update training data`
18  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x', F(x'))\}$;

After the selection and the evaluation of the offspring solution $x'$, we follow the standard process of MOEA/D, with the update of the external archive $\mathcal{EP}$, the update of the reference point $z^\star$, and the replacement of the population. At this step, we compare the scalarized fitness value of the candidate solution $x'$ with the solutions from the current population $\mathcal{P}_\mu$, according to their corresponding weight vectors. The candidate solution $x'$ may enter the population if any improvement is found as discussed previously in Section 2.2. Finally, the training dataset $\mathcal{D}$ is updated with the newly evaluated offspring $x'$ (line 18).

To summarize, SMCO/W is based on three generic components that can be configured in different manners. In the next paragraphs, we propose different possible strategies.

**Algorithm 2:** Multiple Local Search (MLS)

**Input:** $\widetilde{\mathcal{F}} := (\widetilde{f}_1, \ldots, \widetilde{f}_m)$: surrogates; $\mathcal{W}_\mu := \{w^1, \ldots, w^\mu\}$: weight vectors; $\mathsf{g}(\cdot \mid \omega, \cdot)$: Chebychev scalarizing function; $\mathcal{N} : X \leftarrow 2^X$: a neighborhood relation;

```
S ← ∅ ;
for ω^i ∈ W_μ do
    LocalOptimum ← False;
    x* ← random (initial) solution;
    while ! LocalOptimum do
        x'* ← x*;
        foreach x' ∈ N(x*) do
            if g(x'|ω^i, F̃) < g(x'*|ω^i, F̃) then x'* ← x'; ;
        if g(x'*|ω^i, F̃) < g(x*|ω^i, F̃) then
            x* ← x'*;
        else
            LocalOptimum ← True;
    S ← S ∪ {x*} ;
return S
```

### *3.2. Component #1: Surrogate Optimizer*

The choice of an accurate surrogate optimizer is important since it allows us to search for high-quality solutions according to the model $\widetilde{\mathcal{F}}$ without ever calling the true evaluation functions. We investigate three alternative optimizers using seemingly different search paradigms. The considered optimizers are summarized in the following:

**Optim$_{\text{MOEA/D}}$** performs the conventional MOEA/D in order to identify a Pareto set approximation with respect to the surrogates $\widetilde{\mathcal{F}}$. It is initialized with the current population from the main SMCO/W algorithm, and returns the evolved population after a number of generations. Using MOEA/D as an optimizer was recommended in [68]; however, we shall show that it can be substantially out-performed by the other proposed techniques.

**Optim$_{\text{MLS}}$** (Algorithm 2) independently runs multiple local search, one for each defined sub-problem. Given the surrogates $\widetilde{\mathcal{F}}$, a standard single-objective hill-climber is performed for each sub-problem defined by the weight vector $\omega^i$, $i \in [\![1, \mu]\!]$. Each hill-climber is initialized with a random solution, and iteratively selects the best improving solution, if any, using a standard 1-bit-flip neighborhood, until the search stops in a local optimum with respect to the aggregation function $\mathsf{g}(\cdot \mid \omega^i, \widetilde{\mathcal{F}})$. Optim$_{\text{MLS}}$ then returns a pool of $\mu$ (local optimal) solutions, one per sub-problem.

**Optim$_{\text{PLS}}$** (Algorithm 3) is based on the so-called Pareto Local Search [73] algorithm. It maintains

**Algorithm 3:** Pareto Local Search (PLS)

**Input:** $\widetilde{\mathcal{F}} := (\widetilde{f}_1, \ldots, \widetilde{f}_m)$: surrogates; $\mathcal{N} : X \leftarrow 2^X$: a neighborhood relation;

```
1  x ← random (intial) solution;
2  S ← {x} ;                                          // Non dominated archive
3  R ← {x} ;                                          // Non visited solutions
4  while R ≠ ∅ do
5      x' ← select a solution at random in R;
6      foreach x'' ∈ N(x') do
7          if x'' is not dominated by any solution in S then
8              for x ∈ S do
9                  if x is dominated by x'' then S ← S \ {x};
10             S ← S ∪ {x''} ;
11             for x ∈ R do
12                 if x is dominated by x'' then R ← R \ {x};
13             R ← R ∪ {x''};
14     R ← R \ {x'};
15 return S
```

an unbounded archive of mutually non-dominated solutions, initialized with a random solution. At each step, one solution is selected at random from the archive, all its neighbors are evaluated with respect to $\widetilde{\mathcal{F}}$, and the archive is updated accordingly. The current solution is tagged as visited in order to avoid a useless re-exploration of its neighborhood. The search process stops once all solutions in the archive are visited. The content of the archive corresponds to the pool of solutions returned by $\mathsf{Optim}_{\mathsf{PLS}}$.

### *3.3. Component #2: Selection Strategy*

The selection of the solution among the whole pool returned by the surrogate optimizer is a critically important component (line 11 in Algorithm 1). Let us consider some iteration $i \in [\![1, \mu]\!]$, which can be thought as corresponding to a particular sub-problem $\omega^i$. An intuitive strategy is to attempt to improve sub-problems in a round-robin manner, and thus to select the solution whose estimated scalar value (computed on the basis of the surrogate) is the best for the current sub-problem. This was actually implemented in [68]. However, we shall show that this local selection strategy can be sub-optimal, and is out-performed by other carefully designed global strategies. More specifically, for an iteration $i \in [\![i, \mu]\!]$, we study the following four selection strategies:

$\mathsf{Select}_{\mathsf{local}}$ chooses the solution $x'$ with the best aggregation value for the sub-problem considered at the current iteration and correspondingly to the weight vector $\omega^i$.

$$x' := \mathrm{argmin}_{x \in \mathcal{S}}\, \mathsf{g}(x \mid \omega^i, \widetilde{\mathcal{F}})$$

**Select$_{\text{global}}$** chooses the solution $x'$ with the best aggregation value with respect to any sub-problem $\ell \in [\![1, \mu]\!]$, independently of the current iteration:

$$x' := \text{argmin}_{x \in \mathcal{S}} \min_{1 \leq \ell \leq \mu} \mathsf{g}(x \mid \omega^\ell, \widetilde{\mathcal{F}})$$

**Select$_{\text{BI}}$** (best improvement) chooses the solution $x'$ that improves the most the aggregation value of any solution in the population, independently of the current iteration:

$$x' := \text{argmin}_{x \in \mathcal{S}} \min_{1 \leq \ell \leq \mu} \mathsf{g}(x_\ell \mid \omega^\ell, \widetilde{\mathcal{F}}) - \mathsf{g}(x \mid \omega^\ell, \widetilde{\mathcal{F}})$$

**Select$_{\text{NBI}}$** (best normalized improvement) is based on the previous strategy, but the improvement value is normalized by the actual aggregation value of the current solution associated with each sub-problem:

$$x' := \text{argmin}_{x \in \mathcal{S}} \min_{1 \leq \ell \leq \mu} \frac{\mathsf{g}(x_\ell \mid \omega^\ell, \widetilde{\mathcal{F}}) - \mathsf{g}(x \mid \omega^\ell, \widetilde{\mathcal{F}})}{\mathsf{g}(x_\ell \mid \omega^\ell, \widetilde{\mathcal{F}})}$$

### *3.4. Component #3: Walsh Order*

As discussed in Section 2.3, a Walsh surrogate requires an order up to which the coefficients are expanded. The choice of the Walsh order is an important and difficult task for black-box problems, which was not deeply addressed in the past. This choice infers the number of coefficients in the surrogate model (Eq.(4)). This does not only impact the model theoretical accuracy. It also impacts the fitting process itself, because the model complexity increases with the number of coefficients. Let us illustrate this for a problem of size $n = 50$. In such a case, there are 51 coefficients for an order $\mathsf{O} = 1$, 1 276 coefficients for an order $\mathsf{O} = 2$, and 20 876 coefficients for an order $\mathsf{O} = 3$. A low number of coefficients may be insufficient to accurately approximate the objectives. A high number of coefficients eventually improves the approximation quality, but a larger dataset should then be used in order to accurately train the model. Hence, this may seem contradictory with the fact that only a restricted number of evaluated solutions can be used for training.

In this paper, we assume that the order used to train the model is bounded by a constant value $\mathsf{O}_{\max}$. This is a reasonable assumption in practice, since otherwise the number of Walsh coefficients would grow exponentially. We also remark that such an assumption is in line with the fact that many challenging combinatorial optimization problems can be decomposed in an exact manner using Walsh functions of bounded order. However, it remains unclear which concrete order to use within the range $[\![1, \mathsf{O}_{\max}]\!]$. Consequently, we investigate three simple strategies for setting the Walsh order, as described below.

**Order$_{\text{static}}$** corresponds to a baseline strategy for setting the Walsh order. The order is simply a static user-defined parameter, i.e., the Walsh surrogate is fitted considering a fixed order value

$\mathsf{O} \in [\![1, \mathsf{O}_{\max}]\!]$, where $\mathsf{O}$ is assumed to be a parameter provided by the end-user.

**Order$_{\text{random}}$** is a basic dynamic strategy where the order is set randomly every time the model is trained. In other words, at each iteration, the value of the Walsh order is picked uniformly at random within the range $[\![1, \mathsf{O}_{\max}]\!]$. The advantage of this strategy is to use all admissible orders without being completely limited by an improper static setting. This simple strategy allows us to illustrate the benefits of changing dynamically the order value, in comparison to a static setting.

**Order$_{\text{greedy}}$** dynamically adjusts the Walsh order during the search process, depending on the data collected so far. At the end of each iteration $i \in [\![1, \mu]\!]$, we compute the number of sub-problems that are improved (in line 16). Let us denote by $p_j$ the number of improved sub-problems, where $j$ corresponds to the $j^{\text{th}}$ true function evaluation. We thereby track the number of improved sub-problems over a window of size $t$ (last) iterations, where $t$ is a user-defined parameter. At the beginning of each iteration, we compute the average, over the window, of the number of improved sub-problems, denoted $\overline{p}_t$. If $\overline{p}_t < 1$, then we interpret this as a signal for changing the Walsh order. This is perfomed according to a particular schedule, as discussed in the following. Let $\mathsf{O}_{\text{curr}}$ be the order used at a current iteration $i \in [\![1, \mu]\!]$. Let $\mathsf{O}_{\text{prev}}$ be the order used at the previous iteration (i.e., iteration $i - 1$ or $\mu$). At the beginning of the next iteration, let us assume that $\overline{p}_t < 1$. Then, the new order $\mathsf{O}_{\text{new}}$ returned by the selection strategy is given by $\mathsf{O}_{\text{new}} = \mathsf{O}_{\text{curr}} + 1$ if either $\mathsf{O}_{\text{curr}} = 1$ or $\mathsf{O}_{\text{prev}} < \mathsf{O}_{\text{curr}} < \mathsf{O}_{\max}$; otherwise, we set $\mathsf{O}_{\text{new}} = \mathsf{O}_{\text{curr}} - 1$.

The idea behind this greedy dynamic strategy is to start the search with the smallest order 1. If we observe that it allows the current population to improve ($\overline{p}_t \geq 1$), then we keep using the same successful order. Otherwise, we increase the order, and hence the model complexity, in the hope of obtaining a more accurate surrogate. However, when attaining the maximum allowed model complexity $\mathsf{O}_{\max}$, we continue scanning the possible order range by decreasing the order value, and increasing it again if an order of 1 is reached. The current order value stops changing whenever the model is accurate enough to improve the population, and the whole dynamic schedule is repeated as soon as no more improvements are observed.

## 4. Experimental Setup

In the remainder of the paper, we report a comprehensive analysis of the combined effect of the previously described components. Obviously, extensive experiments with real-world expensive black-box MCOPs would be computationally infeasible. Besides, our primary goal is not to tackle an application-specific problem, but to conduct a general-purpose informative study providing generic guidelines for designing a discrete surrogate-assisted approach. For this purpose, we consider two classes of well-established MCOP benchmarks. This is described below together with the experimented algorithm variants.

### 4.1. Benchmark Problems

We consider bi-objective NK-landscapes [74, 75] and bi-objective unconstrained binary quadratic programming problems (UBQP) [76], as a set of challenging and representative pseudo-boolean MCOPs. In fact, NK-landscapes allows one to control the degree of non-linearity of the problem. We hence consider instances with a variable degree of difficulty. UBQP is a computationally challenging problem which is known to embrace a remarkable range of applications in combinatorial optimization [77].

#### 4.1.1. Multi-objective NK-Landscapes (MNK-Landscapes)

Solutions are binary strings of size $n$ and the objective vector, to be maximized, is $F = (f_1, \ldots, f_m) \colon \{0,1\}^n \mapsto [0,1]^m$. Each objective function is defined by [78]:

$$f_i(x) = \frac{1}{n} \sum_{j=1}^{n} c_j^i(x_j, x_{j_1}, \ldots, x_{j_k})$$

where the $c_j^i : \{0,1\}^{k+1} \to [0,1)$ are component functions, and $k$ is a parameter specifying the number of epistatic interactions. The component function $c_j^i$ assigns a real-valued contribution for every combination of $x_j$ and its $k$ epistatic interactions $\{x_{j_1}, \ldots, x_{j_k}\}$. The parameter $k$ defines the degree of non-linearity of the problem, and hence the *ruggedness* of the landscape.

We consider bi-objective MNK-landscapes [75] with the following setting: $n \in \{25, 50\}$ and $k \in \{0, 1, 2\}$. In line with previous studies [57, 68], this allows us to span instances ranging from a small to a relatively large number of variables, and following a linear ($k = 0$), quadratic ($k = 1$) and cubic ($k = 2$) variable interaction scheme.

#### 4.1.2. Multi-objective UBQP (MUBQP)

Given a collection of $n$ items such that each pair of items is associated with multidimensional profit values, the multi-objective unconstrained binary quadratic programming (MUBQP) problem seeks a subset of items that maximizes the sum of their paired values in each dimension [76]. The value of a pair is summed up only if the two corresponding items are selected. A solution can be represented as a binary string of size $n$. Each position from the binary string maps to a particular variable that indicates whether the corresponding item is included in the subset of selected items or not. Given a solution $x = (x_1, \ldots, x_n)$, each objective $f_k$, $k \in [\![1, m]\!]$, is defined as follows:

$$f_k(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}^k x_i x_j$$

where $Q_k = q_k^{ij}$ is an $n \times n$ matrix of constant values, either positive, negative or zero, $k \in [\![1, m]\!]$. We consider bi-objective UBQP instances with $n \in \{25, 50\}$ variables and a density of 0.9 for non-zero entries in the matrix $Q_k$ to infer difficult and highly non-linear search landscapes.

