PUBOi: A Tunable Benchmark with Variable Importance

Sara Tari, Sébastien Verel, Mahmoud Omidvar

PUBO$_i$: a tunable benchmark with variable importance

Sara Tari[1], Sébastien Verel[1], and Mahmoud Omidvar[1]

[1] Univ. Littoral Côte d'Opale, UR 4491, LISIC, Laboratoire d'Informatique Signal et Image de la Côte d'Opale, F-62100 Calais, France,
{sara.tari,verel,omidvar}@univ-littoral.fr

Abstract. In this work, we present the benchmark generator PUBO$_i$, Polynomial Unconstrained Binary Optimization, that combines subproblems to create instances of pseudo-boolean optimization problems. Any mono-objective pseudoboolean functions including existing classical optimization problems can be expressed with Walsh functions. The benchmark generator can tune main features of problems such as problem dimension, non-linearity degree, and neutrality. Additionally, to be able to create instances with properties similar to those of real-like combinatorial optimization problems, the goal of PUBO$_i$ is to introduce the notion of variable importance. Indeed, the importance of decision variables can be tuned using three benchmark parameters. In the version presented here, we consider four subproblems already used in Chook generator for benchmarking quantum computers and algorithms as a basis. We also present the impact of benchmark parameters using a fitness landscape analysis that empirically shows these parameters to significantly impact the variable importance.

Keywords: Benchmark · Fitness landscape · Walsh function · Variable importance.

1 Introduction

There exist numerous evolutionary algorithms, or local search algorithms that efficiently tackle combinatorial optimization problems. One main practical, and theoretical difficulty facing new problem instances is the selection, or the design of efficient optimization algorithm according to the properties of the instance to optimize. Studying such algorithms often require to run them on diverse problem instances, which is usually done successfully in benchmarking studies [1]. This paper attempts to propose a benchmark of pseudo-Boolean functions with tunable relevant properties.

In conjunction of benchmarking efforts, a powerful approach aiming to improve the understanding of optimization algorithms, and thus determine which approach to consider, consists of using fitness landscapes [28]. In evolutionary computation, a fitness landscape represents the search space and fitness function of a given instance according to the links between solutions induced by the

neighborhood relation of the algorithm under consideration. This representation corresponds to a graph from which several characteristics can be estimated by the application of indicators on a sample of solutions [14, 15], allowing a visualization of landscapes and features that induce challenges for optimization methods. Hence, fitness landscapes is a powerful tool to better guide the design and use of neighborhood-based algorithms, even on large landscapes.

To design new optimization algorithms by "hand" or using machine learning techniques, to test existing approaches, or to better understand problem difficulty using fitness landscape analysis often requires a large and diverse set of problem instances with relevant properties. To this aim, there exist several generators or benchmarks of academic problems. While a small subset of such instances are based on real-world data, most of them are randomly generated, with characteristics that could greatly differ from real-world problems. One of the peculiarities of real-like instances is that, contrary to instances generated using random-based techniques, they are structured. Thus some variables of the problems play a more important role in terms of fitness contribution, and can be interdependent to several other variables, making them harder to study and understand [12]. Although variable importance of such problems must be studied, existing benchmark generators do not allow users to tune this parameter, which contributes to the lack of instances having important variables.

In this work, we propose and present PUBO$_i$, a benchmark in which variable importance is tunable, and thus impacts the instance structure. PUBO$_i$ is based upon Walsh functions [25], an orthogonal basis of pseudo-boolean functions for representing any pseudo-boolean function. These functions arouse a strong interest in several scientific communities. In quantum physics, one can use these functions to create benchmarks representing optimization problems that can be tackled using new quantum computers [16]. They are successfully used in black-box combinatorial optimization as surrogate functions for computational-costly fitness functions [24]. Walsh functions also allow a standardized reformulation of various academic optimization problems [7]. This basis of functions can be used for benchmarking purposes, by generating instances with various characteristics, including characteristics similar to those of real-like instances such as the variable importance. The contribution of this paper are (1) the proposition of a new benchmark generator with tunable variable importance, (2) a fitness landscape analysis of the instances set aiming to analyze the impact of benchmark parameters on the shape of fitness landscape, and on the related problem difficulty.

The paper is organized as follows. The next section presents related works on benchmarking and variable importance. Section 3 is devoted to the description of PUBO$_i$ as well as the instances considered in this study. In section 4, we present the methodology and results of our experiments. The last section provides a discussion of this work and points out future work directions.

2 Related Works

In this section we propose a brief overview of previous work on variable importance, test suites for optimization problems and benchmark generators, with a particular focus on a generator proposed by physicists we used as a basis for PUBO$_i$.

2.1 Variable importance, and benchmarks of optimization problems

In the context of optimization problems, we assume a variable is important when its mutation strongly affects the fitness of a solution. When this characteristic is present on instances to optimize, taking it into account can help to optimize the problem more efficiently. There exist a few studies of optimization algorithms focusing on the variable importance of optimization problems, although this characteristic often exists on real data. For example, in [17, 18] the authors study a machine learning-enhanced recombination that incorporates an intelligent variable selection method for multi-objective optimization and show that taking the variable importance into consideration can improve the optimization quality. Another example concerns the fitness landscape analysis of landscapes from the SIALAC benchmark, a benchmark for mobility problems [12]. In this benchmark based on a city mobility simulation, variables have different degrees of importance, and a bandit descent heuristic is proposed to take important variables into account.

