Fourier Transform-based Surrogates for Permutation Problems

Francisco Chicano
chicano@uma.es
ITIS Software, Universidad de Málaga
Málaga, Spain

Bilel Derbel
bilel.derbel@univ-lille.fr
Inria Lille - Nord Europe
Lille, France

Sébastien Verel
verel@univ-littoral.fr
Université du Littoral Côte d'Opale
Calais, France

ABSTRACT

In the context of pseudo-Boolean optimization, surrogate functions based on the Walsh-Hadamard transform have been recently proposed with great success. It has been shown that lower-order components of the Walsh-Hadamard transform have usually a larger influence on the value of the objective function. Thus, creating a surrogate model using the lower-order components of the transform can provide a good approximation to the objective function. The Walsh-Hadamard transform in pseudo-Boolean optimization is a particularization in the binary representation of a Fourier transform over a finite group, precisely defined in the framework of group representation theory. Using this more general definition, it is possible to define a Fourier transform for the functions over permutations. We propose in this paper the use of surrogate functions based on the Fourier transforms over the permutation space. We check how similar the proposed surrogate models are to the original objective function and we also apply regression to learn a surrogate model based on the Fourier transform. The experimental setting includes two permutation problems for which the exact Fourier transform is unknown based on the problem parameters: the Asteroid Routing Problem and the Single Machine Total Weighted Tardiness.

CCS CONCEPTS

• **Computing methodologies → Randomized search**; **Discrete space search**; *Supervised learning*; • **Theory of computation → Random search heuristics**.

KEYWORDS

Surrogate models, permutation problems, group representation theory

ACM Reference Format:
Francisco Chicano, Bilel Derbel, and Sébastien Verel. 2023. Fourier Transform-based Surrogates for Permutation Problems. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590425

1 INTRODUCTION

A *surrogate function* is an easy-to-compute function that replaces a costly evaluation process. In the context of optimization, they are used as a proxy of the real objective function to optimize when it is not possible to compute it or it is too costly [2, 14, 21]. One example of application could be the optimization of some parameters of a nuclear power plant. For security reasons, it is not possible to evaluate the parameters proposed by a search method in a real nuclear power plant and, thus, simulators are used to evaluate the proposed parameters. The simulators themselves require tens of minutes to test each configuration [5]. In these cases, surrogate functions could be created based on a supervised machine learning model that is trained based on the runs of the simulator. Such a general approach has been extensively used in the scientific literature, and a number of related techniques and methodologies are being continuously developed and enhanced. In the scope of continuous optimization problems, surrogates are relatively well understood and the related literature from the statistical and machine learning field is too vast to summarize in the scope of this paper. The reader is referred to some recent surveys on the subject [2, 14, 21]. In this paper, we target combinatorial domains, and more specifically permutation-like problems.

In this context, although a lot of advances have been made, there is still a huge gap to fill with the existing work for continuous domains. In fact, combinatorial search spaces are fundamentally different, as it is even not clear how to model and to infer the underlying discrete landscapes. In particular, designing discrete surrogates has been identified in [21] as among of the five major challenges that require further in-depth investigations from the community. One can however find a number of investigations on the subject, ranging from naive approaches ignoring the discrete structure of the search space, to more domain-specific modeling techniques. In particular, algorithms like CEGO [26] use surrogate functions to speedup the search and/or use a low number of evaluations of the real objective function. The models used by these algorithms require the existence of a distance defined over solutions in the search space. When the search space is the set of permutations of a given size n, denoted with S_n, some examples of distances used in the literature are swap and interchange among others (see [25] for a list of distances).

A different approach to build a surrogate model is based on the Fourier transform over finite groups. For pseudo-Boolean optimization problems, the so-called Walsh basis [24] were shown to constitute a well-suited theoretical tool to derive custom surrogates for arbitrary functions [6, 17, 22, 23]. Discrete Walsh functions form an orthogonal set of functions inferring a Fourier transform to represent any blackbox pseudo-Boolean function in an additive linear form, which can then be approximated by using some standard linear regression techniques. This general working principle can interestingly be leveraged for other combinatorial domains, given we can represent the considered optimization problem in some form that can be decomposed using a dedicated Fourier transform.

One interest in using a Fourier transform instead of a distance-based model, is that distances do not reflect precisely the landscape structure of the fitness function. In fact, having all the components of the Fourier transform is equivalent to having the original function. Previous studies have shown that the lower order components of the Fourier transform have usually a larger influence in the value of the objective function. Thus, creating a surrogate model for the lower order components can provide a good approximation of the objective function.

In this paper, we propose the use of surrogate models based on the Fourier transform of the permutation space. Some features of the Fourier transform for some permutation problems are known. This is the case of Quadratic Assignment Problem (QAP) [15], Traveling Salesperson Problem [19] and the Linear Ordering Problem [7]. In the latter problem, it has been proven that the first order component of the Fourier transform can be solved in polynomial time [7]. However, given an expensive permutation problem, it is difficult (or impossible) to know features of its Fourier transform. Our proposal is to learn the coefficients of the Fourier transform using regression and based on samples of the problem. A similar idea was proposed in the past by Irurozki et al. [13] to learn probability distributions.

The rest of the paper is organized as follows. Section 2 presents the background required to understand our proposal. Section 3 presents the surrogate models based on the Fourier transform. In Section 4, we describe the research questions we want to answer and the experiments performed to do it. Finally, Section 5 concludes the paper.

2 BACKGROUND

Our proposal is based on the Fourier transform in the permutation space [12]. Thus, in this section we do a short introduction to this Fourier transform and the more general concept of Fourier transform over finite groups.

2.1 Group representation

A *group* is a pair $(G, \cdot)$, where G is a set of (group) elements and $\cdot : G \times G \to G$ is a binary operator defined over the elements of the group with the following properties:

- Associative: $g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3$
- Neutral element: there is an element $e \in G$ such that $e \cdot g = g \cdot e = g$ for all $g \in G$.
- Inverse: for each $g \in G$ there is another element g^{-1} such that $g \cdot g^{-1} = g^{-1} \cdot g = e$.

The definition of group is the most general one and quite abstract. Fortunately, there is a mathematical tool that allows us to represent group elements in a concrete way using matrices: *group representation theory*. In the following, we will use G to denote a group, omitting the binary operation for the sake of clarity.