Table 1: Summary of the considered SMCO/W components.

| Component | Proposed configuration values | analyzed in Sec. |
|---|---|---|
| **Optimizer** | $\mathsf{Optim}_{\mathsf{MOEA/D}}$ [68], $\mathsf{Optim}_{\mathsf{MLS}}$, $\mathsf{Optim}_{\mathsf{PLS}}$ | 5.1 & 5.2 |
| **Selection** | $\mathsf{Select}_{\mathsf{local}}$ [68], $\mathsf{Select}_{\mathsf{global}}$, $\mathsf{Select}_{\mathsf{BI}}$, $\mathsf{Select}_{\mathsf{NBI}}$ | 5.1 |
| **Order** | $\mathsf{Order}_{\mathsf{static}}$ [68] ($\mathsf{O} \in \{1, 2, \mathsf{O}_{\max} = 3\}$), $\mathsf{Order}_{\mathsf{random}}$, $\mathsf{Order}_{\mathsf{greedy}}$ | 5.3 |

### *4.2. Experimental set-up*

In order to fully analyze the impact of the proposed strategies, we systematically evaluate the performance of *all* possible combinations. Furthermore, we consider to evaluate the gain in approximation quality in comparison to an evolutionary solving process without any surrogate. For this purpose, we consider the three multi-objective algorithms used to optimize the learned problem model as baselines. In other words, we shall analyze the search performance obtained with MOEA/D, MLS and PLS as a main optimization procedure for the original expensive problem.

For all experimented algorithms, we use 50 weight vectors such that $\omega^i = \left(\frac{i-1}{\mu-1}, \frac{1-(i-1)}{\mu-1}\right)$, $i \in [\![1, 50]\!]$. For Walsh surrogates, a Lasso regression is used to fit the model [72]. The maximum value allowed for the Walsh order is set to $\mathsf{O}_{\max} = 3$ which corresponds to a cubic model. When using the $\mathsf{Order}_{\mathsf{static}}$ strategy, the actual order $d$ used for training is a user-defined parameter, which we set in the range $\mathsf{O} \in \{1, 2, 3 = \mathsf{O}_{\max}\}$, i.e., three $\mathsf{O}$-values are experimented for each variant using $\mathsf{Order}_{\mathsf{static}}$. The greedy strategy to adapt the Walsh order $\mathsf{Order}_{\mathsf{greedy}}$ uses a window of size $t = 5$. Besides, each surrogate optimizer comes with different parameters. $\mathsf{Optim}_{\mathsf{MOEA/D}}$ is executed for 10 generations, as in [68]. It uses a one-point crossover followed by a uniform bit-flip mutation with a rate of $1/n$. The solution neighborhood considered in $\mathsf{Optim}_{\mathsf{MLS}}$ and $\mathsf{Optim}_{\mathsf{PLS}}$ is based on the bit-flip operator; i.e., two solutions are neighbors if their Hamming distance is 1. Finally, $\mathsf{Optim}_{\mathsf{MLS}}$ uses the same 50 weight vectors as defined previously.

Overall, we experiment 60 possible configurations (See Table 1 for a summary): 3 surrogate optimizers $\times$ 4 selection strategies $\times$ $(3 + 2)$ Walsh order settings, together with 3 additional surrogate-less algorithms. Each algorithm is independently executed 10 times on the 8 considered MCOP instances ($2 \times 3$ MNK-landscapes + 2 MUBQP), for a maximum budget of 1 500 (expensive) evaluations and not exceeding 64 CPU hours per run. The experiments were conducted in Python 3.7 on an Intel Core Xeon E5-2630 (2.20 GHZ, 256 GB RAM) under Debian 10.

For performance assessment, we use the additive epsilon indicator [79]. Notice, however, that our results were found to be consistent when using the hypervolume indicator. This is not reported explicitly to avoid flooding the reader with too much data. The epsilon indicator gives the minimum factor by which an approximation set has to be translated in the objective space in order to (weakly) dominate a reference set. The lower the indicator, the better the approximation. The reference set is constructed by merging the solutions found over all runs and configurations for a given instance, and removing dominated ones. For a given run, the archive of all non-dominated solutions found during the search process is used to measure approximation quality. Finally, we shall report the

approximation quality for different intermediate budgets, so as to render the relative anytime performance.

## 5. Experimental Analysis

In this section, we report our analysis and empirical findings. Since we consider an extensive number of configurations, a main challenge is to fairly address the impact of the combined effect of the designed components on search performance. In fact, as our analysis will reveal, we found that there is a non-trivial interaction between the different components, which we report in a step-by-step manner. In Section 5.1, we start by studying the interaction between the surrogate optimizer (Component #1) and the selection strategy (Component #2). In Section 5.2, we analyze the relative performance obtained when the surrogate optimizer is combined with the corresponding best selection strategy. In Section 5.3, we study the impact of the Walsh order setting (Component #3). In Section 5.4, we report the performance gain against surrogate-less approaches.

### *5.1. Impact of Selection on Surrogate Optimizers*

In the following, we address the relative impact of the selection strategy (Component #2) on each *individual* SMCO/W variant obtained with a particular surrogate optimizer (Component #1). For MNK-landscapes, this is summarized in Tables 2, 3, and 4, respectively when using $\mathsf{Optim}_{\mathsf{MOEA/D}}$, $\mathsf{Optim}_{\mathsf{MLS}}$, and $\mathsf{Optim}_{\mathsf{PLS}}$. For MUBQP, this is summarized in Table 5 for all optimizers. The tables show the relative rank of each selection strategy and each *static* setting of the Walsh order. A rank $c$ indicates that the corresponding algorithm is significantly outperformed by $c$ other strategies using a Wilcoxon test with a Bonferroni correction at a significance level of 0.05. Ranks in bold correspond to approaches that are not significantly outperformed by any other. Different budgets, ranging from a small to a medium and a relatively high number of true evaluations, are considered. This is to better render the *anytime* behavior. We also show the average epsilon indicator value obtained for each variant (in parentheses in the tables), which is to provide the reader with a preliminary idea of the overall relative performance (underlined values in the tables show the best averages). Let us comment that the ranks shown in Tables 2, 3, 4, and 5, still do not allow to elicit in a comprehensive manner what is the best possible configuration. A more specific analysis is to follow in the subsequent sections, after analyzing the relative impact of the selection strategy when combined with a particular optimizer. In fact, from these tables, we can for now draw two main general observations.

Firstly, the relative rank obtained by a given selection strategy is overall consistent over the different considered instances, although for MNK-landscapes, the relative performance is a function of the value of $k$, which is the degree of non-linearity/ruggedness of the landscape. This is also related to the Walsh order setting. We can observe that the Walsh order $\mathsf{O}$ providing the best rank depends consistently on the value of $k$, such that $\mathsf{O} = k + 1$. For MUBQP instances, and since

Table 2: Rank and average epsilon indicator value ($\times 10^2$, between brackets) after 200, 700, and 1 500 evaluations for MNK-landscapes. The results are w.r.t. $\mathsf{Optim}_{\mathsf{MOEA/D}}$ and $\mathsf{Order}_{\mathsf{static}}$.

| | | | | $\mathsf{Select}_{\mathsf{local}}$ | | | $\mathsf{Select}_{\mathsf{global}}$ | | | $\mathsf{Select}_{\mathsf{BI}}$ | | | $\mathsf{Select}_{\mathsf{NBI}}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #eval | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 |
| $\mathsf{Optim}_{\mathsf{MOEA/D}}$ | $k=0$ | $n=25$ | 200 | $\mathbf{0}_{(1.4)}$ | $\mathbf{0}_{(1.8)}$ | $3_{(2.4)}$ | $9_{(5)}$ | $9_{(5.4)}$ | $9_{(5.7)}$ | $\mathbf{0}_{(1)}$ | $\mathbf{0}_{(1.2)}$ | $\mathbf{0}_{(1.8)}$ | $\mathbf{0}_{(1.3)}$ | $\mathbf{0}_{(1.4)}$ | $3_{(2.4)}$ |
| | | | 700 | $\mathbf{0}_{(1)}$ | $\mathbf{0}_{(1.1)}$ | $2_{(1.3)}$ | $9_{(4.4)}$ | $9_{(4.1)}$ | $9_{(4.1)}$ | $\mathbf{0}_{(0.7)}$ | $\mathbf{0}_{(0.7)}$ | $\mathbf{0}_{(0.9)}$ | $\mathbf{0}_{(1.1)}$ | $\mathbf{0}_{(1)}$ | $\mathbf{0}_{(1.1)}$ |
| | | | 1 500 | $\mathbf{0}_{(0.8)}$ | $\mathbf{0}_{(0.8)}$ | $\mathbf{0}_{(0.9)}$ | $9_{(3.9)}$ | $9_{(3.6)}$ | $9_{(3.2)}$ | $\mathbf{0}_{(0.6)}$ | $\mathbf{0}_{(0.6)}$ | $\mathbf{0}_{(0.7)}$ | $\mathbf{0}_{(0.8)}$ | $\mathbf{0}_{(0.9)}$ | $\mathbf{0}_{(0.9)}$ |
| | | $n=50$ | 200 | $\mathbf{0}_{(2)}$ | $2_{(3.7)}$ | $2_{(5)}$ | $2_{(7.9)}$ | $6_{(8.6)}$ | $6_{(9.2)}$ | $\mathbf{0}_{(2.3)}$ | $1_{(3.6)}$ | $2_{(4.7)}$ | $6_{(8.6)}$ | $6_{(9.1)}$ | $6_{(9.1)}$ |
| | | | 700 | $\mathbf{0}_{(1.4)}$ | $\mathbf{0}_{(1.8)}$ | $1_{(2.3)}$ | $5_{(6.5)}$ | $6_{(6.3)}$ | $6_{(8)}$ | $\mathbf{0}_{(2)}$ | $1_{(2.3)}$ | $4_{(3)}$ | $6_{(8.3)}$ | $6_{(8.4)}$ | $6_{(8.5)}$ |
| | | | 1 500 | $\mathbf{0}_{(1.1)}$ | $\mathbf{0}_{(1.3)}$ | $\mathbf{0}_{(1.4)}$ | $6_{(3.9)}$ | $5_{(4.2)}$ | $6_{(4.9)}$ | $\mathbf{0}_{(1.7)}$ | $1_{(1.9)}$ | $3_{(2.4)}$ | $8_{(7.6)}$ | $8_{(7.6)}$ | $8_{(7.7)}$ |
| | | **avg. rank** | | **0.56** | | | 7.17 | | | 0.67 | | | 3.5 | | |
| | $k=1$ | $n=25$ | 200 | $3_{(4.2)}$ | $\mathbf{0}_{(1.5)}$ | $\mathbf{0}_{(2.6)}$ | $8_{(8.7)}$ | $6_{(6.8)}$ | $8_{(7.7)}$ | $3_{(4.5)}$ | $\mathbf{0}_{(1.1)}$ | $\mathbf{0}_{(2.5)}$ | $3_{(4.2)}$ | $8_{(7.2)}$ | $\mathbf{0}_{(1.7)}$ |
| | | | 700 | $5_{(3.4)}$ | $\mathbf{0}_{(0.5)}$ | $\mathbf{0}_{(0.5)}$ | $7_{(6.6)}$ | $6_{(5.7)}$ | $7_{(6.2)}$ | $5_{(3.8)}$ | $\mathbf{0}_{(0.5)}$ | $\mathbf{0}_{(0.6)}$ | $5_{(2.8)}$ | $7_{(6)}$ | $\mathbf{0}_{(0.6)}$ |
| | | | 1 500 | $5_{(2.9)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.4)}$ | $7_{(5.7)}$ | $5_{(4.4)}$ | $6_{(4.8)}$ | $5_{(3.5)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.4)}$ | $5_{(2.6)}$ | $7_{(5.7)}$ | $\mathbf{0}_{(0.4)}$ |
| | | $n=50$ | 200 | $\mathbf{0}_{(5.2)}$ | $\mathbf{0}_{(4.8)}$ | $1_{(6.3)}$ | $6_{(9.7)}$ | $6_{(10.6)}$ | $6_{(10.9)}$ | $\mathbf{0}_{(5.8)}$ | $\mathbf{0}_{(4.8)}$ | $\mathbf{0}_{(5.6)}$ | $6_{(9.8)}$ | $6_{(10.7)}$ | $6_{(10.9)}$ |
| | | | 700 | $1_{(4.1)}$ | $\mathbf{0}_{(3.2)}$ | $\mathbf{0}_{(3.7)}$ | $6_{(7.7)}$ | $5_{(7.7)}$ | $6_{(9.1)}$ | $1_{(4.4)}$ | $\mathbf{0}_{(2.8)}$ | $\mathbf{0}_{(3.4)}$ | $6_{(8.7)}$ | $6_{(9.6)}$ | $7_{(9.9)}$ |
| | | | 1 500 | $1_{(3.8)}$ | $\mathbf{0}_{(2.9)}$ | $\mathbf{0}_{(3.1)}$ | $4_{(6.5)}$ | $\mathbf{0}_{(5.6)}$ | $3_{(6)}$ | $\mathbf{0}_{(3.9)}$ | $\mathbf{0}_{(2.8)}$ | $\mathbf{0}_{(2.8)}$ | $6_{(8.3)}$ | $6_{(8.6)}$ | $6_{(8.7)}$ |
| | | **avg. rank** | | 0.89 | | | 5.67 | | | **0.78** | | | 5 | | |
| | $k=2$ | $n=25$ | 200 | $4_{(8)}$ | $\mathbf{0}_{(5.6)}$ | $\mathbf{0}_{(5.9)}$ | $6_{(9)}$ | $6_{(8.3)}$ | $4_{(8.4)}$ | $2_{(7.5)}$ | $\mathbf{0}_{(4.3)}$ | $\mathbf{0}_{(4.7)}$ | $5_{(8.2)}$ | $\mathbf{0}_{(4.7)}$ | $\mathbf{0}_{(5.2)}$ |
| | | | 700 | $6_{(6.5)}$ | $\mathbf{0}_{(3.9)}$ | $\mathbf{0}_{(3)}$ | $6_{(7.7)}$ | $1_{(5.4)}$ | $\mathbf{0}_{(6.2)}$ | $6_{(7.2)}$ | $\mathbf{0}_{(2.9)}$ | $\mathbf{0}_{(2.7)}$ | $5_{(6.8)}$ | $\mathbf{0}_{(3.6)}$ | $\mathbf{0}_{(3.2)}$ |
| | | | 1 500 | $6_{(6.2)}$ | $\mathbf{0}_{(3.4)}$ | $\mathbf{0}_{(2)}$ | $6_{(7.2)}$ | $\mathbf{0}_{(5)}$ | $\mathbf{0}_{(5.3)}$ | $5_{(5.9)}$ | $\mathbf{0}_{(2.4)}$ | $\mathbf{0}_{(2.1)}$ | $5_{(6.1)}$ | $\mathbf{0}_{(2.8)}$ | $\mathbf{0}_{(2.7)}$ |
| | | $n=50$ | 200 | $\mathbf{0}_{(8)}$ | $\mathbf{0}_{(7.7)}$ | $\mathbf{0}_{(7.7)}$ | $5_{(13.3)}$ | $5_{(11.6)}$ | $5_{(12.3)}$ | $\mathbf{0}_{(9.4)}$ | $\mathbf{0}_{(6.5)}$ | $\mathbf{0}_{(7.3)}$ | $5_{(13.3)}$ | $5_{(11.6)}$ | $5_{(12.3)}$ |
| | | | 700 | $\mathbf{0}_{(5.7)}$ | $\mathbf{0}_{(4.4)}$ | $\mathbf{0}_{(4.6)}$ | $6_{(11.2)}$ | $5_{(9.6)}$ | $6_{(10.6)}$ | $\mathbf{0}_{(6.3)}$ | $\mathbf{0}_{(4.6)}$ | $\mathbf{0}_{(4.4)}$ | $6_{(11.4)}$ | $6_{(10.6)}$ | $6_{(11.8)}$ |
| | | | 1 500 | $1_{(5.4)}$ | $\mathbf{0}_{(3.4)}$ | $\mathbf{0}_{(3.3)}$ | $6_{(10.7)}$ | $4_{(7.8)}$ | $4_{(8.3)}$ | $\mathbf{0}_{(5.3)}$ | $\mathbf{0}_{(4.1)}$ | $\mathbf{0}_{(3.7)}$ | $6_{(10.9)}$ | $6_{(9.6)}$ | $6_{(10.5)}$ |
| | | **avg. rank** | | 0.94 | | | 4.17 | | | **0.72** | | | 3.67 | | |