An efficient way of studying optimization algorithms consists of analyzing their behavior on various problems and instances. Over the past decades, a huge effort has been devoted to the proposition of benchmarks of various optimization problems. Benchmarks allow the community to focus on the optimization of a given problem and to study more efficiently various optimization problems. Among classic benchmarks for academic combinatorial optimization problems, Taillard [20] proposes instances covering three basic scheduling problems : the permutation flow shop, the job shop and the open shop scheduling problems. Despite their simplicity compared to real-life scheduling instances, many of these instances are still considered challenging and are widely used to study the flow-shop scheduling problem, both in terms of optimization methods and fitness landscape analysis. Another widely-used testbed is the QAPLIB, that provides several instances of the Quadratic Assignment Problem (QAP) [2], as QAP is a non-trivial problem for which a lot of effort has been devoted. QAPLIB comprises randomly-generated instances as well as instances based on real-life data, such as the testing of self-testable sequential circuits or the flow of patients between different facilities in a hospital. DIMACS instances for the graph coloring problem and maximal clique problem are provided in [11]. The families of provided graphs include random graphs, flat graphs or latin square graphs. While this list of instances libraries is not exhaustive, such libraries focus on a single problem or a small set of the same family of problems, and rarely propose a sufficient number of instances differing in variable importance.

In the field of continuous optimization, testbeds are regularly proposed for the GECCO workshop on Black-Box Optimization Benchmarking (BBOB) [9].

These testbeds can be found on the widely-recognized benchmarking software COCO [10] for black-box optimization. In addition to a noisy suite, COCO has been recently extended to multiobjective [23] and mixed integer [22] problems.

Recently, a benchmarking platform *IOHprofiler* has been proposed for evaluating the performance of iterative optimization heuristics [5]. One of its components, *IOHexperimenter* aims to generate benchmark suites for the benchmarking of such optimization methods. IOHexperimenter covers 23 real-valued (continuous), as well as 25 academic pseudo-boolean problems [5].

The aforementioned platforms implement a large panel of problems and are widely-used. These problems are either continuous or present a uniform structure, in which the weight of variables is generally uniform. While such platforms are particularly useful to study heuristic optimization methods, these studies are conducted on structures that are more common on academic problems than on real-ones, and thus cannot efficiently focus on the peculiarities induced by important variables.

2.2 Tile Planting instances: Chook generator

Perera *et al.* [16] proposed *Chook* a python-based generator that generates instances for the Tile Planting (TP) problem [8]. The goal of this benchmark is to test new quantum computers and algorithms. Indeed, combinatorial problems that can be considered efficiently with quantum computers are expressed as spin-glasses problems, and even more particularly as Quadratic Unconstrained Binary Optimization problems (QUBO). TP problems are a benchmark of pseudo-boolean objective functions (binary string search space) with known global minima, the planting solutions. These functions are determined by a sum of subfunctions. In the most general form, the fitness function (Hamiltonian) is:

$$\mathcal{H}(s) = \sum_{j \in V} h_j s_j + \sum_{k=2}^{n} \sum_{(i_1,\ldots,i_k) \in E} J_{i_1,\ldots,i_k} s_{i_1} s_{i_2} \ldots s_{i_k}$$

with $s = (s_1, s_2, \ldots, s_n)$, $s_i \in \{-1, 1\}$ called spins, and where the hypergraph $G = (V, E)$ with vertices V and edges E describes the interaction between the problem spins (variables). The coefficients $h_j \in \mathbb{R}$ define the local external field (energy between the environment and each spin), and the coefficients $J \in \mathbb{R}$ the intensity of each interaction. Notice that this equation can be mapped into boolean variables $x_i \in \{0, 1\}$ using the classical transformation $x_i = \frac{1}{2}(1 - s_i)$. In most cases these problems can be expressed using only 2-local interactions between spins with a graph G:

$$\mathcal{H}(s) = \sum_{j \in V} h_j s_j + \sum_{(i,j) \in E} J_{ij} s_i s_j$$

In TP instances [8], the problem graph is decomposed into edge-disjoint and vertex-sharing subgraphs. For a decomposition of the graph $G = (V, E)$ into subgraphs $\{G_l = (V_l, E_l)\}$ such that no edges are shared among the subgraphs,

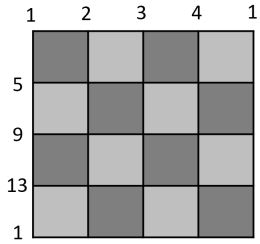

Fig. 1. Depiction of the graph for a 4×4 Tile Planting problem instance.

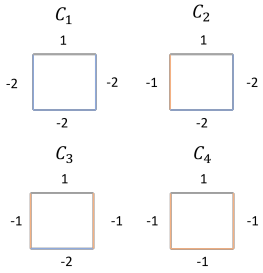

Fig. 2. Description of the four sub-function classes for the Tile Planting in Chook.

the coefficients h_i are equal to 0, and each subgraph is associated to a quadratic energy function: $\mathcal{H}_l(s) = \sum_{(i,j)\in E_l} J_{ij} s_i s_j$

The energy function of the tile planting problem can then be expressed as the sum of energy functions of each sub-graph: $\mathcal{H} = \sum_l \mathcal{H}_l$

The sub-functions $\mathcal{H}_l$ are designed to share a common ground state, *i.e.* the lowest-energy state of a quantum-mechanical system, which corresponds to the ferromagnetic ground state $s = (+1, +1, \ldots, +1)$. Thus by the additive property of $\mathcal{H}$, the Tile Plating problems also have the same ferromagnetic ground state.