A *representation* of a group G is a mapping $\rho : G \to GL(V)$ between the elements of the group G and the automorphisms of a vector space V such that $\rho(g_1 \cdot g_2) = \rho(g_1)\rho(g_2)$. An *automorphism* is an invertible linear map from a vector space V to itself. Without loss of generality, we will assume in the following $V = \mathbb{C}^n$, where the vectors are n-tuples of complex numbers and the automorphisms are non-singular squared complex matrices of size $n \times n$, that is, matrices with a nonzero determinant. In short, a group representation translates the group elements into matrices and the group operation into matrix multiplication. This way we can use the mathematical tools of linear algebra to solve statements related to the group.

Not all group representations are equally important. We say that two representations, ρ_1 and ρ_2 are *equivalent* if there exists a matrix $P \in \mathbb{C}^{n \times n}$ such that $\rho_1(g) = P\rho_2(g)P^{-1}$ for all $g \in G$. Two equivalent representations provide the same information about the group and, thus, we will work only with a set of *inequivalent* representations. We say that one representation ρ is *reducible* if we can write $\rho(g)$ as a block-diagonal matrix of the form:

$$\rho(g) = \left(\begin{array}{c|c} \rho_1(g) & 0 \\ \hline 0 & \rho_2(g) \end{array}\right), \tag{1}$$

for all $g \in G$. Reducible representations do not provide more information than their component representations and, thus, we prefer *irreducible* representations. When the group is finite there is also a finite set of *inequivalent irreducible* representations, called *irreps*, and the cardinality of this set is exactly the number of conjugacy classes of the group [9].

2.2 Fourier transform over finite groups

Let $f : G \to \mathbb{C}$ be a complex-valued function defined on group G and ρ a representation of G. The *Fourier transform* of f at ρ is defined as:

$$\hat{f}(\rho) = \sum_{g \in G} f(g)\rho(g). \tag{2}$$

Observe that $\hat{f}(\rho)$ is, in general, a matrix because $\rho(g)$ is a matrix. The Fourier transform of a function f at a particular representation does not provide all the information to reconstruct f[1], but the Fourier transform at a set of irreps does. The inverse Fourier transform is defined as:

$$f(g) = \frac{1}{|G|} \sum_{\rho \in \text{irreps}} d_\rho \text{Tr}\left(\hat{f}(\rho)\rho(g)^{-1}\right), \tag{3}$$

where d_ρ is the dimension of the vector space associated to representation ρ, Tr is the trace of a matrix, and $\rho(g)^{-1}$ denotes the inverse of $\rho(g)$.

2.3 Fourier transform in permutation space

Permutations form a group with the composition operation, called the *symmetric group* and denoted with S_n, where n is the number of elements in the permutation. Given a permutation $\sigma \in S_n$ we denote with $\sigma(i)$ the i-th element of σ. If $\sigma, \pi \in S_n$ we define the composition operation as:

$$(\sigma \circ \pi)(i) = \sigma(\pi(i)). \tag{4}$$

For example, let $\sigma = (1, 4, 2, 3)$ and $\pi = (3, 1, 4, 2)$ be two permutations of S_4. Then, $\sigma \circ \pi = (2, 1, 3, 4)$.

The application of representation theory to the symmetric group has some special simplifications. In particular, the number of irreps of the symmetric group S_n is the number of *partitions* of n, that is, the number of ways in which n can be written as a sum of positive integers. Furthermore, there is a relationship between these partitions and the irreps and we can label each irrep with a partition.

[1]Unless the group G has only one element.

We will use the notation $\lambda \vdash n$ to express that λ is a partition of number n. Particular partitions will be expressed as a tuple of positive integers in nonincreasing order $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ with $\lambda_i \geq \lambda_{i+1}$. For example, (4), $(3,1)$, $(2,2)$, $(2,1,1)$ and $(1,1,1,1)$ are the five partitions of number 4. We will denote with ρ_λ the irrep associated to partition λ. Thus, the five irreps for S_4 will be denoted with $\rho_{(4)}$, $\rho_{(3,1)}$, $\rho_{(2,2)}$, $\rho_{(2,1,1)}$, $\rho_{(1,1,1,1)}$.

There is a visual way of representing partitions that will be useful for some computations: Young diagrams. A Young diagram has as many rows as elements in the partitions and row i has as many columns as the value λ_i. See Figure 1 for an example. We will denote with C_λ the set of cells of the Young diagram for λ.

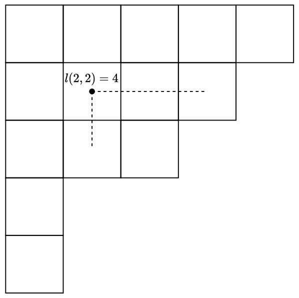

Figure 1: Young diagram for partition $\lambda = (5,4,3,1,1)$ in the symmetric group S_{14}.

With the introduced notation, we can particularize the formulas of the direct and inverse Fourier transforms to the symmetric group as follows:

$$\hat{f}(\rho_\lambda) = \sum_{\sigma \in S_n} f(\sigma) \rho_\lambda(\sigma) \tag{5}$$

$$f(\sigma) = \frac{1}{n!} \sum_{\lambda \vdash n} d_\lambda \mathrm{Tr}\left(\hat{f}(\rho_\lambda) \rho_\lambda(\sigma)^{-1}\right) \tag{6}$$

The dimension of an irrep ρ_λ is given by the hook rule:

$$d_\lambda = \frac{n!}{\prod_{(r,c) \in C_\lambda} l(r,c)}, \tag{7}$$

where $l(r,c)$, called the *hook length*, is the sum of the elements found at the right and below the cell (r,c) in the Young diagram, including cell (r,c) (see Figure 1).

The order of Walsh functions in the pseudo-Boolean domain relates to the ruggedness of a fitness landscape [20]. In the case of the symmetric group we can also establish a relationship between the partition of an irrep and its effect in the fitness landscape. We say that an irrep is of order k if $n-k$ is the highest positive integer in the partition. For example, $\rho_{(n)}$ is the 0-th order irrep of the symmetric group S_n. The 0-th order irrep is always a scalar value and its Fourier transform represents the sum of the values of f for all the permutations. The first order irrep is $\rho_{(n-1,1)}$ and is related to first order marginals when the function f represents a probability distribution. If $n \geq 4$ we find two second order irreps, $\rho_{(n-2,2)}$ and $\rho_{(n-2,1,1)}$, one related to unsorted second order marginals and the second one related to sorted second order marginals. The interested reader can find more details on the interpretation of the Fourier transforms at higher order irreps in the work of Huang et al. [12]. The number of irreps of order k is the number of partitions of k for large enough n. For example, there are five order-4 irreps and seven order-5 irreps[2].