Table 3: Rank and average epsilon indicator value ($\times 10^2$, between brackets) after 200, 700, and 1 500 evaluations for MNK-landscapes. The results are w.r.t. $\mathsf{Optim}_{\mathsf{MLS}}$ and $\mathsf{Order}_{\mathsf{static}}$.

| | | | | $\mathsf{Select}_{\mathsf{local}}$ | | | $\mathsf{Select}_{\mathsf{global}}$ | | | $\mathsf{Select}_{\mathsf{BI}}$ | | | $\mathsf{Select}_{\mathsf{NBI}}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #eval | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 |
| $\mathsf{Optim}_{\mathsf{MLS}}$ | $k=0$ | $n=25$ | 200 | $5_{(4.7)}$ | $3_{(3.3)}$ | $2_{(2.4)}$ | $3_{(2.2)}$ | $4_{(4.7)}$ | $7_{(5.2)}$ | $5_{(3.7)}$ | $3_{(3.2)}$ | $3_{(2.8)}$ | $\mathbf{0}_{(1.1)}$ | $\mathbf{0}_{(0.4)}$ | $1_{(1)}$ |
| | | | 700 | $4_{(3.1)}$ | $3_{(2.4)}$ | $3_{(1.5)}$ | $3_{(2.1)}$ | $8_{(4.1)}$ | $8_{(4.7)}$ | $3_{(2.9)}$ | $3_{(2.3)}$ | $2_{(1.8)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.3)}$ |
| | | | 1 500 | $3_{(2.3)}$ | $3_{(1.7)}$ | $\mathbf{0}_{(1.1)}$ | $3_{(2.1)}$ | $10_{(3.9)}$ | $10_{(4)}$ | $3_{(2.2)}$ | $3_{(1.7)}$ | $2_{(1.3)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.3)}$ |
| | | $n=50$ | 200 | $3_{(5.6)}$ | $1_{(4.6)}$ | $1_{(4.9)}$ | $1_{(4.1)}$ | $5_{(6.8)}$ | $10_{(7.4)}$ | $1_{(3.7)}$ | $1_{(4.1)}$ | $2_{(5.4)}$ | $1_{(3.9)}$ | $\mathbf{0}_{(1.7)}$ | $1_{(4.5)}$ |
| | | | 700 | $9_{(4.3)}$ | $1_{(1.2)}$ | $3_{(1.2)}$ | $4_{(1.6)}$ | $10_{(6.2)}$ | $10_{(6.4)}$ | $5_{(1.9)}$ | $\mathbf{0}_{(0.6)}$ | $2_{(1)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.5)}$ |
| | | | 1 500 | $9_{(3.3)}$ | $1_{(0.5)}$ | $\mathbf{0}_{(0.3)}$ | $8_{(1.6)}$ | $10_{(5.9)}$ | $10_{(6.2)}$ | $6_{(0.8)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.2)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.3)}$ |
| | | **avg. rank** | | 3 | | | 6.89 | | | 2.44 | | | **0.17** | | |
| | $k=1$ | $n=25$ | 200 | $2_{(3.9)}$ | $1_{(2.7)}$ | $1_{(2.8)}$ | $6_{(6.3)}$ | $4_{(5.4)}$ | $6_{(6.3)}$ | $3_{(4.8)}$ | $1_{(1.5)}$ | $1_{(3.2)}$ | $4_{(4.7)}$ | $\mathbf{0}_{(0.6)}$ | $1_{(1.7)}$ |
| | | | 700 | $6_{(2.9)}$ | $3_{(1.2)}$ | $1_{(0.6)}$ | $9_{(5.5)}$ | $7_{(4.8)}$ | $8_{(5.2)}$ | $6_{(3.9)}$ | $2_{(1)}$ | $1_{(0.9)}$ | $6_{(3.8)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.3)}$ |
| | | | 1 500 | $6_{(2.6)}$ | $3_{(0.9)}$ | $1_{(0.5)}$ | $9_{(5.1)}$ | $7_{(4.4)}$ | $9_{(4.9)}$ | $6_{(3.7)}$ | $2_{(0.8)}$ | $1_{(0.8)}$ | $6_{(3.5)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.3)}$ |
| | | $n=50$ | 200 | $1_{(4.7)}$ | $2_{(5.9)}$ | $1_{(6.2)}$ | $1_{(5.2)}$ | $5_{(6.7)}$ | $8_{(8.2)}$ | $\mathbf{0}_{(4.3)}$ | $1_{(5.3)}$ | $3_{(6.1)}$ | $\mathbf{0}_{(4.4)}$ | $\mathbf{0}_{(3.2)}$ | $\mathbf{0}_{(4.5)}$ |
| | | | 700 | $5_{(3.3)}$ | $1_{(1.6)}$ | $1_{(2.4)}$ | $8_{(5.1)}$ | $9_{(6.5)}$ | $9_{(6.6)}$ | $5_{(3.2)}$ | $\mathbf{0}_{(0.8)}$ | $2_{(1.4)}$ | $5_{(3.5)}$ | $\mathbf{0}_{(0.4)}$ | $1_{(1.3)}$ |
| | | | 1 500 | $6_{(2.9)}$ | $4_{(1)}$ | $\mathbf{0}_{(0.6)}$ | $9_{(5)}$ | $9_{(6.4)}$ | $9_{(6.5)}$ | $6_{(3.2)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.4)}$ | $6_{(3.4)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.4)}$ |
| | | **avg. rank** | | 2.5 | | | 7.33 | | | 2.22 | | | **1.61** | | |
| | $k=2$ | $n=25$ | 200 | $\mathbf{0}_{(7.4)}$ | $\mathbf{0}_{(6.4)}$ | $\mathbf{0}_{(5.9)}$ | $4_{(8.7)}$ | $\mathbf{0}_{(5)}$ | $1_{(7.1)}$ | $1_{(8.5)}$ | $\mathbf{0}_{(5.7)}$ | $\mathbf{0}_{(6.1)}$ | $1_{(8.3)}$ | $\mathbf{0}_{(5.3)}$ | $\mathbf{0}_{(5.5)}$ |
| | | | 700 | $4_{(5.6)}$ | $3_{(2.8)}$ | $\mathbf{0}_{(1.1)}$ | $8_{(7.4)}$ | $3_{(3.6)}$ | $4_{(4.8)}$ | $7_{(7.1)}$ | $2_{(2.9)}$ | $\mathbf{0}_{(0.9)}$ | $7_{(7.3)}$ | $2_{(2.5)}$ | $\mathbf{0}_{(0.5)}$ |
| | | | 1 500 | $6_{(5.1)}$ | $3_{(2)}$ | $\mathbf{0}_{(0.7)}$ | $8_{(7.1)}$ | $6_{(3.5)}$ | $6_{(4.6)}$ | $7_{(6.6)}$ | $3_{(2.1)}$ | $\mathbf{0}_{(0.5)}$ | $8_{(6.9)}$ | $3_{(2)}$ | $\mathbf{0}_{(0.5)}$ |
| | | $n=50$ | 200 | $\mathbf{0}_{(7)}$ | $\mathbf{0}_{(7.7)}$ | $\mathbf{0}_{(8.4)}$ | $8_{(10.7)}$ | $\mathbf{0}_{(8.8)}$ | $1_{(9.5)}$ | $\mathbf{0}_{(8.3)}$ | $\mathbf{0}_{(7.3)}$ | $\mathbf{0}_{(7.6)}$ | $1_{(9.1)}$ | $\mathbf{0}_{(7)}$ | $\mathbf{0}_{(6.3)}$ |
| | | | 700 | $2_{(4.7)}$ | $\mathbf{0}_{(3.6)}$ | $\mathbf{0}_{(3.7)}$ | $9_{(10)}$ | $5_{(7.2)}$ | $7_{(8.3)}$ | $6_{(6.3)}$ | $\mathbf{0}_{(3.2)}$ | $\mathbf{0}_{(4)}$ | $6_{(6.6)}$ | $\mathbf{0}_{(2.8)}$ | $\mathbf{0}_{(3.3)}$ |
| | | | 1 500 | $6_{(4.3)}$ | $\mathbf{0}_{(1.9)}$ | $\mathbf{0}_{(1.9)}$ | $11_{(9.8)}$ | $6_{(6.7)}$ | $6_{(7.3)}$ | $7_{(6.1)}$ | $\mathbf{0}_{(1.8)}$ | $\mathbf{0}_{(2)}$ | $7_{(6)}$ | $\mathbf{0}_{(1.4)}$ | $\mathbf{0}_{(1.5)}$ |
| | | **avg. rank** | | **1.33** | | | 5.17 | | | 1.83 | | | 1.94 | | |

Table 4: Rank and average epsilon indicator value ($\times 10^2$, between brackets) after 200, 700, and 1 500 evaluations for MNK-landscapes. The results are w.r.t. $\mathsf{Optim}_{\mathsf{PLS}}$ and $\mathsf{Order}_{\mathsf{static}}$.