TP problems use a regular lattice structure that allows a decomposition of the graph G which contains a subset of the unit cells as subgraphs. In the square lattices version of the problem, the resulting unit-cells form a checkboard pattern (see Fig. 1) and the problem graph corresponds to a toric square matrix. A portfolio of four sub-function classes $\{C_1, C_2, C_3, C_4\}$ (see Fig. 2) are defined as follows: $C_1(s) = -2s_0s_1 - 2s_1s_2 - 2s_2s_3 + s_3s_0$; $C_2(s) = -2s_0s_1 - 2s_1s_2 - s_2s_3 + s_3s_0$; $C_3(s) = -2s_0s_1 - s_1s_2 - s_2s_3 + s_3s_0$; $C_4(s) = -s_0s_1 - s_1s_2 - s_2s_3 + s_3s_0$. The portfolio is designed such that each function class C_j has j local minima[1]. Individually, function C_4 is supposed to be more difficult to solve than C_1.

In Chook, instances are generated by (1) assigning a sub-function class to each subgraph in the problem, (2) a random rotation of the plaquette in the lattice. Instance classes are defined using a probability distribution over the sub-function classes: p_i the probability of selecting sub-functions from class C_i such that $\sum_i p_1 = 1$. Generating Tile Planting problems with Chook requires to set 4 parameters: the problem dimension n, and the probabilities p_1, p_2, and p_3. Perera *et al.* [16] analyze the performance of quantum algorithms (simulated quantum annealing) according to the main benchmark parameters p_i which can tune the problem difficulty. However in TP instances, the square toric lattice gives an identical role to binary variables. Moreover, the 2d shape of variables interactions allows the solving of such instances in polynomial time complexity [6].

[1] Indeed, j pairs of symmetric local minima

3 PUBO$_i$ problems

Here, we propose PUBO$_i$ for Polynomial Unconstrained Binary Optimization with importance, a generator in which benchmarks are expressed as Walsh functions, and takes the importance of variables into account. This version uses the same subproblems as in the Tile Planting generator, whose benchmarks correspond to Walsh functions of order 2. These subproblems are straightforward, which allows us to focus on our aim: to tune the variable importance to create more structured instances. Therefore, we focus on the introduction of three new parameters dedicated to this aim. In the following, we provide a mathematical description of Walsh functions. Then we present the first version of PUBO$_i$, as well as the benchmarks considered in our experiments.

3.1 PUBO problems

Quadradratic Unconstrained Binary Optimization (QUBO), also known as Unconstrained Binary Quadratic Problem (UBQP) in the context of combinatorial optimization [13], are well known pseudo-boolean functions in the field of physics [4]. These functions are quadratic functions that can be generalized to any order. Although there exists several names, we denote this extension as PUBO for Polynomial Unconstrained Binary Optimization [7]. Notice that this extension is also known as spin glasses problems in physic. Several construction of PUBO exists, here we choose to present PUBO from the point of view of Walsh functions.

Given a bit string of dimension n, Walsh functions compose a finite set of 2^n pseudo-boolean functions defined for all integer k from $[0, 2^n - 1]$ by:

$$\begin{cases} \varphi_k : \{0,1\}^n \to \{-1,1\} \\ \qquad\qquad x \mapsto (-1)^{\sum_{i=0}^{n-1} k_i x_i} \end{cases}$$

where x_i is the ith bit of x, and k_i is the ith bit representing the integer k. Walsh functions [25] is an orthonormal basis of pseudo-boolean functions. Any pseudo-boolean function $f : \{0,1\}^n \to \mathbb{R}$ can be written as: $f(x) = \sum_{k=0}^{2^n} w_k \varphi_k(x)$ with $w_k \in \mathbb{R}$. Indeed, Walsh transform is similar to Fourier transformation, and coefficients w_k are computed using the orthogonal projection for the L_2-norm of f on functions φ_k: $w_k = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)\varphi_k(x)$.

Walsh functions allow a decomposition equivalent to a polynomial decomposition. Indeed, using the transformation of bit $x_i \in \{0,1\}$ into spin $s_i \in \{-1,1\}$ defined by $s_i = (-1)^{x_i}$, the kth Walsh function can be rewritten as $\varphi_k(s) = \prod_{i:k_i=1} s_i$. Thus, the order of a Walsh function φ_k is defined by the number of bits equal to 1 in the binary representation of k. For example, an order 2 Walsh function written as: $w_0^{(0)} + \sum_{i=0}^{n} w_i^{(1)} s_i + \sum_{i<j} w_{i,j}^{(2)} s_i s_j$

For benchmarking purposes, in addition to being able to represent any evaluation function, Walsh functions allow to fine-tune both the interdependence between the variables (non-zero terms of the polynomial) and the intensity of the interactions ($|w_k|$ values). Using integer numbers for w_k, neutrality levels (plateaus) can also be tuned.

3.2 Definition of PUBO$_i$

The set of PUBO functions is a vector space of dimension 2^n where n is the bit string length. A benchmark based on PUBO has to define a subspace from this large vector space. The proposed benchmark PUBO$_i$ is defined with two principles. In order to reduce the dimension, following the TP benchmark design principle, PUBO$_i$ decomposes the objective function into a sum of sub-functions from a portfolio. In addition, the binary variables in each sub-function are selected according to tunable parameters of variable importance, to introduce a real-like property. As Tile Planting instances, PUBO$_i$ defines the objective as a sum of sub-functions:

$$\forall x \in \{0,1\}^n, \quad f(x) = \sum_{i=1}^{m} f_i(x)$$

where m is the number of sub-functions, and each sub-function f_i is selected at random according to probabilities p_j of each sub-function class C_j from the portfolio (see Sect. 2.2).