There are some permutation optimization problems that have been analyzed from the point of view of the Fourier transform. In particular, the Linear Assignment Problem (LAP) has only nonzero Fourier transforms at representations of order zero and one [7]. The Quadratic Assignment Problem (QAP) has nonzero Fourier transform at irreps of order two or less [15]. The Traveling Salesperson Problem (TSP) is a particular case of QAP that has a zero Fourier transform at $\rho_{(n-1,1)}$ and nonzero in at least one of $\rho_{(n-2,2)}$ or $\rho_{(n-2,1,1)}$.

2.4 Young orthogonal representation

We need to define a family of irreps before we apply the direct and inverse Fourier transform. There is an infinite number of possibilities here but there are some for which the math can be simplified. One of these is the *Young Orthogonal Representation* (YOR). First, YOR produces always real-valued matrices, which simplifies things when we are working with real-valued permutation problems, as it happens in the context of combinatorial optimization. The second interesting property of YOR is that it always produce orthogonal matrices. This means that computing the inverse of matrix only requires transposing the elements of the matrix, that is $\rho(g)^{-1} = \rho(g)^T$. This latter property allows us to simplify Eq. (6) for the inverse of the Fourier transform to:

$$f(\sigma) = \frac{1}{n!} \sum_{\lambda \vdash n} d_\lambda \mathrm{Tr}\left(\hat{f}(\rho_\lambda) \rho_\lambda(\sigma)^T\right), \tag{8}$$

where $\rho_\lambda(\sigma)^{-1}$ has been replaced by $\rho_\lambda(\sigma)^T$.

3 FOURIER-BASED SURROGATE MODELS

In this section we present the main idea in this paper: use the Fourier transform on the permutation space to define a surrogate model for a permutation problem. In order to do that, we have to develop Eq. (8) in a way that is useful to present the idea. For this, we will write the trace explicitly. Let's do that first for two arbitrary squared matrices A and B of size n:

$$\mathrm{Tr}\,(AB) = \sum_{i=1}^{n} (AB)_{ii} = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} B_{j,i}. \tag{9}$$

We can apply this expression to Eq. (8) taking into account that $(B^T)_{ji} = B_{ij}$. The result is:

$$f(\sigma) = \frac{1}{n!} \sum_{\lambda \vdash n} d_\lambda \sum_{i,j=1}^{d_\lambda} \hat{f}(\rho_\lambda)_{i,j} \rho_{\lambda,i,j}(\sigma) \tag{10}$$

where the permutation functions $\rho_{\lambda,i,j}$ are orthogonal among each other (this is a result of representation theory) [9]. All the values in Eq. (10) are scalar values (there are no matrices or vectors). The functions $\rho_{\lambda,i,j}$ are given by the Young orthogonal representation and they do not depend on the particular permutation problem. The only terms that depend on the problem are the constants $\hat{f}(\rho_\lambda)_{i,j}$.

[2]See the number series at https://oeis.org/A000041

We propose to empirically find the values $\hat{f}(\rho_\lambda)_{i,j}$ using linear regression for a given permutation problem f with a set of irreps associated to partitions $\lambda \in \Lambda$, and use (10) as a surrogate model for f, that is:

$$\boxed{f(\sigma) \approx \frac{1}{n!} \sum_{\lambda \in \Lambda} d_\lambda \sum_{i,j=1}^{d_\lambda} \hat{f}(\rho_\lambda)_{i,j} \rho_{\lambda,i,j}(\sigma)} \tag{11}$$

Let σ_l for $1 \leq l \leq m$ be a set of permutation samples for which we have the values of f. We need to solve the following minimization problem:

$$\min \sum_{l=1}^{m} \left| f(\sigma_l) - \frac{1}{n!} \sum_{\lambda \in \Lambda} d_\lambda \sum_{i,j=1}^{d_\lambda} \hat{f}(\rho_\lambda)_{i,j} \rho_{\lambda,i,j}(\sigma_l) \right|^2 \tag{12}$$

where the unknowns are the Fourier coefficients $\hat{f}(\rho_\lambda)_{i,j}$.

One last issue we should consider is the number of unknowns in the surrogate model. This is the number of degrees of freedom that the model will have to adapt to the real function. The number depends on the family Λ of irreps that will be used in Eq. (11). The number of unknowns $\hat{f}(\rho_\lambda)_{i,j}$ associated to an irrep λ is d_λ^2. We can compute d_λ using the hook rule (7). The number d_λ increases with the order, k, of the irrep up to $k = n/2$, and decreases after that. In particular, $d_{(n)} = 1$, $d_{(n-1,1)} = n - 1$, $d_{(n-2,2)} = (n-1)(n-3)/2$ and $d_{(n-2,1,1)} = (n-1)(n-2)/2$. In general, $d_\lambda \in O(n^k)$ if $k \leq n/2$ as the next proposition proves.

Proposition 3.1. *The dimension d_λ of the vector space associated to an irrep $\lambda = (n-k, \ldots)$ of the symmetric group S_n with $k \leq n/2$ a constant is $O(n^k)$.*

Proof. We can prove the proposition by applying the hook rule (7) to compute d_λ. Let's denote with $l(r,c) \geq 1$ the hook length of cell in row r and column c of the Young diagram for λ. First, observe that since $\lambda_1 = n - k$ we have $\sum_{i \geq 2} \lambda_i = k$ and, as a consequence, all the hook lengths for the cells in rows greater than 1 will be upper bounded by k, which is a constant independent of n if $k < n/2$. Thus, we can focus on the hook lengths of the cells in the first row, which are the only ones depending on n. We can easily find that $l(1,c) \geq n - k - c + 1$, because $n - k - c + 1$ is the number of elements at the right of cell $(1,c)$ including the cell. That is, $l(1,1) \geq n-k$, $l(1,2) \geq n-k-1$, etc. Now, applying hook rule and using the lower bounds for $l(r,c)$ we have:

$$d_\lambda = \frac{n!}{\prod_{(r,c) \in C_\lambda} l(r,c)} \leq \frac{n!}{\prod_{c=1}^{n-k} l(1,c)} = \frac{n!}{\prod_{c=1}^{n-k} (n-k-c+1)}$$
$$= \frac{n!}{(n-k)!} \leq n^k$$

□

Proposition 3.1 implies that the number of degrees of freedom (unknowns) provided by an irrep of order k is $O(n^{2k})$, since $d_\lambda \in O(n^k)$ and there are d_λ^2 unknown coefficients $\hat{f}(\rho_\lambda)_{i,j}$. The sum of d_λ^2 for all $\lambda \vdash n$ is $n!$, the size of the search space, which is also the total number of unknowns in Eq. (10). This motivates the use of only some irreps for the surrogate model.