| | | | | $\mathsf{Select}_{\mathsf{local}}$ | | | $\mathsf{Select}_{\mathsf{global}}$ | | | $\mathsf{Select}_{\mathsf{BI}}$ | | | $\mathsf{Select}_{\mathsf{NBI}}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #eval | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 |
| $\mathsf{Optim}_{\mathsf{PLS}}$ | $k=0$ | $n=25$ | 200 | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(\underline{0.4})}$ | $6_{(1.7)}$ | $10_{(6.4)}$ | $9_{(5.4)}$ | $9_{(4.9)}$ | $\mathbf{0}_{(0.5)}$ | $3_{(0.8)}$ | $3_{(1.2)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.5)}$ | $2_{(1)}$ |
| | | | 700 | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(\underline{0.3})}$ | $9_{(5.4)}$ | $9_{(4.6)}$ | $9_{(4.4)}$ | $1_{(0.5)}$ | $4_{(0.8)}$ | $4_{(1.1)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.5)}$ | $\mathbf{0}_{(0.5)}$ |
| | | | 1 500 | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(\underline{0.3})}$ | $10_{(4.7)}$ | $9_{(4.3)}$ | $9_{(3.9)}$ | $1_{(0.5)}$ | $4_{(0.8)}$ | $4_{(1.1)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.5)}$ | $\mathbf{0}_{(0.5)}$ |
| | | $n=50$ | 200 | $\mathbf{0}_{(\underline{0.4})}$ | $3_{(2.3)}$ | $6_{(4.7)}$ | $9_{(8.1)}$ | $9_{(6.9)}$ | $7_{(6.6)}$ | $\mathbf{0}_{(0.5)}$ | $3_{(2.2)}$ | $6_{(4.7)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(1.5)}$ | $4_{(4.1)}$ |
| | | | 700 | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(\underline{0.3})}$ | $2_{(0.7)}$ | $10_{(7.2)}$ | $9_{(6.3)}$ | $9_{(6)}$ | $1_{(0.5)}$ | $\mathbf{0}_{(0.6)}$ | $4_{(0.8)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.3)}$ | $1_{(0.5)}$ |
| | | | 1 500 | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(\underline{0.2})}$ | $9_{(6.7)}$ | $9_{(6.1)}$ | $9_{(5.9)}$ | $2_{(0.5)}$ | $\mathbf{0}_{(0.6)}$ | $6_{(0.7)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(0.4)}$ |
| | | **avg. rank** | | 0.94 | | | 9.06 | | | 2.56 | | | **0.39** | | |
| | $k=1$ | $n=25$ | 200 | $5_{(3.7)}$ | $\mathbf{0}_{(0.8)}$ | $1_{(1.5)}$ | $8_{(6.8)}$ | $8_{(6.5)}$ | $7_{(6.1)}$ | $6_{(5)}$ | $\mathbf{0}_{(0.9)}$ | $1_{(2.1)}$ | $5_{(4.4)}$ | $\mathbf{0}_{(\underline{0.5})}$ | $1_{(1.3)}$ |
| | | | 700 | $6_{(2.8)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(\underline{0.3})}$ | $8_{(5.6)}$ | $8_{(5.6)}$ | $7_{(5.5)}$ | $7_{(4.6)}$ | $\mathbf{0}_{(0.6)}$ | $2_{(0.6)}$ | $6_{(4.1)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.4)}$ |
| | | | 1 500 | $6_{(2.8)}$ | $\mathbf{0}_{(0.3)}$ | $\mathbf{0}_{(\underline{0.3})}$ | $8_{(5.2)}$ | $7_{(4.8)}$ | $7_{(5)}$ | $7_{(4.3)}$ | $\mathbf{0}_{(0.6)}$ | $3_{(0.6)}$ | $6_{(4)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.4)}$ |
| | | $n=50$ | 200 | $1_{(4.6)}$ | $\mathbf{0}_{(3.5)}$ | $2_{(5.4)}$ | $7_{(7.1)}$ | $8_{(7.5)}$ | $7_{(7.7)}$ | $1_{(4.9)}$ | $1_{(4)}$ | $1_{(5.2)}$ | $1_{(4.6)}$ | $\mathbf{0}_{(\underline{2.4})}$ | $1_{(4.3)}$ |
| | | | 700 | $6_{(3.5)}$ | $\mathbf{0}_{(0.6)}$ | $3_{(1.7)}$ | $9_{(6.7)}$ | $9_{(7.3)}$ | $9_{(6.7)}$ | $6_{(4.1)}$ | $\mathbf{0}_{(0.6)}$ | $3_{(1.9)}$ | $6_{(4)}$ | $\mathbf{0}_{(\underline{0.4})}$ | $1_{(1)}$ |
| | | | 1 500 | $6_{(3.4)}$ | $\mathbf{0}_{(\underline{0.4})}$ | $\mathbf{0}_{(0.4)}$ | $9_{(6.6)}$ | $9_{(7.1)}$ | $9_{(6.4)}$ | $6_{(4)}$ | $\mathbf{0}_{(0.6)}$ | $\mathbf{0}_{(0.5)}$ | $6_{(4)}$ | $\mathbf{0}_{(0.4)}$ | $\mathbf{0}_{(0.4)}$ |
| | | **avg. rank** | | 2 | | | 8 | | | 2.44 | | | **1.83** | | |
| | $k=2$ | $n=25$ | 200 | $3_{(6.4)}$ | $\mathbf{0}_{(4.8)}$ | $\mathbf{0}_{(5)}$ | $9_{(8.9)}$ | $\mathbf{0}_{(5.8)}$ | $\mathbf{0}_{(6.4)}$ | $3_{(7.7)}$ | $\mathbf{0}_{(3.9)}$ | $\mathbf{0}_{(5.3)}$ | $1_{(6.8)}$ | $\mathbf{0}_{(\underline{3.8})}$ | $\mathbf{0}_{(4)}$ |
| | | | 700 | $7_{(6)}$ | $3_{(2.3)}$ | $\mathbf{0}_{(1)}$ | $8_{(8)}$ | $4_{(3.6)}$ | $4_{(4.7)}$ | $5_{(5.9)}$ | $3_{(2.5)}$ | $\mathbf{0}_{(0.7)}$ | $6_{(5.9)}$ | $3_{(2.1)}$ | $\mathbf{0}_{(\underline{0.7})}$ |
| | | | 1 500 | $6_{(5.7)}$ | $3_{(1.8)}$ | $\mathbf{0}_{(\underline{0.2})}$ | $8_{(7.1)}$ | $6_{(3.6)}$ | $6_{(4.5)}$ | $6_{(5.4)}$ | $3_{(2.3)}$ | $\mathbf{0}_{(0.6)}$ | $7_{(5.7)}$ | $3_{(1.9)}$ | $\mathbf{0}_{(0.5)}$ |
| | | $n=50$ | 200 | $2_{(8)}$ | $\mathbf{0}_{(\underline{5.6})}$ | $\mathbf{0}_{(6.6)}$ | $7_{(10.7)}$ | $\mathbf{0}_{(8.1)}$ | $5_{(9.5)}$ | $5_{(9)}$ | $\mathbf{0}_{(6.4)}$ | $\mathbf{0}_{(6.7)}$ | $4_{(8.5)}$ | $\mathbf{0}_{(6.2)}$ | $\mathbf{0}_{(5.9)}$ |
| | | | 700 | $4_{(5.9)}$ | $\mathbf{0}_{(3)}$ | $\mathbf{0}_{(3.8)}$ | $10_{(9.9)}$ | $4_{(6.6)}$ | $6_{(7.8)}$ | $4_{(6.2)}$ | $\mathbf{0}_{(2.7)}$ | $\mathbf{0}_{(4)}$ | $6_{(6.3)}$ | $\mathbf{0}_{(\underline{2.4})}$ | $\mathbf{0}_{(3.3)}$ |
| | | | 1 500 | $6_{(5.2)}$ | $\mathbf{0}_{(2.2)}$ | $\mathbf{0}_{(2.3)}$ | $11_{(9.3)}$ | $6_{(5.6)}$ | $7_{(7.2)}$ | $6_{(5.7)}$ | $\mathbf{0}_{(1.9)}$ | $\mathbf{0}_{(2.6)}$ | $6_{(6)}$ | $\mathbf{0}_{(\underline{1.8})}$ | $\mathbf{0}_{(1.8)}$ |
| | | **avg. rank** | | **1.89** | | | 5.61 | | | 1.94 | | | 2 | | |

Table 5: Rank and average epsilon indicator value ($\times 10^2$, between brackets) after 200, 700, and 1 500 evaluations for MUBQP. The results are w.r.t. $\mathsf{Order}_{\mathsf{static}}$. A dash (—) indicates that the algorithm was not able to reach the corresponding budget within the maximum CPU time allowed in our experiments.

| | | | $\mathsf{Select}_{\mathsf{local}}$ | | | $\mathsf{Select}_{\mathsf{global}}$ | | | $\mathsf{Select}_{\mathsf{BI}}$ | | | $\mathsf{Select}_{\mathsf{NBI}}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #eval | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 | O=1 | O=2 | O=3 |
| $\mathsf{Optim}_{\mathsf{MOEA/D}}$ | $n=25$ | 200 | $1_{(371.5)}$ | $\mathbf{0}_{(194.6)}$ | $\mathbf{0}_{(252.2)}$ | $5_{(756.2)}$ | $4_{(639.8)}$ | $3_{(658.9)}$ | $3_{(515.5)}$ | $\mathbf{0}_{(\underline{179.3})}$ | $\mathbf{0}_{(234.9)}$ | $5_{(776.2)}$ | $4_{(666.2)}$ | $5_{(709.6)}$ |
| | | 700 | $4_{(280.7)}$ | $\mathbf{0}_{(\underline{33.7})}$ | $1_{(112)}$ | $6_{(638.3)}$ | $4_{(499.9)}$ | $4_{(388.3)}$ | $4_{(389.5)}$ | $\mathbf{0}_{(55.8)}$ | $1_{(98)}$ | $5_{(663.9)}$ | $4_{(565.1)}$ | $4_{(476)}$ |
| | | 1 500 | $4_{(260)}$ | $\mathbf{0}_{(\underline{24.6})}$ | $\mathbf{0}_{(49.7)}$ | $8_{(576.6)}$ | $4_{(344.4)}$ | $4_{(285.6)}$ | $4_{(370.6)}$ | $1_{(47.6)}$ | $\mathbf{0}_{(46.8)}$ | $8_{(577.3)}$ | $4_{(528.6)}$ | $4_{(278.7)}$ |
| | $n=50$ | 200 | $\mathbf{0}_{(\underline{1421.3})}$ | $\mathbf{0}_{(1853.3)}$ | $\mathbf{0}_{(1841.4)}$ | $6_{(3302.9)}$ | $6_{(2957.1)}$ | $6_{(2986.2)}$ | $\mathbf{0}_{(1551.5)}$ | $\mathbf{0}_{(1784.6)}$ | $\mathbf{0}_{(1728.7)}$ | $6_{(3324.9)}$ | $6_{(2860.8)}$ | $6_{(3027.2)}$ |
| | | 700 | $\mathbf{0}_{(\underline{935.6})}$ | $\mathbf{0}_{(1063.3)}$ | $\mathbf{0}_{(1031.7)}$ | $6_{(2713.7)}$ | $6_{(2359.3)}$ | $5_{(2079.5)}$ | $\mathbf{0}_{(937.2)}$ | $\mathbf{0}_{(1050)}$ | $\mathbf{0}_{(978.8)}$ | $7_{(3003.5)}$ | $6_{(2549.3)}$ | $5_{(2277.9)}$ |
| | | 1 500 | $1_{(826.4)}$ | $\mathbf{0}_{(\underline{564.6})}$ | $1_{(883.3)}$ | $6_{(2222.2)}$ | $6_{(1664.4)}$ | $2_{(1403.7)}$ | $1_{(829.2)}$ | $\mathbf{0}_{(668.9)}$ | $1_{(937.1)}$ | $8_{(2762.7)}$ | $7_{(2360.7)}$ | $1_{(1637.8)}$ |
| | **avg. rank** | | **0.67** | | | 5.06 | | | 0.83 | | | 5.28 | | |
| $\mathsf{Optim}_{\mathsf{MLS}}$ | $n=25$ | 200 | $\mathbf{0}_{(290.1)}$ | $\mathbf{0}_{(399.1)}$ | $2_{(407.8)}$ | $10_{(707.4)}$ | $2_{(424)}$ | $2_{(469.3)}$ | $\mathbf{0}_{(360.2)}$ | $\mathbf{0}_{(333.2)}$ | $\mathbf{0}_{(335.2)}$ | $\mathbf{0}_{(431.7)}$ | $\mathbf{0}_{(\underline{233.6})}$ | $\mathbf{0}_{(315.6)}$ |
| | | 700 | $5_{(241.7)}$ | $1_{(71.8)}$ | $5_{(239.1)}$ | $11_{(648.2)}$ | $7_{(412.5)}$ | $6_{(350.6)}$ | $6_{(339.6)}$ | $\mathbf{0}_{(48.2)}$ | $1_{(108.1)}$ | $5_{(404.7)}$ | $\mathbf{0}_{(\underline{16.3})}$ | $1_{(59.7)}$ |
| | | 1 500 | $5_{(239.4)}$ | $1_{(38.3)}$ | $2_{(90.4)}$ | $11_{(584.1)}$ | $7_{(408.6)}$ | $6_{(327.2)}$ | $6_{(333.1)}$ | $\mathbf{0}_{(36.3)}$ | $1_{(57.3)}$ | $6_{(383.3)}$ | $\mathbf{0}_{(\underline{14.9})}$ | $1_{(31.5)}$ |
| | $n=50$ | 200 | $\mathbf{0}_{(\underline{1132.8})}$ | $3_{(2800.1)}$ | $3_{(2670.3)}$ | $3_{(2352.1)}$ | $3_{(2661.8)}$ | $3_{(2775.1)}$ | $\mathbf{0}_{(1260.6)}$ | $3_{(3077.3)}$ | $3_{(2741.5)}$ | $\mathbf{0}_{(1239.1)}$ | $3_{(2602.7)}$ | $3_{(2823.9)}$ |
| | | 700 | $\mathbf{0}_{(\underline{602.4})}$ | $3_{(1562)}$ | — | $7_{(2309.4)}$ | $3_{(1585)}$ | — | $1_{(857.1)}$ | $3_{(1373)}$ | — | $1_{(830.1)}$ | $3_{(1598.5)}$ | — |
| | | 1 500 | $2_{(557.9)}$ | $2_{(793.6)}$ | — | $7_{(2273.3)}$ | $6_{(1401.1)}$ | — | $3_{(856.7)}$ | $\mathbf{0}_{(149.2)}$ | — | $3_{(830.1)}$ | $\mathbf{0}_{(\underline{115.1})}$ | — |
| | **avg. rank** | | 1.89 | | | 5.22 | | | 1.5 | | | **1.44** | | |
| $\mathsf{Optim}_{\mathsf{PLS}}$ | $n=25$ | 200 | $\mathbf{0}_{(343.9)}$ | $\mathbf{0}_{(223)}$ | $\mathbf{0}_{(248.4)}$ | $11_{(765.4)}$ | $4_{(516.3)}$ | $1_{(459.7)}$ | $1_{(409.6)}$ | $\mathbf{0}_{(\underline{201.5})}$ | $\mathbf{0}_{(269.4)}$ | $2_{(440.3)}$ | $\mathbf{0}_{(341.2)}$ | $\mathbf{0}_{(363.6)}$ |
| | | 700 | $6_{(272.3)}$ | $\mathbf{0}_{(\underline{19.4})}$ | $1_{(116.7)}$ | $10_{(644.9)}$ | $9_{(486.2)}$ | $6_{(391.7)}$ | $6_{(310.7)}$ | $1_{(69.8)}$ | $1_{(74.9)}$ | $6_{(320.2)}$ | $\mathbf{0}_{(55.4)}$ | $1_{(107.1)}$ |
| | | 1 500 | $6_{(264.3)}$ | $\mathbf{0}_{(\underline{15.3})}$ | $\mathbf{0}_{(50.4)}$ | $11_{(606)}$ | $9_{(470)}$ | $6_{(369.6)}$ | $6_{(287.5)}$ | $1_{(69.8)}$ | $1_{(58.5)}$ | $6_{(307.9)}$ | $\mathbf{0}_{(55.4)}$ | $1_{(68.4)}$ |
| | $n=50$ | 200 | $\mathbf{0}_{(\underline{1187.5})}$ | $3_{(2139.8)}$ | $3_{(2106.1)}$ | $8_{(2939)}$ | $3_{(2471.9)}$ | $3_{(2410.4)}$ | $\mathbf{0}_{(1308.5)}$ | $3_{(2352.7)}$ | $3_{(2403.8)}$ | $\mathbf{0}_{(1309.1)}$ | $3_{(2471.9)}$ | $3_{(2410.4)}$ |
| | | 700 | $\mathbf{0}_{(\underline{781.3})}$ | $\mathbf{0}_{(1049.5)}$ | — | $7_{(2726)}$ | $6_{(1588.1)}$ | — | $1_{(1118.7)}$ | $\mathbf{0}_{(899.5)}$ | — | $\mathbf{0}_{(1015.3)}$ | $\mathbf{0}_{(945.9)}$ | — |
| | | 1 500 | $3_{(734.3)}$ | $\mathbf{0}_{(142.1)}$ | — | $7_{(2600.6)}$ | $6_{(1588)}$ | — | $3_{(1054.8)}$ | $\mathbf{0}_{(177.8)}$ | — | $3_{(960.5)}$ | $\mathbf{0}_{(\underline{115.3})}$ | — |
| | **avg. rank** | | **1.22** | | | 5.94 | | | 1.5 | | | 1.39 | | |

the underlying problem is quadratic, a Walsh order of 2 is always performing better. This is with no surprise since performance is tightly related to the ability of the Walsh surrogate to faithfully approximate the underlying landscape, which is directly related to the choice of the Walsh order.