The main originality of the proposed benchmark lies around the notion of variable importance. Each sub-function depends on a limited number of variables. For example in TP instances, each sub-function depends on 4 variables selected according to the square lattices. In PUBO$_i$, variables are selected according to a degree of importance. Indeed, intuitively in real-world problems, important variables should appear more often in sub-problems than least important variables. To define the importance of variables, the set of variables $X = \{x_1, \ldots, x_n\}$ is split into k disjoint classes of importance: $c_i \subset X$ such that $\cup_k c_k = X$, and $c_i \cap c_j = \emptyset$ for each pair $\{i, j\}$. The number n_i of variables in each class is a parameter of the benchmark. Each class of importance c_i has a degree of importance $d_i \in \mathbb{R}^+$. In PUBO$_i$, the probability of selecting a variable in a sub-function is then proportional to the degree of variable importance of its class i: $p_{c_i} = \frac{d_i}{\sum_{j=1}^{k} d_j}$.

We also introduce another parameter to tune the co-appearance of variables in the same sub-function. In a real-world problem, one can assume important variables are not randomly distributed among the sub-problems. For some problem instances, important variables could appear together in the same sub-problems, and for other ones, important variables could be linked to less important ones. The following paragraph introduces the principle of co-appearance of important variables in PUBO$_i$.

For a sub-function of arity a, when classes of importance are independent, the probability of having variables of classes $c_{i_1}, \ldots, c_{i_a}$ is the product of probabilities $p_{c_{i_1}} \ldots p_{c_{i_a}}$. To tune this probability differently, the probability related to the number of each class is necessary. For simplification purposes, let us first suppose there are only 2 classes of importance (0 and 1). We denote $p_i^{(a)}$ the probability of having i variables of class 1 in the same sub-function of arity a. Then, we can define this probability recursively. For arity $a = 1$, $p_0^{(1)} + p_1^{(1)} = 1$. Thus,

$p_0^{(1)} = p_{c_0}$, and $p_1^{(1)} = 1 - p_{c_0}$. The probability to select the class 0 for each sub-function variable should remain the same, hence the additional variable for arity $a = 2$ should not change the marginal probability, and we have the following equations:

$$\begin{cases} p_0^{(2)} + p_1^{(2)} & = p_0^{(1)} \\ p_1^{(2)} + p_2^{(2)} & = p_1^{(1)} \end{cases}$$

The introduction of a new parameter p'_{c_0} to tune the different probabilities is possible. By setting $p_0^{(2)} = p'_{c_0} p_0^{(1)}$, we obtain:

$$\begin{cases} p_0^{(2)} = p'_{c_0} p_{c_0} \\ p_1^{(2)} = (1 - p'_{c_0}) p_{c_0} \\ p_2^{(2)} = (1 - p'_{c_0})(1 - p_{c_0}) + p'_{c_0} - p_{c_0} \end{cases}$$

When $p'_{c_0} = p_{c_0}$, the classes of importance of each variable are independent. When p'_{c_0} is greater than p_{c_0}, then the probability to have both variables in the same class of importance is higher. On the contrary, when p'_{c_0} is lower than p_{c_0}, the sub-functions have more heterogeneous classes of importance.

More generally, an additional variable in the sub-function leads to the recurrence formula: $\forall a \geq 2, \forall i \in \{0, \ldots, a\},\ p_i^{(a)} + p_{i+1}^{(a)} = p_i^{(a-1)}$

By setting the first probability $p_0^{(a)}$, one can deduce all other probabilities. It would be possible to introduce a parameter at each arity level: $p_0^{(a)} = p'^{(a)}_{c_0} p_0^{(a-1)}$. However, to simplify the benchmark tuning, we use one single parameter p'_{c_0}: $p_0^{(a)} = p'_{c_0} p_0^{(a-1)}$. Therefore we obtain the following values:

$$\begin{cases} p_0^{(a)} = (p'_{c_0})^{a-1} p_{c_0} \\ \quad \ldots \\ p_i^{(a)} = (1 - p'_{c_0})^i (p'_{c_0})^{a-1-i} p_{c_0} \\ \quad \ldots \\ p_a^{(a)} = (1 - p'_{c_0})^{a-1}(1 - p_{c_0}) + (1 - (1 - p'_{c_0})^{a-1})(1 - \frac{p_{c_0}}{p'_{c_0}}) \end{cases}$$

As for the arity 2, the value of p'_{c_0} determines the independence degree of co-appearance of the same class. Let us rewrite the parameter $p'_{c_0} = \alpha\ p_{c_0}$. When $\alpha = 1$, the co-appearance of variables importance classes are independent. When $\alpha > 1$, variables from the same class of importance have a higher probability to appear in the same sub-function, and conversely when $\alpha < 1$. Remember that globally on all sub-functions, each class of importance appears with a probability proportionally to its degree of importance. To extend to more than 2 classes of importance, we have to consider iteratively the probability to be in class 0, and not to be in class 0, then probability to be in class 1, and in a class greater than 1, etc.

We also propose to shift randomly the global minimum of each sub-function class. Naturally, knowing the global minimum could be useful, yet it would introduce a bias. Indeed, when a sub-problem is solved with the ground state 1111,

it also helps the search to solve another sub-problem (shared variables are set to the optimal value). In PUBO$_i$, we propose to shift to a random binary string the minimum for each sub-function to design more challenging instances compared to those of Tile Planting. Of course, the knowledge of ground state is lost and should be approximated by the best known solution for each instance. However, a lower bound of minima can be computed by summing the minima value of each sub-function. Tab. 1 summarizes the parameters of PUBO$_i$ benchmark.

The code of the PUBO$_i$ generator is available on git `https://gitlab.com/verel/pubo-importance-benchmark`. The generator produces a file in json format which follows the same format of Chook generator. As a consequence, all solvers (quantum or classical) can be used to solve PUBO$_i$ instances.

Table 1. Parameters of PUBO$_i$ benchmark.