Previous experience with Fourier-based surrogate models in the binary space [23] suggests that in many interesting optimization problems the lower order terms of the Fourier decomposition have a higher impact in the function evaluation. Thus, it seems a natural choice to add first the lower order terms of the Fourier transforms to the surrogate model and keep adding higher order terms to increase the quality of the approximation.

If we use the irreps up to order k in the Fourier expansions of f, we have $O(n^{2k})$ unknowns in Eq. (11). The number of unknowns in the surrogate model, $O(n^{2k})$, is also the minimum number of samples we need from the problem to solve the regression problem expressed in Eq. (12) without ambiguity. Kondor [15] and Elorza et al. [8] proved that in some permutation problems, like QAP, the nonzero matrices $\hat{f}(\rho_\lambda)$ have rank 1 or 2. This fact could be used to reduce the number of considered unknowns.

4 EXPERIMENTAL SECTION

In this experimental section we want to check how good is our proposed surrogate model for permutation problems for which we do not know the Fourier transform. In particular, we want to answer the following two research questions:

- **RQ1**: What is the importance of the Fourier transform at different orders in the value of the permutation function? In order to answer this question we compute the exact Fourier transform of a set of instances of two problems at all the irreps and build surrogate models based on the exact Fourier transform with higher orders omitted.
- **RQ2**: How much different is the learned surrogate model compared to the original function? In practice, we do not know the exact Fourier decomposition of an optimization problem (except for some particular cases), and we have to learn it from samples of the function. In order to answer this question, we use small instances of the same two problems used in RQ1 and we will learn a surrogate model like in Eq. (11). We will compare the model with the original function to see the differences in different aspects (detailed below).

This experimental section is organized as follows. We introduce the permutation problems and instances used in Subsection 4.1. Then, we describe the metrics used to compare the surrogate models with the original function in Subsection 4.2, followed by a description of the experimental setting (Subsection 4.3). The main results are split in three separate subsections. First, Subsection 4.4 analyzes the Fourier transforms of the instances of the problems. Then, research questions **RQ1** and **RQ2** are addressed in Subsections 4.5 and 4.6, respectively.

4.1 Problems and instances

For the purpose of our analysis, we use two problems for which we do not know the Fourier transform: the Single Machine Total Weighted Tardiness Problem (SMTWTP) and the Asteroid Routing Problem (ARP). In this section we describe both problems and the instances used for the experiments. The instances, together with the code for the experiments are available in the replication package accompanying this paper[3].

[3]Available at https://doi.org/10.5281/zenodo.7850763

4.1.1 *Single machine total weighted tardiness.*

The Single Machine Total Weighted Tardiness Problem (SMTWTP) is a well-known strongly NP-hard problem [16]. It is defined by a set of n jobs. Each job has to be processed without interruption on a single machine that can only process one single job at a time. Each job has a processing time p_j, a due date d_j and an associated weight w_j (reflecting the importance of the job). The tardiness of a job j is defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of job j in the current sequence of jobs. The goal is then to find a job sequence σ minimizing the sum of weighted tardiness defined by $\sum_{i=1}^{n} w_{\sigma(i)} \cdot T_{\sigma(i)}$. A number of works have considered it as a benchmark to study the properties of a number of optimization methods, e.g., [1, 4, 10, 11] to cite a few. As such, we follow the common procedure to generate SMTWTP benchmark instances [3]. More precisely, the processing times p_j and the weights w_j are drawn uniformly at random respectively in the integer intervals $[1, 100]$ and $[1, 10]$. The due dates are drawn uniformly at random within the interval $[P \cdot (1 - \text{TF} - \text{RDD}/2), P \cdot (1 - \text{TF} + \text{RDD}/2)]$, where $P = \sum_{j=1}^{n} p_j$ and RDD and TF are two parameters controlling respectively the relative $\underline{\text{r}}$ange of $\underline{\text{d}}$ue $\underline{\text{d}}$ates and the average $\underline{\text{t}}$ardiness $\underline{\text{f}}$actor TF. Overall, we consider 750 instances.

4.1.2 *Asteroid routing problem.*

The Asteroid Routing Problem was defined by López-Ibáñez et al. [18] as a benchmark for black-box optimization in the context of permutation problems. The problem consists in finding a route for a spacecraft launched from Earth to visit a given set of n asteroids $A = \{a_1, a_2, \ldots, a_n\}$ with the goal of minimizing the sum of velocity changes required by the route (related to fuel consumption) and the total time required to visit all of them. Although there are two objectives in the problem, the authors combine both of them in one single objective using a weighted sum.

A solution to the problem is a pair $(\pi, \vec{t})$, where $\pi \in S_n$ is a permutation representing the order in which the asteroids are visited and $\vec{t} \in \mathbb{R}^{2n}_{\geq 0}$ is a vector of $2n$ real numbers representing parking and transit times to reach each asteroid.

The problem is solved as a bilevel optimization problem, where the visit order of the asteroids is in the higher level and the transit and parking times is in the lower level. For the lower level, a Sequential Least Squares Programming (SLSQP) is used in [18] to determine $\vec{t}$ when the visit order of asteroids π is known (given by the higher level). Thus, the higher level problem is a permutation problem where the optimal visit order of the asteroids π must be determined. One evaluation of a permutation (visit order) requires solving an optimization problem with SLSQP where for each evaluation of the complete solution, that is $(\pi, \vec{t})$, it is required to solve several Lambert problems and move a set of n asteroids plus the Earth in a simulated environment. This makes the objective function computationally costly and a good candidate for replacing it with surrogate models.

López-Ibáñez et al. [18] prepared an instance generator for the problem that randomly selects n asteroids from the 83 453 asteroids provided by the GTOC11 competition to create an instance of the ARP. We used this generator to generate 60 instances of sizes $n = 5$ to $n = 10$ (ten instances for each value of n), small enough to compute the metrics we use in the study.

4.2 Metrics used for checking similarity

In order to answer research questions **RQ1** and **RQ2** we need to compare the differences between the original function and the built surrogate model. We use two metrics for this.

The first one is the *Normalized Mean Absolute Error* (NMAE) for the original function and the surrogate model in all the solutions of the search space, which is defined as:

$$\text{NMAE} = \frac{\frac{1}{n!} \sum_{\sigma \in S_n} |f(\sigma) - s(\sigma)|}{f_{\max} - f_{\min}}, \tag{13}$$

where f the original function, s is the surrogate model, $f_{\max} = \max_{\sigma \in S_n} f(\sigma)$ and $f_{\min} = \min_{\sigma \in S_n} f(\sigma)$.