Table 6: Sorted selection strategies by average rank ($L = \mathsf{Select}_{\mathsf{local}}$, $G = \mathsf{Select}_{\mathsf{global}}$, $B_C = \mathsf{Select}_{\mathsf{BI}}$, $B_N = \mathsf{Select}_{\mathsf{NBI}}$).

| | | $\mathsf{Optim}_{\mathsf{MOEA/D}}$ | $\mathsf{Optim}_{\mathsf{MLS}}$ | $\mathsf{Optim}_{\mathsf{PLS}}$ |
|---|---|---|---|---|
| MNK-landscapes | $k = 0$ | $L \prec B_C \prec B_N \prec G$ | $B_N \prec B_C \prec L \prec G$ | $B_N \prec L \prec B_C \prec G$ |
| MNK-landscapes | $k = 1$ | $B_C \prec L \prec B_N \prec G$ | $B_N \prec B_C \prec L \prec G$ | $B_N \prec L \prec B_C \prec G$ |
| MNK-landscapes | $k = 2$ | $B_C \prec L \prec B_N \prec G$ | $L \prec B_C \prec B_N \prec G$ | $L \prec B_C \prec B_N \prec G$ |
| MUBQP | | $L \prec B_C \prec G \prec B_N$ | $B_N \prec B_C \prec L \prec G$ | $L \prec B_N \prec B_C \prec G$ |

Overall, we can hence conclude that the rank obtained when using a surrogate optimizer with a particular selection strategy changes consistently with the choice of the order, suggesting that there is an optimal setting that depends on the underlying landscape. Additionally, we observe that the ranks may vary depending on the considered budget. This indicates that the different algorithm configurations may expose different anytime behaviors. This is to be analyzed in more detail afterwards, in light of the other proposed *dynamic* strategies for setting the Walsh order.

Secondly, we can clearly see that the selection strategy providing the best rank is *different* depending on which surrogate optimizer is considered. This is an important finding showing that there is a strong dependency between these two components, i.e., for optimal performance, the selection strategy has to be configured differently depending on the adopted surrogate optimizer. More specifically, the two local search optimizers ($\mathsf{Optim}_{\mathsf{MLS}}$ in Table 3 and $\mathsf{Optim}_{\mathsf{PLS}}$ in Table 4) perform at their best when using the $\mathsf{Select}_{\mathsf{NBI}}$ strategy for MNK-landscapes with $k = 0$ and $k = 1$, whereas using the $\mathsf{Select}_{\mathsf{local}}$ strategy is a slightly better choice when $k = 2$. In contrast, for MUBQP (Table 5), these two optimizers perform at their best when using different selection strategies: $\mathsf{Optim}_{\mathsf{PLS}}$ performs better with the $\mathsf{Select}_{\mathsf{local}}$ strategy, and $\mathsf{Optim}_{\mathsf{MLS}}$ performs better with the $\mathsf{Select}_{\mathsf{NBI}}$ and $\mathsf{Select}_{\mathsf{BI}}$ strategies. Looking at the third $\mathsf{Optim}_{\mathsf{MOEA/D}}$ optimizer, the situation is seemingly different: $\mathsf{Select}_{\mathsf{BI}}$ is the best performing strategy for MNK-landscapes (in Table 2), whereas $\mathsf{Select}_{\mathsf{local}}$ is better for MUBQP (in Table 5). We also found that these two selection strategies are definitely a better choice when using $\mathsf{Optim}_{\mathsf{MOEA/D}}$ compared to the $\mathsf{Select}_{\mathsf{NBI}}$ and $\mathsf{Select}_{\mathsf{global}}$ strategies. In contrast to the work in [68], where $\mathsf{Select}_{\mathsf{local}}$ is recommended, these observations show that other selection strategies can be more accurate, depending on both the considered problem and the surrogate optimizer.

To summarize these complex dependencies, we provide in Table 6 an overview of the relative performance of the selection strategies for the different surrogate optimizers, obtained by sorting the selection strategies according to the average ranks from Tables 2, 3, 4 and 5 over the considered instances and budgets (as given by the 'avg. rank' rows). Looking at the impact of the selection strategy over the different surrogate optimizers, it is interesting to remark that for $\mathsf{Optim}_{\mathsf{PLS}}$ and $\mathsf{Optim}_{\mathsf{MLS}}$, the $\mathsf{Select}_{\mathsf{NBI}}$ strategy is to be preferred, with an average rank of 1.40 and 1.29 respectively, whereas its average rank is of 4.36 when used in combination with $\mathsf{Optim}_{\mathsf{MOEA/D}}$. In other words, the $\mathsf{Select}_{\mathsf{NBI}}$ strategy can be viewed as a relatively good selection strategy, except for $\mathsf{Optim}_{\mathsf{MOEA/D}}$. Besides, the $\mathsf{Select}_{\mathsf{global}}$ strategy shows the worst results overall and independently

of the considered surrogate optimizer.

In the rest of the paper, and unless explicitly stated, we shall always consider the best performing selection strategy when dealing with a particular surrogate optimizer, as depicted in Table 6.

### *5.2. Impact of the Surrogate Optimizer*

In this section, we compare the different surrogate optimizers when different static Walsh orders are used. This is depicted in Fig. 1 and 2 showing the convergence (anytime) profile of the different configurations respectively for MNK-landscapes and MUBQP.

As a first observation, we clearly see that the $\mathsf{Optim}_{\mathsf{MLS}}$ and $\mathsf{Optim}_{\mathsf{PLS}}$ optimizers have significantly better convergence profiles compared to $\mathsf{Optim}_{\mathsf{MOEA/D}}$. This is consistent over all considered instances, with very few exceptions. At this stage of the analysis, we recall that the proposed Walsh-based surrogate approach uses the decomposition paradigm to both structure the population, and to select the next solution to evaluate. Hence, this observation shows that, (i) such an approach does *not* necessarily need to be configured with a decomposition-based *inner* optimizer to handle the underlying surrogate; and more importantly, (ii) other more accurate optimizers should be used specifically to the underlying Walsh surrogate. This is again to be contrasted with the work in [68], where the decomposition-based MOEA/D algorithm was employed at the different design stages.

Let us now analyze in more detail the two $\mathsf{Optim}_{\mathsf{MLS}}$ and $\mathsf{Optim}_{\mathsf{PLS}}$ optimizers, which were found to lead to a better approximation quality. Although overall they expose a similar behavior, their relative anytime profile depends both on the difficulty of the tackled problem, and on the complexity of the Walsh surrogate, as implied by the choice of the Walsh order. This is discussed in the next paragraphs by splitting our experimented instances into three classes: (i) linear, i.e., MNK-landscapes with $k = 0$, (ii) quadratic, i.e., MNK-landscapes with $k = 1$ and MUBQP, and (iii) cubic, i.e., MNK-landscapes with $k = 2$. It is important to remark that the order of the exact Walsh transform of any function in these three classes is bounded respectively by 1, 2, and 3.

For the two linear instances (with $k = 0$), $\mathsf{Optim}_{\mathsf{PLS}}$ used in combination with the (exact) Walsh order of 1 converges very quickly to a high-quality approximation set, and $\mathsf{Optim}_{\mathsf{MLS}}$ is only able to obtain a better quality when a higher number of evaluations is affordable. When using an over-estimated Walsh order of 2 or 3, $\mathsf{Optim}_{\mathsf{MLS}}$ performs significantly better than $\mathsf{Optim}_{\mathsf{PLS}}$ independently of the available budget.

For quadratic instances, when setting the Walsh order to the exact transform order of 2, we found that $\mathsf{Optim}_{\mathsf{MLS}}$ and $\mathsf{Optim}_{\mathsf{PLS}}$ have seemingly the same convergence profile for $n = 25$. By contrast, for $n = 50$, $\mathsf{Optim}_{\mathsf{PLS}}$ is slightly better under a restricted budget, while $\mathsf{Optim}_{\mathsf{MLS}}$ is slightly better under higher budgets, which is to recall their behavior for linear instances. When the Walsh order is set to an underestimated value of 1, the approximation quality obtained with all optimizers drops down in comparison to an (exact) order of 2. Interestingly, using an underestimated order value does not prevent to find improvements during the very first iterations, that

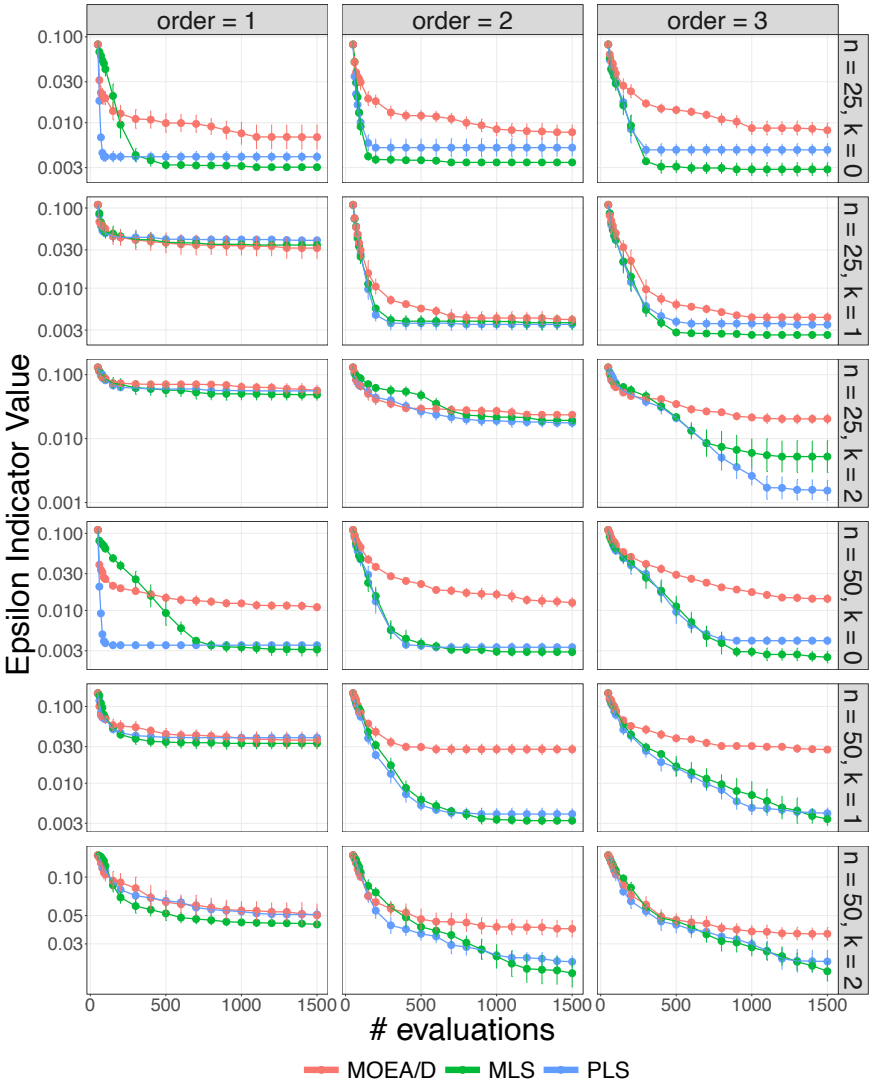


Figure 1: Convergence profiles when using the surrogate optimizers with their corresponding best selection strategy for MNK-landscapes.

is, with a very restricted budget. When the Walsh order is set to an overestimated value of 3, the overall approximation quality of both optimizers seems to be comparable to the exact setting of the order.

Finally, for the most complex cubic instances ($k = 2$) and for $n = 25$, an underestimated order of 1 or 2 is not competitive with the exact transform order 3, independently of the surrogate optimizer. For dimension $n = 50$, all algorithms are clearly performing very poorly when an underestimated order of 1 is used. However, the situation improves when using an order of 2, which indicates that for such difficult problems, a sufficiently large but non-exact Walsh order still allows for a reasonable approximation.

From this set of observations, we can draw two general conclusions. First, the two local search

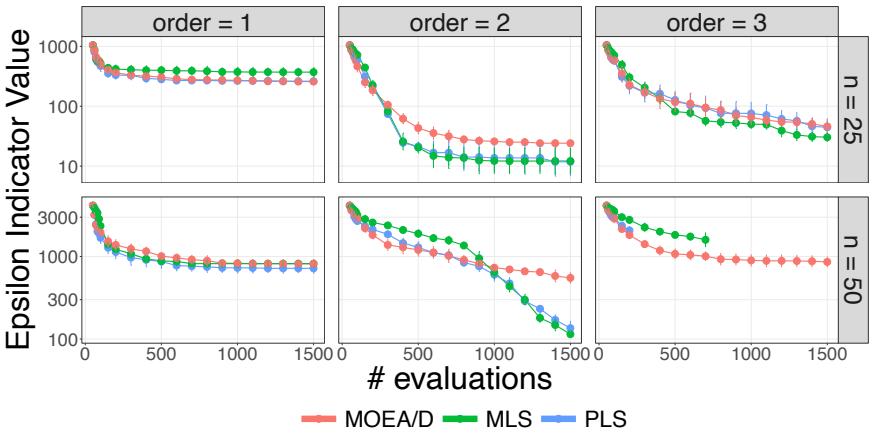


Figure 2: Convergence profiles when using the surrogate optimizers with their corresponding best selection strategy for MUBQP.

optimizers $\mathsf{Optim}_{\mathsf{MLS}}$ and $\mathsf{Optim}_{\mathsf{PLS}}$ are efficient in dealing with the discrete Walsh surrogate, which is to contrast with the $\mathsf{Optim}_{\mathsf{MOEA/D}}$ optimizer. Second, an effective surrogate optimizer alone is not able to provide a good performance independently of the available budget and the Walsh order setting. The impact of the order choice strategy is studied next, hence addressing the last component of the considered framework.