Parameter	Description	Experimental values
n	Problem dimension	$[1000, 5000]$
m	Number of sub-functions	$[0.01, 0.2] \times \frac{n(n-1)}{2}$
$\mathcal{C}$	Portfolio of sub-functions	Tile Planting
p_i	Probabilities of sub-function class	$[0, 1]$
k	Number of class of variable importance	2
n_i	Number of variables in each class of importance	$n_0 = 0.25n$, $n_1 = n - n_0$
d_i	Degree of importance of each class	$d_0 \in [1, 10]$, $d_1 = 1$
α	Probability of importance class co-appearance	$[1, 1/(p_{c_0} - 1)]$

3.3 Instances set

This instances set aims to provide a large set of PUBO$_i$ instances for the analysis of the benchmark parameters impact on the properties of the problems, but also to offer a diverse set instances to train, and test new efficient optimization algorithms for this type of problems, and real-world problems.

In this set of instances, we consider $k = 2$ classes of importance to distinguish important variables, and non-important variables. The number of important variables is set to 25% of the total number, and as a consequence, 75% of variables are considered non-important. The degree of importance of non-important variables is always set to $d_1 = 1$, while the degree of important variables is higher from the range $[1, 10]$. Important variables could be up to 10 times more frequent than non-important ones. The factor α is set to be larger than 1 up to the larger possible $\frac{1}{p_{c_0}-1}$. The problem dimension n is between 1000, and 5000 which is larger than in the TP set, and medium to large size according to the UBQP standard [13]. The number of sub-functions m is proportional to the square of problem dimension. Indeed, the portfolio of PUBO$_i$ is the one of TP which defines functions with 4 quadratic Walsh terms. Following the UBQP standard of matrix density, we choose m between 1% and 20% of $\frac{n(n-1)}{2}$. Tab. 1 summarizes the range of parameters value.

Instead of factorial design of experiments, we generate 1000 instances of PUBO_i using *Latin hypercube sampling* [3] (LHS), a statistical method for the quasi-random sampling based on a multivariate probability distribution. Thus, in our context, LHS leads to a set of instances that covers a large panel of different parameter combination while significantly reducing the computational effort devoted to the study of such parameters. Indeed, the factorial design alternative either leads to a poorer coverage of the possible combinations or require the consideration of more instances, leading to a tedious process and a significantly higher computation cost. To respect the constraint $p_1 + p_2 + p_3 \leq 1$ to define, we reject samples from LHS which do not respect this constraint in order to avoid scaling bias. All instance files in json format are available online at the url `https://gitlab.com/verel/pubo-importance-benchmark/instances`.

4 Experimental analysis

This set of experiments aims to highlight whether the parameters we propose impact variable importance, and how variable importance impacts some classic features of fitness landscapes.

4.1 Fitness landscapes characterization

The fitness landscape analysis mostly focuses on classic features: the ruggedness/multimodality and the neutrality. The ruggedness, or multimodality of a landscape mainly refers to the number of local optima, their distribution, and the size of their basins of attraction. This property reflects the difficulty to optimize the instance with local search algorithm based on given neighborhood relation. A *rugged* landscape has several peaks (local optima) hard to attain (small basins of attraction), whereas a *smooth* landscape has a few peaks easy to attain (large basins of attraction).

Here, we use a widely-used ruggedness indicator, the fitness *autocorrelation* [26]. Given a random walk $(x_t, x_{t+1}, \ldots)$ where the solution x_{i+1} is a neighbor of x_i, the autocorrelation function ρ of the fitness function f corresponds to autocorrelation function of the time series $(f(x_t), f(x_{t+1}), \ldots)$:

$$\rho(\ell) = \frac{E[f(s_t)f(s_{t+\ell})] - E[f(s_t)]E[f(s_{t+\ell})]}{var[f(s_t)]}$$

where $E[f(s_t)]$ is the expected value of $f(s_t)$, and $var[f(s_t)]$ its variance. In the following, we only consider the first coeffficient of autocorrelation $\rho(1)$, as it usually is enough to summarize the ruggedness levels of landscapes.

The neutrality of a fitness landscape refers to the proportion of neutral neighbors in the landscape. A neutral neighbor of a solution has the same fitness value. Neutrality often induces plateaus in the landscape, in which optimization methods can easily be trapped, preventing them to attain better-quality solutions. Thus, a fitness landscape with high neutrality levels can be challenging to optimize. We use the neutrality degree [19] on neutral moves performed during the

random walks $(x_t, x_{t+1}, \ldots)$: $deg_n(x) = \#\{f(x_{i+1}) = f(x_i) \ : \ i \in \{0, 1, \ldots\}\}$. More precisely, we report the neutral mutation probability $\frac{deg_n(x)}{n}$ which is independent of problem dimension.

The last measure considered for the fitness landscape analysis is the length of the adaptive walk. An adaptive walk is a sequence of neighboring and improving solutions: $(x_0, x_1, \ldots, x_\ell)$ such for all $i \in \{0, \ldots, \ell\}$, x_{i+1} is a neighbor of x_i, and $f(x_{i+1})$ is better than $f(x_i)$. Such a measure highlights the multimodality (i.e., the presence of several peaks) of fitness landscapes. A short walk is generally the sign of a multimodal landscape, whereas a long walk usually indicates a monomodal landscape. Several adaptive walks exist, and on combinatorial fitness landscape hill-climbers are often considered to this aim. Here, we use a first improvement hill-climber which consists of randomly selecting an improving neighbor at each step of the search, until a local optimum is met. The length of adaptive walk is then the number of steps ℓ.

4.2 Methodology

We consider the set of 1000 instances generated with LHS. For each instance, we conducted a run of random walk of length $30n$. This length is sufficient to provide a sample of solutions, as each variable is flipped 30 times on average, and allows to compute statistics for each variable. We consider a classic neighborhood-relation of pseudo-boolean optimization problems, 1-flip. 30 adaptive walks are conducted for each landscape, starting from a randomly generated solution, and stopping on a local optimum. The average length of adaptive walks is reported. As our goal is to observe the impact of benchmark parameters, and in particular variable importance on landscape characteristics, we use the mgvc R package [27] to generate *generalized additive models* (GAMs) to highlight the significance of these parameters on the variable importance. GAMs are statistical models that merge the properties of the generalized linear model with those of the additive model. These results are presented through scatterplots of the regression model between landscape features and benchmark parameters.