The second metric is the *Normalized number of Preserved Global Optima* (NPGO), which is the number of global optima of f that are also global optima in s, normalized by the total number of global optima of f that is,

$$\text{NPGO} = \frac{|\text{GO}(f) \cap \text{GO}(s)|}{|\text{GO}(f)|}. \tag{14}$$

4.3 Experimental setting

For the experiments we used 750 instances of SMTWTP and 60 instances of ARP. For the SMTWP, we follow the specialized literature as described previously in Section 4.1.1. The values for n (size of the permutation) vary between $n = 5$ and $n = 10$, and the two parameters RDD and TF are set in the range: 0.2, 0.4, 0.6, 0.8 and 1.0. For each combination of n, RDD and TF five instances with different random seeds were generated. This is a total of 125 instances for each value of n.

In the case of ARP, we used the generator provided in [18]. The values of n vary from $n = 5$ to $n = 10$. For each value of n we generated 10 random instances with ten different seeds. The seeds used are 7, 11, 13, 17, 19, 23, 29, 31, 42 and 73.

We used the Snob2[4] library for python to compute the Fourier transforms and the scikit-learn package[5] for learning the surrogate models. The experiments were run in the Picasso supercomputing facility of the University of Malaga with 126 SD530 servers with Intel Xeon Gold 6230R (26 cores each) at 2.10GHz, 200 GB of RAM and an InfiniBand HDR100 network.

4.4 Fourier coefficients

Before answering the research questions we wonder how does the Fourier transform of the SMTWTP and ARP instances looks like. In particular, we would like to know how large the matrix coefficients in the irreps are and how sparse the matrices are. Figure 2 shows the average absolute values of the matrix coefficients of the Fourier transforms at different irreps (left) and the proportion of nonzero matrix coefficients (right). Only the results for instances with $n = 5$ (top), 7 (center), and 9 (bottom) are reported. The results aggregate matrix coefficients by irrep and instances (with the same n).

We can observe a clear difference between the instances of both problems. ARP has a higher proportion of nonzero coefficients and their absolute values are also higher. This is an indication that the objective function of ARP is more noisy, something that was already mentioned in the original paper where the problem

[4]Available at https://github.com/risi-kondor/Snob2
[5]Available at https://scikit-learn.org/stable/

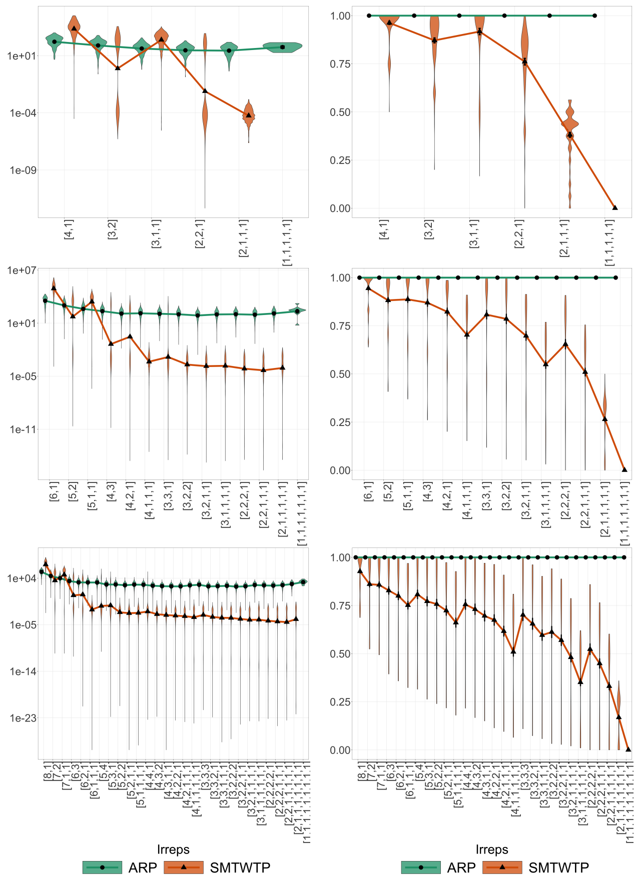

Figure 2: Left: average absolute value of the matrix coefficient for different irreps (y-axis is in log-scale). Right: proportion of nonzero coefficients. The x-axis shows the irreps in increasing order.

was proposed [18]. In contrast, the proportion of nonzero coefficients of SMTWTP decreases as we increase the order of the irreps. The absolute values for these coefficients also decrease with the higher order irreps. This suggests that SMTWTP will probably be well-approximated by low-order irreps. We will check this in the following subsections.

4.5 Answering RQ1: truncated surrogate model

In this section we conduct some experiments to answer **RQ1**: what is the importance of the Fourier transform at different orders in the value of the permutation function? With the help of the Fourier transforms at the different irreps, computed in Subsection 4.4, we build a new function in which some of the coefficients are set to zero: the coefficients of irreps of higher order. We call these functions *truncated surrogate model* because they have the same coefficients as the original objective function for the irreps of the low orders and zero for the irreps of higher orders. That is, we "truncate" the Fourier transform at higher orders.

Technically, we start with the original Fourier transform at all irreps and, step by step, we set to zero the coefficients of irreps of order p in decreasing order, starting with $p = n-1$ and ending with $p = 1$. At each step, we compute the inverse Fourier transform, and this is used as the truncated surrogate model that we evaluate.

Figure 3 shows the NMAE of the truncated surrogate model at different orders and for $n = 5$ to 9 for both, SMTWTP and ARP. We can observe in this figure a clearly different result for SMTWTP and ARP. While the truncated model of SMTWTP jumps to very low values of NMAE at some order between 3 and 5 for all n, ARP requires high order irreps to have a low value for NMAE. This means that SMTWTP can be very well approximated with irreps of low order. In many cases, order 2 is enough to get an NMAE of 10^{-3}. In contrast, ARP needs high order irreps to have a good approximation of the objective function. This supports, once again, the idea that ARP is a more noisy problem, hard to solve.

Since our aim is to use the surrogate models as a proxy of the original optimization function, we wonder if the global optima are preserved when we truncate the high order irreps. Figure 4 shows NPGO for SMTWTP and ARP. We observe that adding irreps of order 2 or 3 is usually enough to recover some global optima. Thus, the truncated surrogate model could be used to find solutions to the original optimization problems, with a faster evaluation function.