### *5.3. Impact of the Walsh Order*

To study the impact of the strategy used for the Walsh order, we focus on $\mathsf{Optim}_{\mathsf{MLS}}$ and $\mathsf{Optim}_{\mathsf{PLS}}$, since they were found to substantially outperform $\mathsf{Optim}_{\mathsf{MOEA/D}}$. This is shown in Fig. 3 and 4, respectively for MNK-landscapes and MUBQP.

From these two figures, it becomes very clear that underestimating the order value (relatively to the exact Walsh transform) has a dramatic impact on the overall performance. Besides, overestimating the order value does not always allow for competitive results. This might seem surprising, since overestimating the order makes the Walsh model more complex, and should not decrease its accuracy. While such a claim sounds true in theory, increasing the Walsh order makes the model much more challenging to fit accurately in practice. To illustrate this issue, we show in Fig. 5 the mean absolute error of the Walsh model as a function of the size of the training dataset. We clearly see that overestimating the order can lead to a worst fitting quality depending on the characteristic on the degree of non-linearity $k$, and the size of the training set. This is to be attributed to the fact that the more complex the model is, the largest the training data should be in order for the Lasso regression to be successful in finding a good fit.

Furthermore, in comparison to a static setting of the Walsh order ($\mathsf{Order}_{\mathsf{static}}$), the considered dynamic strategies ($\mathsf{Order}_{\mathsf{random}}$ and $\mathsf{Order}_{\mathsf{greedy}}$) are clearly a better option. Interestingly, there is no clear separation between the greedy and the random strategy independently of the underlying surrogate optimizer, problem instance and budget. For example, we can see that $\mathsf{Order}_{\mathsf{greedy}}$ out-

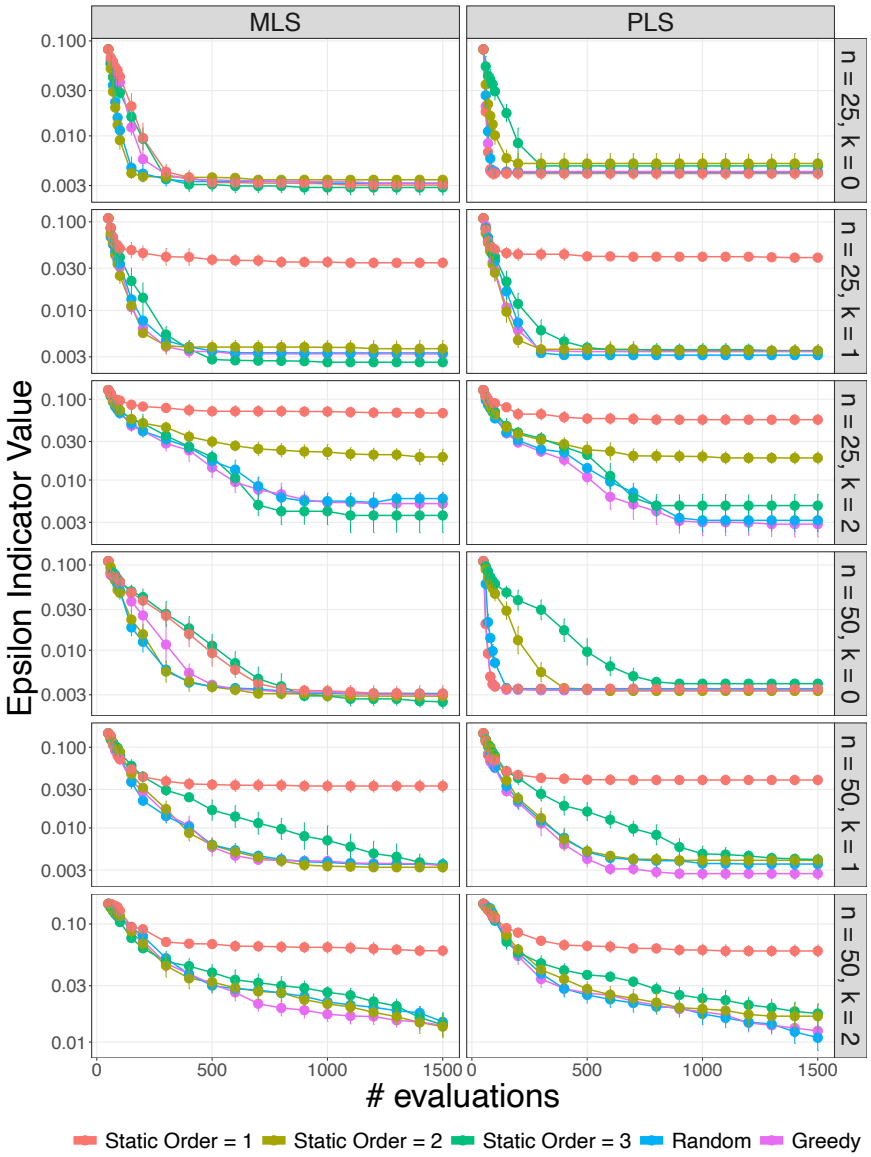


Figure 3: Convergence profile of when using different Walsh order settings with the $\mathsf{Select}_{\mathsf{NBI}}$ strategy for MNK-landscapes.

performs $\mathsf{Order}_{\mathsf{random}}$ for MUBQP instances of size $n = 50$. However, this is no more true for $n = 25$ and the $\mathsf{Optim}_{\mathsf{PLS}}$ surrogate optimizer. Similarly, $\mathsf{Order}_{\mathsf{greedy}}$ obtains a better convergence profile than $\mathsf{Order}_{\mathsf{random}}$ when using $\mathsf{Optim}_{\mathsf{PLS}}$ for a MNK-landscapes with $n = 50$ and $k = 0$, whereas this is no more true when combining $\mathsf{Order}_{\mathsf{greedy}}$ with $\mathsf{Optim}_{\mathsf{MLS}}$. Overall, we conclude that both dynamic strategies for setting the Walsh order obtain very competitive results. A dynamic order strategy is always better than an underestimated static order, and is better than an overestimated static order in most cases. Surprisingly, it is even better than a static strategy using the exact Walsh transform order for some instances; e.g. MNK-landscapes with $n = 50$ and $k = 1$. These

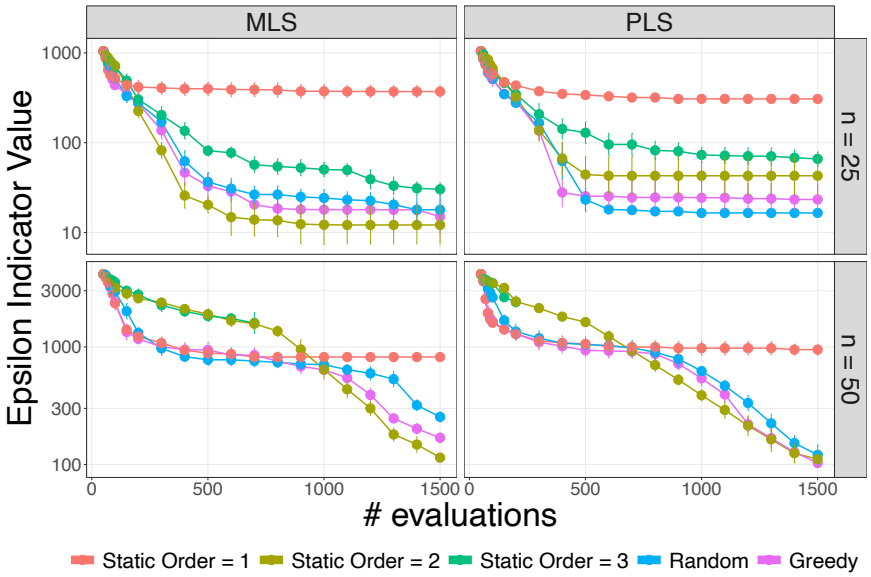


Figure 4: Convergence profile when using different Walsh order settings with the $\mathsf{Select}_{\mathsf{NBI}}$ strategy for MUBQP.

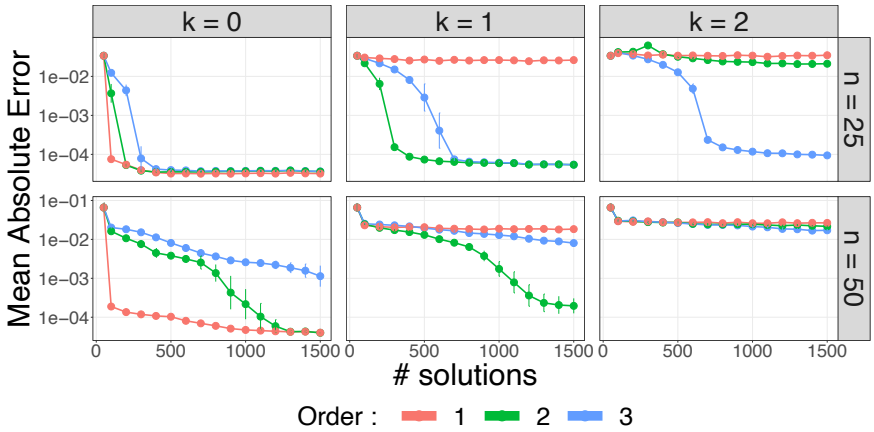


Figure 5: Mean absolute error (MAE) obtained by the Walsh surrogate according to the number of solutions in the training dataset with the $\mathsf{Optim}_{\mathsf{MLS}}$ optimizer combined to the $\mathsf{Select}_{\mathsf{NBI}}$ strategy for order $\mathsf{O} \in \{1, 2, 3\}$ on MNK-landscapes.

results suggest that more sophisticated dynamic strategies for setting the Walsh order are worth to be investigated in the future.

### *5.4. Approximation quality of* SMCO/W *vs. Surrogate-less algorithms*

In this section, we show the benefits behind using a surrogate model to assist the search process. For this purpose, we compare the approximation quality obtained when running the three algorithms MOEA/D, PLS, and MLS on the original problems (without using Walsh surrogate). Our results are reported in Fig. 6. We only show the results obtained with the $\mathsf{Select}_{\mathsf{NBI}}$ strategy and the $\mathsf{Order}_{\mathsf{greedy}}$ strategy, since these two strategies were found to provide fairly good results over

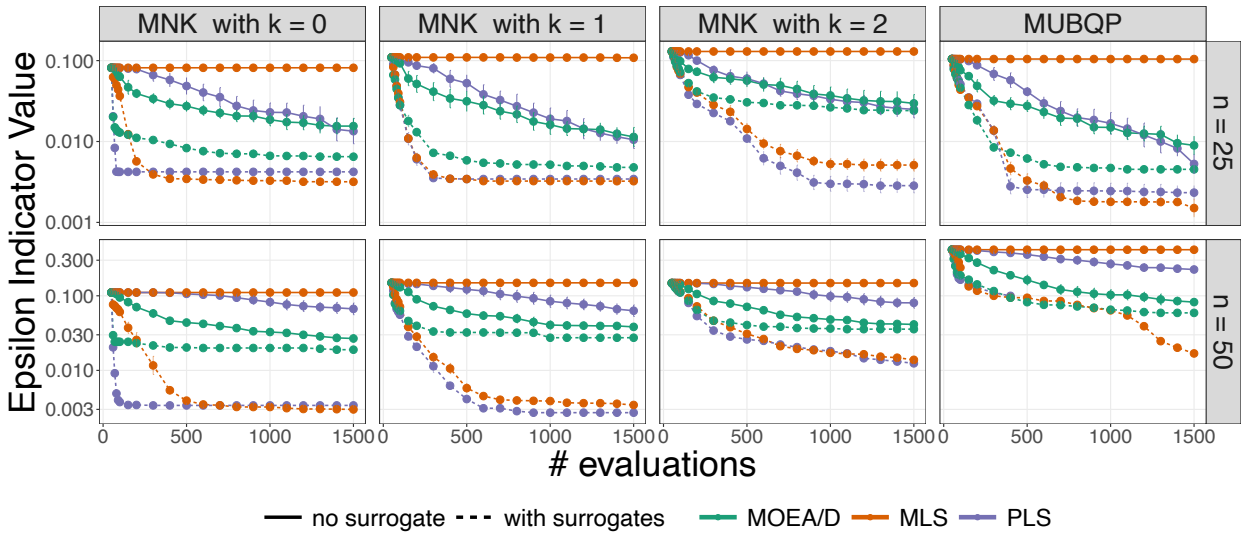


Figure 6: Comparison of surrogate-less approaches with SMCO/W using a greedy Walsh order ($\mathsf{Order}_{\mathsf{greedy}}$) and the best normalized improvement selection strategy ($\mathsf{Select}_{\mathsf{NBI}}$). For MUBQP, the indicator has been scaled by a factor $10^{-2}$ for a better readability of the figures.

all the experimented instances. We can see that the so-obtained surrogate-assisted algorithms lead to substantially better approximation sets, independently of the considered instance and budget. This provides evidence on the accuracy of the designed approach and its effectiveness in reducing the number of function evaluations required to compute a high quality approximation set.

The results presented in Fig. 6 hold under the standard assumption that the evaluation function is very costly, in the sense that its CPU time cost dominates all the other parts of the considered algorithms. However, it is well-known that using a surrogate model, like in any machine learning technique, can incur some computational time. Mitigating such a computational cost with respect to the real cost of the expensive evaluation can be important depending on the problem considered in practice. This aspect is only discussed to a small extent in the literature. For completeness, and although our work is not concerned with a particular target application problem, we conclude our analysis by conducting additional experiments allowing us to fairly discuss the computational CPU times incurred by using a Walsh surrogate and its implications, while adopting the same benchmark-oriented methodology.