4.3 Results

Fig. 3 shows the scatter plot with the GAM model regression of landscape features, autocorrelation coefficient, the neutral mutation probability, and the length of adaptive walks (see Sect. 4.1), according to the benchmark parameters. Each plot reports the *effective degrees of freedom* (edf) of GAM, which reflects the degree of non linearity: $edf = 1$ corresponds to a linear relationship, $1 < edf \leq 2$ is a quadratic, or weakly non-linear relationship and $edf > 2$ a highly non-linear relationship [29]. The significance values (Signif.) are also reported. The asterisks indicates the p-value at different levels of the statistical significance: '***', '**', '*', '.', and '-' correspond respectively to p-value levels of 0, 0.001, 0.01, 0.1, 0.05, and 1.

The problem dimension (n), the number of sub-functions (m), the degree of importance, and the probability p_1 of choosing the sub-problem C_1 have a

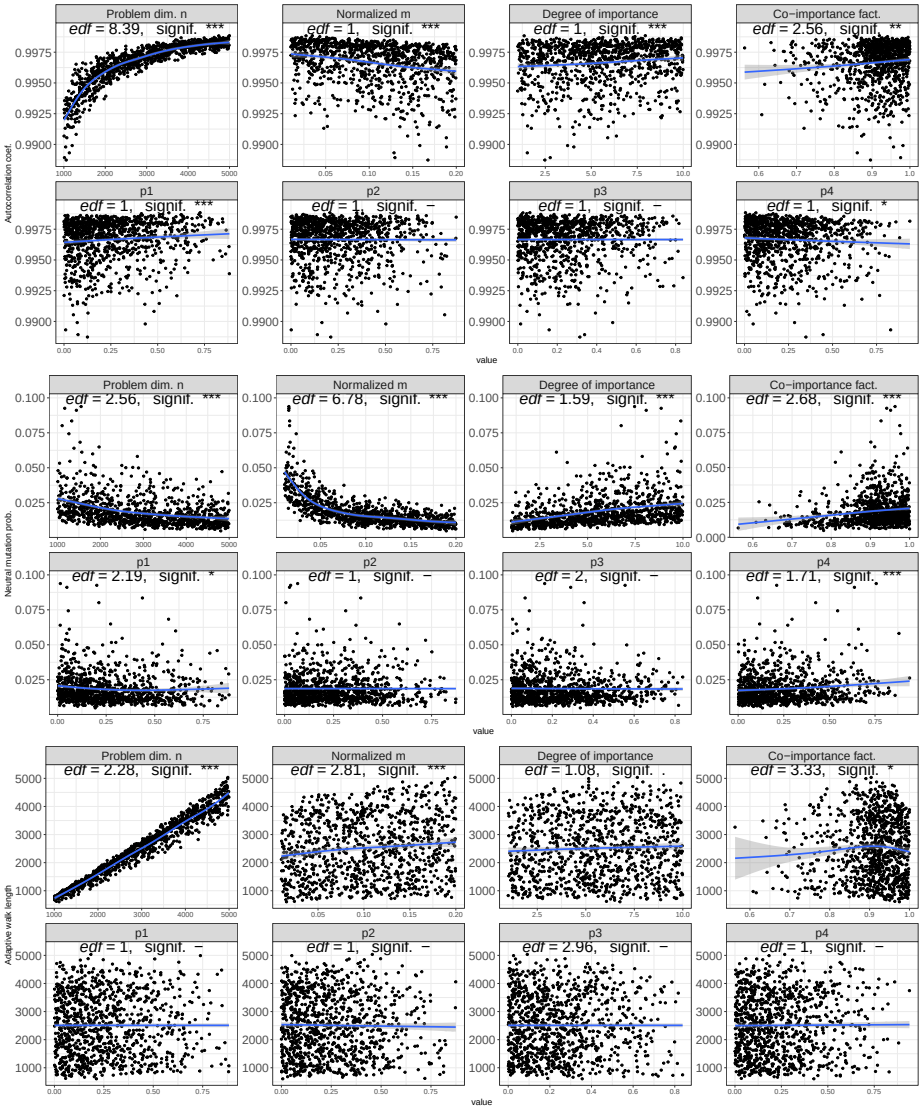

Fig. 3. Scatterlots features *vs.* benchmark parameters with GAM model regression.

highly significant impact on the autocorrelation (ruggedness) of landscapes. The factor of independence α has a significant impact on the autocorrelation. The autocorrelation increases, meaning ruggedness level decreases, with the problem dimension, and the factor α. The relation between autocorrelation, and problem dimension is a highly non-linear relationship. Ruggedness levels also decrease with the degree of importance d_0, and the presence of subproblem C_1. For these two parameters, the relationship with the autocorrelation is linear. With larger

values of m, the ruggedness level increases linearly. Higher ruggedness levels are coherent with a higher number of subproblems as it increases the complexity of the problem, and the interdependence between subproblems. n has the highest impact on the autocorrelation, which is consistent with general results on landscapes. Specific parameters to PUBO$_i$ on variable importance also impacts ruggedness levels, among those α has the highest impact.

Another facet of problem difficulty is the neutrality feature. All benchmark parameters n, m, d_0, α and p_4 (except p_1, p_2, and p_3) have a significant impact on neutrality level. While increasing both problem dimension and number of sub-functions decrease neutrality rates, higher values of the importance degree d_0, factor α and p_4 increase these rates. Parameters m and α have the highest effect on neutrality of the problem instances.