On a technical side, we observe that adding enough irreps we can find 100% of the global optima in the case of ARP, as it should be, since we recover the original objective function at the end. This does not happen with SMTWTP, where only between 40% and 75% of the global optima are the same in the truncated and original objective function when all irreps are added. The explanation of this surprising phenomenon has to do with the floating point errors in the process of computing the direct and inverse Fourier transform. Unfortunately, these errors are difficult to avoid using the Young Orthogonal Representation, where some matrix elements are irrational numbers and we cannot use any exact fractional implementation for the mathematical operations.

In summary, we conclude that low order irreps are enough in some problems, like SMTWTP, to represent the original objective function. Other problems, like ARP require also high order irreps to reach a low value for NMAE.

4.6 Answering RQ2: learned surrogate model

In this section we explore the performance of the learned surrogate model. We use Lasso as technique to solve the linear regression problem of Eq. (12) because it was more stable than others in some preliminary experiments. In particular, we use the Lasso implementation of scikit-learn with regularization parameter $\alpha = 0.000001$ and the default values for the other parameters.

For each instance we learned surrogate models with different orders. For $n = 5$ to 7 we used surrogate models with irreps having maximum order $p = 0$ to 4. For $n = 8$ to 10 the models had irreps with maximum order $p = 0$ to 2 (for higher orders Lasso had problems to do the regression due to memory limits). The surrogate models were trained with a varying number of random samples (permutations and their associated fitness value) to check how the models converge as the number of samples increases. The number of samples used varies for different values for n. In all the cases, the maximum number of samples we use is enough to learn all the matrix coefficients associated to all the irreps in the model. In

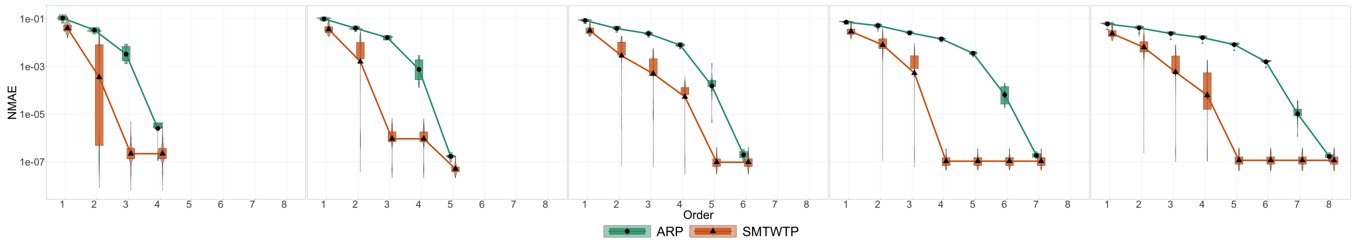

Figure 3: Boxplots of the normalized MAE for the truncated surrogate model. The Y axes is in log scale. The plots correspond to sizes $n = 5, 6, 7, 8, 9$, from left to right.

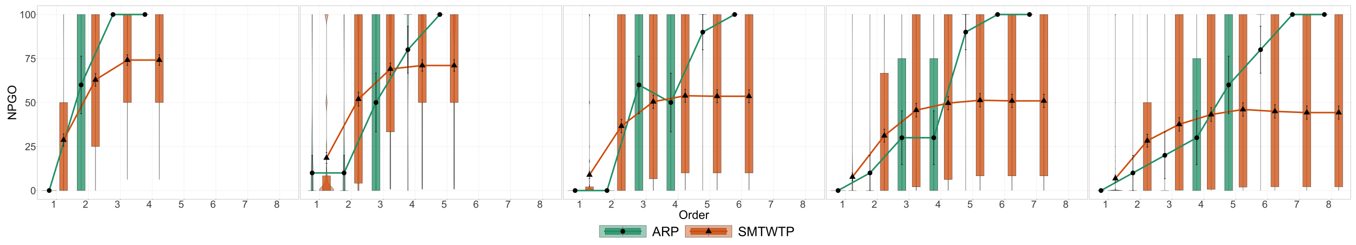

Figure 4: Boxplots of the normalized PGO for the truncated surrogate model. The Y axes represents percentage. The plots correspond to sizes $n = 5, 6, 7, 8, 9$ from left to right

particular, the ranges of number of samples used for training are shown in Table 1. For each instance, maximum order and number of samples, we repeated the learning process ten times using ten different random seeds, in order to sample the solutions in the search space in a different order and, thus, reduce any bias due to the samples used. In each run we build the surrogate model by applying the inverse Fourier transform and computed all the metrics described in Subsection 4.2.

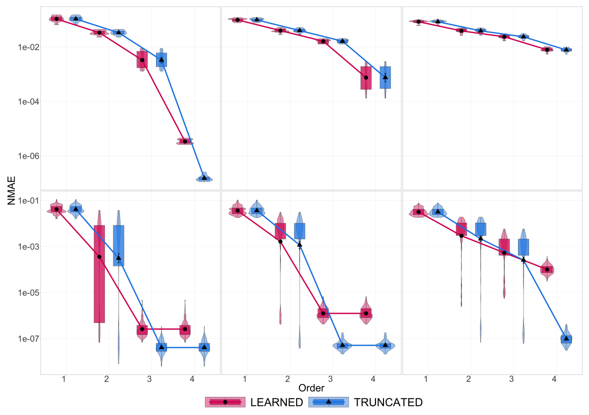

Figure 5: NMAE values for the learned and truncated surrogate models using different maximum orders of the irreps for ARP (top) and SMTWTP (bottom). In different columns are the different problem sizes $n = 5, 6, 7$ (from left to right).

Figure 5 shows the boxplots of the NMAE obtained by the learned surrogate model for ARP (top row) and SMTWTP (bottom row) for problem sizes $n = 5, 6, 7$. We add the truncated model for an easy

Table 1: Parameters of the learned surrogate models for different problem size: maximum order of the irreps considered in the regression and number of training samples. We also show the number of total regressions performed per instance.

	Maximum order			# of training samples			Number of
n	Min.	Max.	Step	Min.	Max.	Step	regressions
5	0	4	1	1	120	1	6000
6	0	4	1	6	720	6	6000
7	0	4	1	50	5000	50	5000
8	0	2	1	18	1800	18	3000
9	0	2	1	32	3200	32	3000
10	0	2	1	52	3640	52	2100

comparison. The X axis displays the maximum order used in each model. For example, maximum order 3 means that all the irreps up to order 3 are included in the surrogate model. We aggregate the results for all the instances of each problem (10 for each n in the case of ARP and 125 in the case of SMTWTP). The models shown are the ones trained with the largest value for the number of samples (see column 6 in Table 1).