### *5.4.1. Considerations on the Walsh surrogate CPU time*

To analyze the CPU time cost incurred by the Walsh surrogate, we adopt the following experimental procedure. For each problem instance, we generate successively $s \in \{250, 500, 1000, 1500\}$ random binary string solutions. The Walsh surrogate is trained as discussed previously for each order $\mathsf{O} \in \{1, 2, 3\}$. The average CPU (*training*) time required for this phase is then reported (over 100 independent repetitions). Using each trained model, we predict the objective value for a randomly generated binary string solution, and the average CPU (*prediction*) time is reported as well.

Table 7: CPU time of Walsh surrogate training phase (in *seconds*) and prediction phase (in *milliseconds*) using a sample size $s \in \{250, 500, 1000, 1500\}$. Results are shown for every static order $\mathsf{O}$ and each problem instance.

| | | | $\mathsf{O}=1$ | | | | $\mathsf{O}=2$ | | | | $\mathsf{O}=3$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MNK-landscapes | | | MUBQP | MNK-landscapes | | | MUBQP | MNK-landscapes | | | MUBQP |
| | | $s$ | $k=1$ | $k=2$ | $k=3$ | | $k=1$ | $k=2$ | $k=3$ | | $k=1$ | $k=2$ | $k=3$ | |
| Training (in $s$) | $n=25$ | 250 | 0.04 | 0.05 | 0.06 | 0.06 | 0.17 | 0.19 | 0.23 | 0.19 | 1.35 | 1.49 | 1.54 | 1.25 |
| | | 500 | 0.04 | 0.05 | 0.05 | 0.06 | 0.27 | 0.28 | 0.28 | 0.28 | 2.43 | 2.51 | 3.03 | 2.36 |
| | | 1000 | 0.05 | 0.05 | 0.06 | 0.06 | 0.53 | 0.52 | 0.55 | 0.51 | 4.50 | 4.60 | 4.85 | 4.48 |
| | | 1500 | 0.05 | 0.06 | 0.07 | 0.07 | 0.79 | 0.82 | 0.81 | 0.78 | 6.60 | 6.57 | 6.74 | 6.60 |
| | $n=50$ | 250 | 0.07 | 0.08 | 0.09 | 0.16 | 0.67 | 0.78 | 0.77 | 0.72 | 13.82 | 13.66 | 14.63 | 10.49 |
| | | 500 | 0.07 | 0.09 | 0.10 | 0.17 | 1.15 | 1.23 | 1.39 | 1.24 | 27.68 | 24.19 | 28.31 | 19.05 |
| | | 1000 | 0.09 | 0.10 | 0.12 | 0.18 | 2.23 | 2.28 | 2.94 | 2.38 | 43.38 | 35.01 | 48.07 | 32.86 |
| | | 1500 | 0.10 | 0.12 | 0.13 | 0.19 | 3.31 | 3.30 | 3.55 | 3.37 | 56.00 | 52.96 | 58.71 | 46.64 |
| Prediction (in $ms$) | $n=25$ | 250 | 0.06 | 0.06 | 0.06 | 0.06 | 0.06 | 0.13 | 0.41 | 0.51 | 0.07 | 0.51 | 0.60 | 4.27 |
| | | 500 | 0.06 | 0.06 | 0.06 | 0.06 | 0.06 | 0.11 | 0.55 | 0.55 | 0.07 | 0.16 | 1.12 | 4.58 |
| | | 1000 | 0.06 | 0.06 | 0.06 | 0.06 | 0.07 | 0.11 | 0.60 | 0.59 | 0.07 | 0.11 | 0.32 | 4.47 |
| | | 1500 | 0.06 | 0.06 | 0.06 | 0.06 | 0.07 | 0.18 | 0.63 | 0.61 | 0.08 | 0.12 | 0.30 | 4.34 |
| | $n=50$ | 250 | 0.10 | 0.10 | 0.10 | 0.10 | 0.24 | 0.46 | 0.50 | 2.00 | 1.11 | 1.32 | 1.77 | 14.53 |
| | | 500 | 0.10 | 0.10 | 0.10 | 0.10 | 0.12 | 0.26 | 0.81 | 2.13 | 0.50 | 1.80 | 2.40 | 41.41 |
| | | 1000 | 0.10 | 0.10 | 0.10 | 0.10 | 0.12 | 0.21 | 1.70 | 2.02 | 0.25 | 0.58 | 3.57 | 34.21 |
| | | 1500 | 0.10 | 0.10 | 0.10 | 0.10 | 0.13 | 0.22 | 2.40 | 2.23 | 0.20 | 0.50 | 3.41 | 34.02 |

These additional experiments were conducted in Python 3.7 on a personal Intel Core i7, 6-core processor (2.6 GHZ, 16 GB RAM) under macOS, without any further low-level code optimization. The obtained average CPU times for training and prediction are reported in Table 7.

We can first see that all CPU times (for both training and prediction) are strongly correlated with the considered Walsh order $\mathsf{O}$ and the problem size $n$, i.e., CPU times are higher for larger orders and larger dimensions. This is without surprise since higher orders/dimensions imply a more complex model, with more Walsh coefficients to estimate.

The CPU time required for the prediction phase appears to be overall relatively low, i.e., from few to at most some dozens of milliseconds. The CPU time for prediction can be viewed as the cost of the inferred surrogate function, that will be optimized internally by the embedded surrogate optimizer (e.g., MOEA/D, PLS, MLS). Hence, we can conclude that the obtained Walsh surrogate function is relatively cheap. Having such a cheap surrogate function is very desirable in order to be later extensively tackled by the embedded evolutionary optimizer. However, on the other hand, the CPU training time appears to be more substantial. In the worst case scenarios, using the largest training sample of size 1500 and the largest dimension $n = 50$, the training cost ranges from 0.13 seconds in average (over all instances) for a Walsh order $\mathsf{O} = 1$, to 3.38 seconds for $\mathsf{O} = 2$, and 53.58 seconds for $\mathsf{O} = 3$. Nonetheless, such CPU times can still be considered as reasonable in a practical expensive context, where the objective function cost can typically be at least on the order of minutes or hours. Notice in fact that the Walsh model needs to be trained once, before serving as a cheap substitute that can in turn be searched more extensively, hopefully in less time than the cost of one real expensive evaluation. For example, using the highest Walsh order $\mathsf{O} = 3$, it costs less than 2 minutes for an internal surrogate optimizer embedded in SMCO/W to evaluate the expected quality of more than $10^4$ (respectively $10^3$) candidate solutions for the

hardest MNK-landscape with $n = 50$ and $k = 2$ (respectively MUBQP with $n = 50$).

*5.4.2. Considerations on the acceleration ratio*

Having these CPU times in mind, we push our analysis further by eliciting how many function evaluations a surrogate-less optimizer needs in order to reach the same approximation quality than the proposed Walsh-assisted SMCO/W approach. This shall fairly complement the observations from Fig. 6, where all algorithms including the surrogate-less ones have been run with exactly the same number of real (expensive) function evaluations. More precisely, we execute the two surrogate-less algorithms, MOEA/D and PLS, using $10^5$ real function evaluations, which is a reasonably high budget for the considered instances. We then compute the number of function evaluations required for each surrogate-less algorithm to reach an approximation set having a given target epsilon indicator value. We choose different target values so as to render to relative difference with the SMCO/W approach when executed with a range of different budgets of $B_q = q \times 100$ evaluations, with $q \in [\![2, 15]\!]$. As such, we choose a target value $t_q$ to be the average epsilon indicator value attained by SMCO/W after $B_q$ evaluations, and using the greedy Walsh order ($\mathsf{Order}_{\mathsf{greedy}}$) and the best normalized improvement selection strategy ($\mathsf{Select}_{\mathsf{NBI}}$), as previously considered in Fig. 6. This experiment is repeated for 10 different seeds. Let $B$ be the number of true function evaluations needed by a surrogate-less algorithm to hit target $t_q$ as defined previously. We then define the *acceleration ratio* to be: $B/B_q$. Since SMCO/W was found to be always better than the considered surrogate-less approaches (Fig. 6), the acceleration ratio allows us to *quantify* in a more fine-grained manner *by how much* SMCO/W is superior. In fact, an acceleration ratio value of $x > 1$, for some target $t_q$, means that the corresponding surrogate-less algorithm needs $x$ times more function evaluations to reach the same average approximation quality attained by SMCO/W when using $B_q$ evaluations. The average acceleration ratios are depicted in Fig. 7 under the different experimented conditions, and as a function of the target affordable number of real function evaluations $B_q$. Notice that it might happen that one execution of the surrogate-less algorithms (MOEA/D and PLS) terminates *without* hitting a given target value $t_q$. In that case, following the standard definition of the so-called empirical running time (ERT) [80], we adopt an empirical-based assessment technique, where the surrogate-less algorithm is assumed to be restarted with a seed sampled uniformly from the 10 considered ones. In the case none of the 10 seeds were successful in hitting a target value $t_q$, Fig. 7 does not show any ratio; hence, indicating that all the 10 executions of the surrogate-less algorithm failed in hitting the target $t_q$ after exhausting the $10^5$ evaluations.

We can observe different trends depending on the characteristics of considered problem instance. For the easiest MNK-landscape with $n = 25$ and $k = 0$, PLS performs better than MOEA/D, and the acceleration ratio due to use of the Walsh surrogate decreases as the affordable budget $B_q$ increases. For example for a very restricted budget $B_q = 200$ evaluations, SMCO/W requires 11.5 (respectively, 47.5) times less evaluations than PLS (respectively, MOEA/D), whereas it requires around 2 (respectively, 27) times less evaluations than PLS (respectively, MOEA/D), for the highest

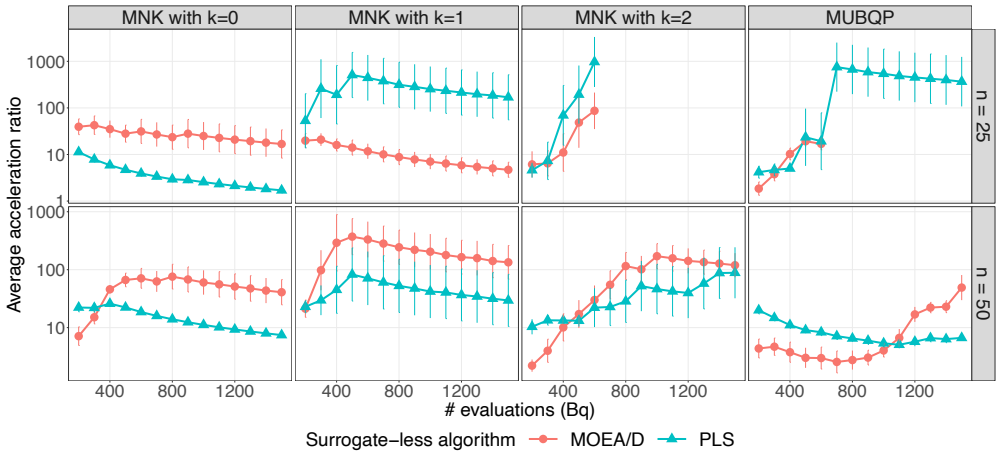


Figure 7: Average acceleration ratio between surrogate-less algorithms and SMCO/W (using $\mathsf{Order}_{\mathsf{greedy}}$ and $\mathsf{Select}_{\mathsf{NBI}}$) for every target approximation quality $t_q$ and as a function of the number of corresponding function evaluations $B_q$.

budget $B_q = 1500$. For the hardest MNK-landscape with $n = 50$ and $k = 2$, MOEA/D performs better than PLS for a very restricted budget $B_q = 200$, with an acceleration ratio of about 2.1 in favor of SMCO/W. With a high budget $B_q = 1500$, SMCO/W requires around 150 times less evaluations than both MOEA/D and PLS. For MUBQP, only PLS is able to hit all the targets. The acceleration ratio is again found to be substantial depending on the different targets and dimensions.

Generally speaking, the acceleration ratios presented in Fig. 7 indicate that the SMCO/W approach requires overall many order of magnitudes less expensive evaluations than its surrogate-less counterparts to attain a high-quality approximation set. Combined with the CPU computational times incurred by the Walsh surrogate from Table 7, these results provide a relatively clear picture on the benefits of using the Walsh-assisted SMCO/W approach when tackling a concrete optimization scenario having a computationally expensive objective vector.

## 6. Conclusion

In this paper, we have investigated the design and analysis of a modular surrogate-assisted framework for expensive multi-objective combinatorial optimization using Walsh basis. Based on extensive experiments, we provided a systematic study of the combined effects of different design components. In particular, we found that there is a non-trivial interaction between the strategy allowing to optimize the surrogate, and the strategy used for selecting the next solution to evaluate. We showed that a selection strategy based on the predicted improvements of candidate solutions with respect to a set of decomposed single-objective sub-problems is highly effective. Besides, we found that local search is to be preferred for the inner optimization of the discrete Walsh surrogate. Finally, we highlighted the importance of the Walsh order, and we proposed simple

dynamic strategies to accommodate the search to a range of black-box optimization functions having different degrees of difficulty.

We leave open a number of research issues are worthwhile exploring. For instance, we rely on the decomposition paradigm to structure the population of evaluated solutions, and to estimate which new offspring is the most promising for a true evaluation. It would be interesting to support different selection strategies based on other multi-objective paradigms. Another challenging issue is to deal with problems having a high number of objectives, as well as to address other discrete optimization domains, such as permutation problems, for which other single-objective surrogates exist in the specialized literature. A main question is then to study at what extent our findings hold for different objective and decision spaces.

Finally, the work conducted in this paper is of fundamental nature in the sense that we adopt a benchmark-oriented validation methodology, without targeting a specific real-word optimization setting. Considering a real-world application implies to deal with a number of additional and difficult questions. For instance, it could happen that for some complex applications, a restricted, yet not empty, subset of Walsh functions of possibly different high orders, are to be used so that the considered real-world problem can be fit accurately. This suggests to derive improved learning techniques to discover those functions without experiencing scalability issues at the training phase. Besides, the amount of computational ressources, in terms of parallel and distributed CPU cores, that are available in practice when tackling a real-world expensive application, can be substantial. This suggests that an ensemble of Walsh surrogates can be coordinated and trained in parallel, hence allowing one to sample an eventually large number of solutions that cover the Pareto front in a more effective way. It is our hope that the comprehensive study conducted in this paper will accelerate the establishment of a unified methodology for designing and analyzing discrete surrogate-assisted multi-objective optimization algorithms.