The impact of benchmark parameters on the length of adaptive walk is usually lower than for the autocorrelation and neutrality rates. The problem dimension significantly increases this length, as on most combinatorial problems. A larger number of subproblems seems to increase the multimodality of landscapes. Nonetheless, the walk length does not take into account the variation of fitness between two steps of adaptive walks, limiting further interpretations.

The autocorrelation coefficient, and probability of neutral mutation can be computed individually for each variable. This allows to study the impact of benchmark parameters on each importance class of variables. Fig. 4 shows the scatter plot with GAM model regression of the landscape features difference between important and non-important variables *versus* the benchmark parameters. Both for the autocorrelation and neutrality rates, values are negative, meaning that the contribution of important variables is lower than the one of non-important variables: the subspace of important variables is more rugged, and less neutral (flat) than the subspace of non-important variables. With the increase of n, and m, the impact of important variables on the autocorrelation significantly decreases, with average values respectively ranging from -0.012 to -0.003, and 0.0075 to 0.004. The opposite happens with the importance degree d_0, where the average difference of autocorrelation values significantly increases from 0.0025 to 0.007. Increasing the importance degree significantly increases the ruggedness contribution of important variables. The factor of independence α has a significant impact on the contribution of important variables to the autocorrelation. When this factor is increased by a small gap, important variable contribute more to the ruggedness. Interestingly, while ruggedness level is low, it seems higher around important variables, possibly indicating these landscapes are globally smooth but locally rugged, as some UBQP landscapes [21].

Except for the subproblems C_i, PUBO$_i$ parameters strongly impact the neutrality level of important variables. Increasing the degree of importance d_0 and the factor α raise the contribution of important variables on neutrality rates. On larger instances and instances with more subproblems, the contribution of important variables wanes while remaining higher than the one of non-important variables. Note that some portfolio subproblems have a higher impact on the

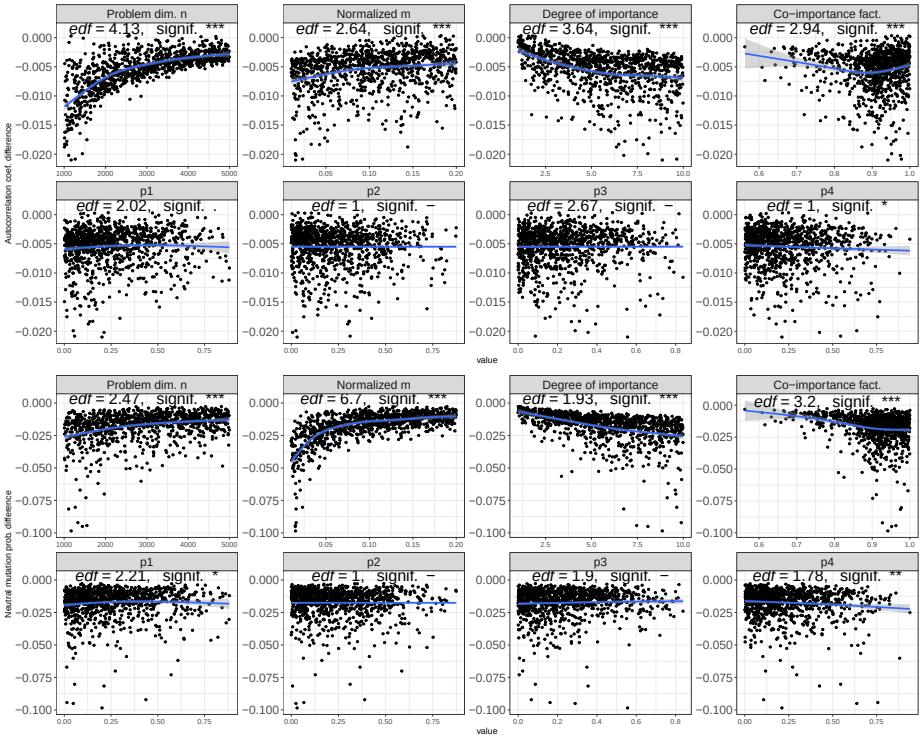

Fig. 4. Scatterlots of features difference between important and non-important variables *vs.* benchmark parameters with GAM model regression.

difference for neutrality levels than ruggedness, indeed C_1 and C_4 contain 3 out of 4 same contribution values.

5 Discussion and Future Work

The experimental analysis shows that except for parameters which tune the proportion of subproblem classes (p_i), all of PUBO$_i$ parameters have a significant impact on ruggedness, multimodality and neutrality levels of landscapes. In particular, parameters related to variable importance could have the same impact on landscape, and so on the problem difficulty, than classical parameter such as problem dimension. Moreover, the tunable importance of the benchmark leads to non isotropic landscapes where the features of landscapes are different for the subspace of important variables. To our best knowledge, this property of importance is rarely taking into account in benchmark design. The difference of landscape features between important and non-important variables shows that this property should be considered in the design of evolutionary, and local search algorithms. In particular, the design of local search operator, and neighborhood should be designed according to the variable importance either by an expert,

using machine learning technique, or the both. The PUBO$_i$ generator allows us to facilitate this approach by bringing a large set of diverse instances.

One natural perspective is to able to compare real-world combinatorial problems to PUBO$_i$ instances, in particular according to the variable importance property. It would also be relevant to study other possible benchmark parameters such as the number of importance classes, and more deeply the composition of portfolio. Although we have been able to analyze the fitness landscapes of PUBO$_i$ instances, new analysis should be conducted using for example Local Optima Network. Moreover, new fitness landscape analysis tools should be designed in order to sharply describe the anisotropy of a landscape. Of course, the goal of this benchmark is to train, test, and understand new optimization algorithms (quantum or classical), and further developments could be conducted in this research direction. The design methodology of variable importance used in PUBO$_i$ is generic, and another considered research direction is to extend the generator to other type of optimization problems (continuous, etc.).