We clearly see how the learned model produces an NMAE which is similar to the truncated model for orders 1 and 2. We should highlight here that the learned coefficients do not have to be similar to the correct ones (of the truncated model). In spite of that, both models get approximately the same NMAE. There is a larger variation in the NMAE of the learned model for SMTWTP, probably due to the different behaviour in the different kind of instances. In SMTWTP, we observe that using irreps up to order 2 for all n is enough to obtain an NMAE of 10^{-3}. In the case of ARP, however, we need higher order irreps to reduce the NMAE when n increases.

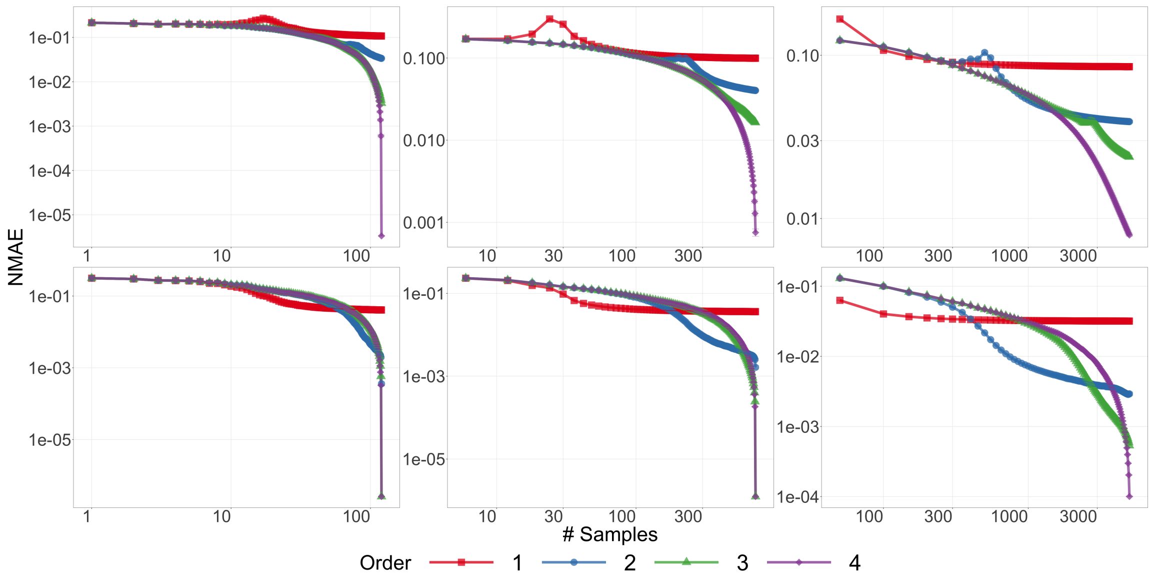

Figure 6: NMAE convergence as a function of sample size for ARP (top) and SMTWTP (bottom). The different lines represent learned surrogate models with irreps up to different order. The sizes of the problems are $n = 5, 6, 7$ (from left to right)

Finally, we analyzed the convergence behaviour of the learned model as a function of the number of samples used for the training. The results for ARP and SMTWTP and problem sizes $n = 5, 6, 7$ are in Figure 6. When the number of samples approaches the number of unknowns, the NMAE stabilizes. We observe some "peaks" in NMAE around this value for ARP. For example, the number of samples required to learn a surrogate model with irreps $\rho_{(n)}$ and $\rho_{(n-1,1)}$ (order 1) is $(n-1)^2+1$. This expression is 17 for $n = 5$ and 26 for $n = 6$, and we observe the peaks around these values for the curve corresponding to order 1 in ARP. We defer to future work an explanation for this phenomenon. More importantly, we can see that surrogates of higher orders do not systematically provide better NMAE, independently of the sample size. In fact, using small sample size with the SMTWTP problem, low order learned surrogates have better NMAE. Only when the sample size increases higher order are found to be more accurate. This can be attributed to the difficulty in learning complex models, having many parameters, when the available training data is restricted. This is important in practice, since it suggests that low order surrogates might be interesting to consider and more specific learning techniques should be considered to improve model accuracy.

In summary, we conclude that there are problems, like SMTWTP, for which a learned surrogate model with low orders can provide a good approximation of the objective function. In other problems, like ARP, higher orders are needed as n increases. It is not practical to increase the order p with n, because the coefficients to learn increase exponentially with p (if $p << n$).

5 CONCLUSIONS

We have presented in this paper a new approach to build surrogate models for permutation problems based on the Fourier transform in the space of permutations. We applied our proposal to two permutation problems: ARP and SMTWTP. We first analyzed the importance of the terms of different orders in the objective function and, then, learned a surrogate model using random samples of the problems. The results show that SMTWTP can be very well approximated using surrogate models including up to order 2 coefficients, while ARP needs higher order terms to be approximated.

This work represents the first stone in the use of Fourier-based surrogate models for permutation problems. Future work can focus on adding other metrics for the comparison between the original an surrogate function, like the difference in the ranking of the solutions in the search space. It should also be interesting to analyze how this approach can be used in practice to design algorithms able to optimize computationally costly functions with the help of Fourier-based surrogate models. It is important to limit the order of the irreps used in the model, because the number of coefficients to learn depends exponentially on the order. But there could be other ways to reduce the number of unknowns, like learning a few coefficients of each order setting the rest to zero to be able to reach higher order irreps without increasing the cost of the regression. Another idea would be to map the permutation space into a higher dimension permutation space, and use the low orders of that permutation space to learn the original function (as support vector machines do with real vectors).

ACKNOWLEDGMENTS

This work is partially funded by *Universidad de Málaga*, *Ministerio de Ciencia, Innovación y Universidades del Gobierno de España* under grants PID 2020-116727RB-I00 and PRX21/00669, and by EU Horizon 2020 research and innovative program (grant 952215, TAILOR ICT-48 network). Thanks to the Supercomputing and Bioinnovation Center (SCBI) of *Universidad de Málaga* for their provision of computational resources and support. And special thanks to José A. Lozano, for an inspiring keynote that was the seed of this work.

REFERENCES

[1] T.S. Abdul-Razaq, C.N. Potts, and L.N. Van Wassenhove. 1990. A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics* 26, 2 (1990), 235–253. https://doi.org/10.1016/0166-218X(90)90103-J

[2] Thomas Bartz-Beielstein and Martin Zaefferer. 2017. Model-based methods for continuous and discrete global optimization. *Applied Soft Computing* 55 (2017), 154–167.