## Acknowledgement

This work was partially supported by the French National Research Agency (ANR-16-CE23-0013-01) and the Research Grants Council of Hong Kong (A-CityU101/16). We gratefully acknowledge the anonymous reviewers for their comments and suggestions that helped us improving our work.

## References

[1] J. Stork, M. Friese, M. Zaefferer, T. Bartz-Beielstein, A. Fischbach, B. Breiderhoff, B. Naujoks, T. Tusar, Open issues in surrogate-assisted optimization, in: T. Bartz-Beielstein, B. Filipic, P. Korosec, E. Talbi (Eds.), High-Performance Simulation-Based Optimization, Vol. 833 of Studies in Computational Intelligence, Springer, 2020, pp. 225–244.

[2] T. Chugh, K. Sindhya, J. Hakanen, K. Miettinen, A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms, Soft Computing 23 (9) (2019) 3137–3166.

[3] T. Bartz-Beielstein, M. Zaefferer, Model-based methods for continuous and discrete global optimization, Applied Soft Computing 55 (2017) 154–167.

[4] T. Bartz-Beielstein, A survey of model-based methods for global optimization, in: International Conference on Bioinspired Optimization Methods and Their Applications (BIOMA), 2016, pp. 1–18.

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

[6] Y. Jin, H. Wang, T. Chugh, D. Guo, K. Miettinen, Data-driven evolutionary optimization: An overview and case studies, IEEE Transactions on Evolutionary Computation 23 (3) (2019) 442–458.

[7] K. Deb, R. Hussein, P. C. Roy, G. T. Pulido, A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization, IEEE Transactions on Evolutionary Computation 23 (1) (2019) 104–116.

[8] D. Horn, T. Wagner, D. Biermann, C. Weihs, B. Bischl, Model-based multi-objective optimization: Taxonomy, multi-point proposal, toolbox and benchmark, in: Evolutionary Multi-Criterion Optimization, Springer International Publishing, 2015, pp. 64–78.

[9] R. Allmendinger, M. T. M. Emmerich, J. Hakanen, Y. Jin, E. Rigoni, Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case, Journal of Multi-Criteria Decision Analysis 24 (1-2) (2017) 5–24.

[10] K. Miettinen, Nonlinear multiobjective optimization, Vol. 12 of International series in operations research and management science, Kluwer, 1998.

[11] M. Ehrgott, Multicriteria optimization, 2nd Edition, Springer, 2005.

[12] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, 2001.

[13] C. A. Coello Coello, G. B. Lamont, D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd Edition, Springer, 2007.

[14] D. Krige, A statistical approach to some basic mine valuation problems on the Witwatersrand, Journal of the Southern African Institute of Mining and Metallurgy 52 (6) (1951) 119–139.

[15] C. E. Rasmussen, Gaussian processes in machine learning, in: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), Advanced Lectures on Machine Learning, ML Summer Schools, Canberra, Australia, February 2-14, Tübingen, Germany, August 4-16, Revised Lectures, Vol. 3176 of Lecture Notes in Computer Science, Springer, 2003, pp. 63–71.

[16] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, V. Vapnik, Support vector regression machines, in: M. Mozer, M. I. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems, NIPS, MIT Press, 1996, pp. 155–161.

[17] M. D. Buhmann, Radial Basis Functions - Theory and Implementations, Vol. 12 of Cambridge monographs on applied and computational mathematics, Cambridge University Press, 2009.

[18] I. J. Goodfellow, Y. Bengio, A. C. Courville, Deep Learning, Adaptive computation and machine learning, MIT Press, 2016.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Chapman and Hall/CRC, Monterey, CA, 1984.

[20] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.

[21] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A survey of multiobjective evolutionary algorithms based on decomposition, IEEE Transactions on Evolutionary Computation 21 (3) (2017) 440–462.

[22] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[23] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[24] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints, IEEE Transactions on Evolutionary

Computation 18 (4) (2014) 577–601.

[25] H. Jain, K. Deb, An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 602–622.

[26] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 181 (3) (2007) 1653–1669.

[27] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: Parallel Problem Solving from Nature - PPSN VIII, Lecture Notes in Computer Science, Springer, 2004, pp. 832–842.

[28] J. Knowles, ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Transactions on Evolutionary Computation 10 (1) (2006) 50–66.

[29] Q. Zhang, W. Liu, E. Tsang, B. Virginas, Expensive multiobjective optimization by MOEA/D with gaussian process model, IEEE Transactions on Evolutionary Computation 14 (3) (2010) 456–474.

[30] S. Zapotecas Martinez, C. A. Coello Coello, MOEA/D assisted by RBF networks for expensive multi-objective optimization problems, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2013, pp. 1405–1412.

[31] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, K. Sindhya, A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization, IEEE Transactions on Evolutionary Computation 22 (1) (2018) 129–142.

[32] R. Hussein, K. Deb, A generative kriging surrogate model for constrained and unconstrained multi-objective optimization, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2016, pp. 573–580.

[33] W. Ponweiser, T. Wagner, D. Biermann, M. Vincze, Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection, in: Parallel Problem Solving from Nature – PPSN X, Lecture Notes in Computer Science, Springer, 2008, pp. 784–794.

[34] J. Li, P. Wang, H. Dong, J. Shen, C. Chen, A classification surrogate-assisted multi-objective evolutionary algorithm for expensive optimization, Knowledge-Based Systems 242 (2022) 108416.

[35] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, Y. Jin, A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization, IEEE Transactions on Evolutionary Computation 23 (1) (2019) 74–88.

[36] H. Dong, J. Li, P. Wang, B. Song, X. Yu, Surrogate-guided multi-objective optimization (SGMOO) using an efficient online sampling strategy, Knowledge-Based Systems 220 (2021) 106919.

[37] N. Berveglieri, B. Derbel, A. Liefooghe, H. Aguirre, K. Tanaka, Surrogate-assisted multiobjective optimization based on decomposition: A comprehensive comparative analysis, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2019, pp. 507—-515.

[38] M. Zaefferer, Surrogate models for discrete optimization problems, Ph.D. thesis, University of Dortmund (2018).

[39] A. Moraglio, A. Kattan, Geometric generalisation of surrogate model based optimisation to combinatorial spaces, in: Evolutionary Computation in Combinatorial Optimization, Springer, 2011, pp. 142–154.

[40] M. Zaefferer, J. Stork, M. Friese, A. Fischbach, B. Naujoks, T. Bartz-Beielstein, Efficient global optimization for combinatorial problems, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2014, pp. 871–878.

[41] R. Baptista, M. Poloczek, Bayesian optimization of combinatorial structures (2018). arXiv:1806.08838.

[42] H. Dong, P. Wang, B. Song, Y. Zhang, X. An, Kriging-assisted discrete global optimization (KDGO) for black-box problems with costly objective and constraints, Applied Soft Computing 94 (2020) 106429.

[43] Q. Gu, Q. Wang, X. Li, X. Li, A surrogate-assisted multi-objective particle swarm optimization of expensive constrained combinatorial optimization problems, Knowledge-based Systems 223 (2021) 107049.

[44] Q. Gu, Q. Wang, N. N. Xiong, S. Jiang, L. Chen, Surrogate-assisted evolutionary algorithm for expensive constrained multi-objective discrete optimization problems, Complex & Intelligent Systems 8 (2021) 2699–2718.

[45] R. Sun, Q. Duan, X. Mao, A multi-objective adaptive surrogate modelling-based optimization algorithm for constrained hybrid problems, Environmental Modelling & Software 148 (2022) 105272.

[46] R. G. Regis, A two-phase surrogate approach for high-dimensional constrained discrete multi-objective optimization, in: The Genetic and Evolutionary Computation Conference, GECCO Companion Volume, ACM, 2021, pp. 1870–1878.

[47] V. Drouet, S. Verel, J.-M. Do, Surrogate-assisted asynchronous multiobjective algorithm for nuclear power plant operations, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2020, pp. 1073–1081.

[48] J. Hakanen, J. Malmberg, V. Ojalehto, K. Eyvindson, Data-driven interactive multiobjective optimization using a cluster-based surrogate in a discrete decision space, in: International Conference on Machine Learning, Optimization, and Data Science, Lecture Notes in Computer Science, Springer, 2018, pp. 104–115.

[49] R. G. Regis, High-dimensional constrained discrete multi-objective optimization using surrogates, in: International Conference on Machine Learning, Optimization, and Data Science, Springer, 2020, pp. 203–214.

[50] H. Wang, Y. Jin, A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems, IEEE Transactions on Cybernetics 50 (2) (2020) 536–549.

[51] Y. Tian, S. Yang, L. Zhang, F. Duan, X. Zhang, A surrogate-assisted multiobjective evolutionary algorithm for large-scale task-oriented pattern mining, IEEE Transactions on Emerging Topics in Computational Intelligence 3 (2) (2019) 106–116.

[52] F. Leprêtre, C. Fonlupt, S. Verel, V. Marion, Combinatorial surrogate-assisted optimization for bus stops spacing problem, in: International Conference on Artificial Evolution (Evolution Artificielle), Springer, 2020, pp. 42–52.

[53] M. Rieser, Adding transit to an agent-based transportation simulation: Concepts and implementation, Ph.D. thesis, der Technischen Universitat Berlin (06 2010). doi:10.14279/depositonce-2581.

[54] J. P. Romero, A. Ibeas, J. L. Moura, J. Benavente, B. Alonso, A simulation-optimization approach to design efficient systems of bike-sharing, Procedia - Social and Behavioral Sciences 54 (2012) 646–655, Proceedings of EWGT2012 - 15th Meeting of the EURO Working Group on Transportation, Paris.

[55] D. M. Negoescu, P. I. Frazier, W. B. Powell, The knowledge-gradient algorithm for sequencing experiments in drug discovery, INFORMS Journal on Computing 23 (3) (2011) 346–363.

[56] S. Vérel, B. Derbel, A. Liefooghe, H. E. Aguirre, K. Tanaka, A surrogate model based on walsh decomposition for pseudo-boolean functions, in: Parallel Problem Solving from Nature - PPSN XV, Vol. 11102 of Lecture Notes in Computer Science, Springer, 2018, pp. 181–193.

[57] F. Leprêtre, S. Verel, C. Fonlupt, V. Marion, Walsh functions as surrogate model for pseudo-boolean optimization problems, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2019, pp. 303–311.

[58] A. Dushatskiy, A. M. Mendrik, T. Alderliesten, P. A. Bosman, Convolutional neural network surrogate-assisted GOMEA, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2019, pp. 753–761.

[59] A. Deshwal, S. Belakaria, J. R. Doppa, Bayesian optimization over hybrid spaces, in: International Conference on Machine Learning, PMLR, 2021, pp. 2632–2643.

[60] K. Swingler, Learning and searching pseudo-boolean surrogate functions from small samples, Evolutionary Computation 28 (2) (2020) 317–338.

[61] L. Han, H. Wang, A random forest assisted evolutionary algorithm using competitive neighborhood search for expensive constrained combinatorial optimization, Memetic Computing 13 (2021) 19–30.

[62] F. Leprêtre, C. Fonlupt, S. Verel, V. Marion, Combinatorial surrogate-assisted optimization for bus stops spacing problem, in: International Conference on Artificial Evolution (Evolution Artificielle), Springer, 2019, pp. 42–52.

[63] A. Dushatskiy, T. Alderliesten, P. A. Bosman, A novel approach to designing surrogate-assisted genetic algorithms by combining efficient learning of Walsh coefficients and dependencies, ACM Transactions on Evolutionary Learning and Optimization 1 (2) (2021) 1–23.

[64] I. Unanue, M. Merino, J. A. Lozano, A general framework based on Walsh decomposition for combinatorial

optimization problems, in: Congress on Evolutionary Computation (CEC), IEEE, 2021, pp. 391–398.
[65] A. Deshwal, S. Belakaria, J. R. Doppa, A. Fern, Optimizing discrete spaces via expensive evaluations: A learning to search framework, in: The AAAI Conference on Artificial Intelligence, AAAI Press, 2020, pp. 3773–3780.
[66] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: Learning and Intelligent Optimization, Springer, 2011, pp. 507–523.
[67] J. L. Walsh, A closed set of normal orthogonal functions, American Journal of Mathematics 45 (1) (1923) 5.
[68] G. Pruvost, B. Derbel, A. Liefooghe, S. Verel, Q. Zhang, Surrogate-assisted multi-objective combinatorial optimization based on decomposition and Walsh basis, in: The Genetic and Evolutionary Computation Conference, GECCO, ACM, 2020, pp. 542–550.
[69] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 284–302.
[70] A. D. Bethke, Genetic algorithms as function optimizers, Ph.D. thesis, University of Michigan (1980).
[71] T. Hastie, R. Tibshirani, M. Wainwright, Statistical Learning with Sparsity: The Lasso and Generalizations, 1st Edition, Chapman and Hall/CRC, 2015.
[72] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society: Series B (Methodological) 58 (1) (1996) 267–288.
[73] L. Paquete, T. Stützle, A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices, European Journal of Operational Research 169 (3) (2006) 943–959.
[74] H. Aguirre, K. Tanaka, Working principles, behavior, and performance of MOEAs on MNK-landscapes, European Journal of Operational Research 181 (3) (2007) 1670–1690.
[75] S. Verel, A. Liefooghe, L. Jourdan, C. Dhaenens, On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives, European Journal of Operational Research 227 (2) (2013) 331–342.
[76] A. Liefooghe, S. Verel, J.-K. Hao, A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming, Applied Soft Computing 16 (2014) 10–19.
[77] G. Kochenberger, J.-K. Hao, F. Glover, M. Lewis, Z. Lü, H. Wang, Y. Wang, The unconstrained binary quadratic programming problem: A survey, Journal of Combinatorial Optimization 28 (1) (2014) 58–81.
[78] S. A. Kauffman, The Origins of Order, Oxford University Press, 1993.
[79] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. Grunert da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.
[80] N. Hansen, A. Auger, S. Finck, R. Ros, Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup, Research Report RR-6828, INRIA (2009).