Acknowledgements

Experiments presented in this paper were carried out using the CALCULCO computing platform, supported by SCoSI/ULCO (Service COmmun du Système d'Information de l'Université du Littoral Côte d'Opale).

References

1. Bartz-Beielstein, T., Doerr, C., Berg, D.v.d., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., et al.: Benchmarking in optimization: Best practice and open issues. arXiv preprint arXiv:2007.03488 (2020)
2. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB–a quadratic assignment problem library. Journal of Global optimization **10**(4), 391–403 (1997)
3. Carnell, R.: lhs: Latin hypercube samples, r package version 0.16 (2018)
4. Date, P., Patton, R.M., Schuman, C.D., Potok, T.E.: Efficiently embedding QUBO problems on adiabatic quantum computers. Quantum Inf. Process. **18**(4), 117 (2019)
5. Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O.M., Bäck, T.: Benchmarking discrete optimization heuristics with IOHprofiler. Applied Soft Computing **88**, 106027 (2020)
6. Galluccio, A., Loebl, M., Vondrák, J.: Optimization via enumeration: a new algorithm for the max cut problem. Mathematical Programming **90**(2), 273–290 (2001)
7. Glover, F., Hao, J.K., Kochenberger, G.: Polynomial unconstrained binary optimisation–part 1. International Journal of Metaheuristics **1**(3), 232–256 (2011)
8. Hamze, F., Jacob, D.C., Ochoa, A.J., Perera, D., Wang, W., Katzgraber, H.G.: From near to eternity: spin-glass planting, tiling puzzles, and constraint-satisfaction problems. Physical Review E **97**(4), 043303 (2018)
9. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: GECCO. pp. 1689–1696 (2010)
10. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: A platform for comparing continuous optimizers in a black-box setting. Optimization Methods and Software **36**(1), 114–144 (2021)

11. Johnson, D.S., Trick, M.A.: Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993, vol. 26. American Mathematical Soc. (1996)
12. Leprêtre, F., Fonlupt, C., Verel, S., Marion, V., Armas, R., Aguirre, H., Tanaka, K.: Fitness landscapes analysis and adaptive algorithms design for traffic lights optimization on sialac benchmark. Applied Soft Computing **85**, 105869 (2019)
13. Lü, Z., Glover, F., Hao, J.K.: A hybrid metaheuristic approach to solving the UBQP problem. European Journal of Operational Research **207**(3), 1254–1262 (2010)
14. Malan, K.M., Engelbrecht, A.P.: A survey of techniques for characterising fitness landscapes and some possible ways forward. Information Sciences **241** (Aug 2013)
15. Malan, K.M.: A survey of advances in landscape analysis for optimisation. Algorithms **14**(2), 40 (2021)
16. Perera, D., Akpabio, I., Hamze, F., Mandra, S., Rose, N., Aramon, M., Katzgraber, H.G.: Chook–a comprehensive suite for generating binary optimization problems with planted solutions. arXiv preprint arXiv:2005.14344 (2020)
17. Sagawa, M., Aguirre, H., Daolio, F., Liefooghe, A., Derbel, B., Verel, S., Tanaka, K.: Learning variable importance to guide recombination. In: 2016 IEEE SSCI. pp. 1–7 (2016)
18. Sagawa, M., Aguirre, H., Daolio, F., Liefooghe, A., Derbel, B., Verel, S., Tanaka, K.: Learning variable importance to guide recombination on many-objective optimization. In: 6th IIAI-AAI. pp. 874–879. IEEE (2017)
19. Schuster, P., Fontana, W., Stadler, P.F., Hofacker, I.L.: From sequences to shapes and back: a case study in RNA secondary structures. Proceedings of the Royal Society of London. Series B: Biological Sciences **255**(1344), 279–284 (1994)
20. Taillard, E.: Benchmarks for basic scheduling problems. european journal of operational research **64**(2), 278–285 (1993)
21. Tari, S., Basseur, M., Goëffon, A.: Sampled walk and binary fitness landscapes exploration. In: International Conference on Artificial Evolution (Evolution Artificielle). pp. 47–57. Springer (2017)
22. Tušar, T., Brockhoff, D., Hansen, N.: Mixed-integer benchmark problems for single- and bi-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 718–726 (2019)
23. Tušar, T., Brockhoff, D., Hansen, N., Auger, A.: COCO: The bi-objective black box optimization benchmarking (bbob-biobj) test suite. ArXiv e-prints (2016)
24. Verel, S., Derbel, B., Liefooghe, A., Aguirre, H., Tanaka, K.: A Surrogate Model Based on Walsh Decomposition for Pseudo-Boolean Functions. In: PPSN XV, vol. 11102, pp. 181–193. Cham (2018)
25. Walsh, J.L.: A Closed Set of Normal Orthogonal Functions. American Journal of Mathematics **45**(1), 5 (1923)
26. Weinberger, E.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological cybernetics **63**(5), 325–336 (1990)
27. Wood, S.: mgcv: GAMs in r. Generalized Additive Mixed Models Using mgcv and lme4 (2012)
28. Wright, S.: The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In: Proceedings of the Sixth International Congress of Genetics. vol. 1, pp. 356–366 (1932)
29. Zuur, A.F., Ieno, E.N., Walker, N.J., Saveliev, A.A., Smith, G.M.: Things are not always linear; additive modelling. In: Mixed effects models and extensions in ecology with R, pp. 35–69. Springer (2009)