[3] H. A. J. Crauwels, Chris N. Potts, and Luk N. Van Wassenhove. 1998. Local Search Heuristics for the Single Machine Total Weighted Tardiness Scheduling Problem. *INFORMS J. Comput.* 10, 3 (1998), 341–350. https://doi.org/10.1287/ijoc.10.3.341

[4] Matthijs den Besten, Thomas Stützle, and Marco Dorigo. 2001. Design of Iterated Local Search Algorithms. In *Applications of Evolutionary Computing*, Egbert J. W. Boers (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 441–451.

[5] Bilel Derbel and Sébastien Vérel. 2020. Fitness landscape analysis to understand and predict algorithm performance for single- and multi-objective optimization. In *GECCO '20: Genetic and Evolutionary Computation Conference, Companion Volume, Cancún, Mexico, July 8-12, 2020*, Carlos Artemio Coello Coello (Ed.). ACM, 993–1042. https://doi.org/10.1145/3377929.3389893

[6] Arkadiy Dushatskiy, Tanja Alderliesten, and Peter AN Bosman. 2021. 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.

[7] Anne Elorza, Leticia Hernando, and José Antonio Lozano. 2022. Transitions from P to NP-hardness: the case of the Linear Ordering Problem. In *IEEE Congress on Evolutionary Computation, CEC 2022, Padua, Italy, July 18-23, 2022*. IEEE, 1–8. https://doi.org/10.1109/CEC55065.2022.9870392

[8] Anne Elorza, Leticia Hernando, and Jose A. Lozano. 2023. Characterizing Permutation-Based Combinatorial Optimization Problems in Fourier Space. *Evolutionary Computation* (01 2023), 1–37. https://doi.org/10.1162/evco_a_00315

[9] William Fulton and Joe Harris. 2004. *Representation theory. A first course.* Springer.

[10] Martin Josef Geiger. 2004. An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters* 32, 1 (2004), 68–72.

[11] Martin Josef Geiger. 2010. On heuristic search for the single machine total weighted tardiness problem – Some theoretical insights and their empirical verification. *European Journal of Operational Research* 207, 3 (2010), 1235–1243.

[12] Jonathan Huang, Carlos Guestrin, and Leonidas Guibas. 2009. Fourier Theoretic Probabilistic Inference over Permutations. , 997-1070 pages. https://doi.org/10.1145/1577069.1577106

[13] Ekhine Irurozki, Borja Calvo, and Jose A. Lozano. 2011. Learning Probability Distributions over Permutations by Means of Fourier Coefficients. In *Advances in Artificial Intelligence*, Cory Butz and Pawan Lingras (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 186–191.

[14] Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. 2019. Data-Driven Evolutionary Optimization: An Overview and Case Studies. *IEEE Transactions on Evolutionary Computation* 23, 3 (2019), 442–458.

[15] Risi Kondor. 2010. A Fourier Space Algorithm for Solving Quadratic Assignment Problems. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms* (Austin, Texas) *(SODA '10)*. Society for Industrial and Applied Mathematics, USA, 1017–1028.

[16] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. 1977. Complexity of Machine Scheduling Problems. In *Studies in Integer Programming*, P.L. Hammer, E.L. Johnson, B.H. Korte, and G.L. Nemhauser (Eds.). Annals of Discrete Mathematics, Vol. 1. Elsevier, 343–362. https://doi.org/10.1016/S0167-5060(08)70743-X

[17] Florian Leprêtre, Sébastien Verel, Cyril Fonlupt, and Virginie Marion. 2019. Walsh Functions as Surrogate Model for Pseudo-Boolean Optimization Problems. In *The Genetic and Evolutionary Computation Conference, GECCO*. ACM, 303–311.

[18] Manuel López-Ibáñez, Francisco Chicano, and Rodrigo Gil-Merino. 2022. The Asteroid Routing Problem: A Benchmark for Expensive Black-Box Permutation Optimization. In *Proceedings of EvoApps 2022*, Vol. 13224 LNCS. 124–140. https://doi.org/10.1007/978-3-031-02462-7_9

[19] Dan Rockmore, Peter Kostelec, Wim Hordijk, and Peter F. Stadler. 2002. Fast Fourier Transform for Fitness Landscapes. *Applied and Computational Harmonic Analysis* 12 (1 2002), 57–76. Issue 1. https://doi.org/10.1006/acha.2001.0346

[20] P. F. Stadler. 1995. Toward a theory of landscapes. In *Complex Systems and Binary Networks*, R. López-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertruche (Eds.). Springer-Verlag, 77–163.

[21] Jörg Stork, Martina Friese, Martin Zaefferer, Thomas Bartz-Beielstein, Andreas Fischbach, Beate Breiderhoff, Boris Naujoks, and Tea Tusar. 2020. Open Issues in Surrogate-Assisted Optimization. In *High-Performance Simulation-Based Optimization*, Thomas Bartz-Beielstein, Bogdan Filipic, Peter Korosec, and El-Ghazali Talbi (Eds.). Studies in Computational Intelligence, Vol. 833. Springer, 225–244.

[22] Imanol Unanue, María Merino, and Jose A Lozano. 2021. A General Framework Based on Walsh Decomposition for Combinatorial Optimization Problems. In *Congress on Evolutionary Computation (CEC)*. IEEE, 391–398.

[23] Sébastien Vérel, Bilel Derbel, Arnaud Liefooghe, Hernán E. Aguirre, and Kiyoshi Tanaka. 2018. A Surrogate Model Based on Walsh Decomposition for Pseudo-Boolean Functions. In *Parallel Problem Solving from Nature - PPSN XV (Lecture Notes in Computer Science, Vol. 11102)*. Springer, 181–193.

[24] J. L. Walsh. 1923. A Closed Set of Normal Orthogonal Functions. *American Journal of Mathematics* 45, 1 (1923), 5.

[25] Martin Zaefferer, Jörg Stork, and Thomas Bartz-Beielstein. 2014. Distance Measures for Permutations in Combinatorial Efficient Global Optimization. 373–383. https://doi.org/10.1007/978-3-319-10762-2_37

[26] Martin Zaefferer, Jörg Stork, Martina Friese, Andreas Fischbach, Boris Naujoks, and Thomas Bartz-Beielstein. 2014. Efficient global optimization for combinatorial problems. *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, 871–878. https://doi.org/10.1145/2576768.2598282