Optics EXPRESS

Shapley-guided global optimization algorithm with applications in integrated photonics inverse design

MOHAMED SADEK RADWAN,[1,*] SEAN HOOTEN,[2] THOMAS VAN VAERENBERGH,[3] AND PETER BIENSTMAN[1]

[1] *Department of Information Technology, Ghent University/imec, Ghent, Belgium*
[2] *Hewlett Packard Labs, Hewlett Packard Enterprise, California, USA*
[3] *Hewlett Packard Labs, HPE Belgium, Diegem, Belgium*
[*] *Mohamed.Radwan@UGent.be*

Abstract: This study introduces an optimization algorithm, Shapley-Guided Stochastic Optimization (SGSO), which incorporates Shapley values to steer the search towards optimal solutions. The algorithm was tested on some well-known global optimization benchmark functions, like the Easom and Ackley functions, to validate its efficiency before applying it to more complex real-world scenarios, like the inverse design of photonic structures, specifically a 3dB splitter, a grating coupler, and a multilayer broadband mirror. The SGSO algorithm demonstrated its capability to direct the search process to generate highly performing designs while maintaining computational efficiency. Additionally, we propose a simplified approach for computing the Shapley values that can lower the algorithm's computational cost while still achieving satisfactory convergence to the global optimum. The results were benchmarked against Basin Hopping, one of the established metaheuristic optimization techniques, highlighting the potential of SGSO in navigating complex optimization landscapes. The SGSO is linked to Basin Hopping through the shared local optimization step and also shares connections with Genetic Algorithms, particularly in the crossover process between the different obtained solutions.

1. Introduction

Optimization is a key part of solving many complex problems, from engineering designs to machine learning models. Traditional methods like Gradient Descent, Genetic Algorithms [1], and Simulated Annealing [2] have been widely used in various fields to find the best solutions. However, as problems become more complex, in areas like inverse design in integrated photonics, there is a growing need for newer, more efficient algorithms.

Recent developments in combining machine learning with optimization have opened up new possibilities. One such advancement is the use of surrogate models, like neural networks, which can quickly estimate complex functions, making the optimization process faster and less resource-intensive [3]. Additionally, Shapley values, a concept from game theory, have been used in AI to explain the importance of different features in a model [4,5]. However, their application to optimization is still emerging, with limited literature directly addressing this approach [6]. This presents a novel opportunity to enhance search strategies by identifying key variables that most significantly impact the objective function.

In traditional optimization scenarios, variables are often treated with uniform significance across the search process. However, the intrinsic value of Shapley values lies in their ability to provide a quantifiable understanding of importance of each variable, thereby enabling a more focused and informed search strategy. Indeed, by pinpointing variables with higher influence on the objective function, optimization algorithms can allocate computational resources more

#543719 https://doi.org/10.1364/OE.543719
Journal © 2025 Received 2 Oct 2024; revised 24 Dec 2024; accepted 16 Jan 2025; published 12 Feb 2025

efficiently, prioritizing the exploration of variable configurations that have the most promise for reaching optimal solutions.

Lloyd Shapley introduced the concept of the Shapley values in his foundational work in 1951 to address the problem of fairly distributing reward given the outcome in a cooperative game [4]. The Shapley method ensures that each player's contribution to the collective effort is recognized in proportion to their input, considering all possible coalitions they could be a part of (a coalition refers to any group of players who cooperate to achieve a certain game's outcome). This concept is formalized as follows in Shapley's original work:

$$\phi_i = \sum_{S \subseteq F-i} \frac{|S|!(|F|-|S|-1)!}{|F|!}[v(S \cup i) - v(S)] \tag{1}$$

Here, ϕ_i represents the Shapley value for player i, which should correspond to this player's contribution to the cooperative game outcome. F represents the grand coalition including all players, and $|F|$ indicates the total number of these players. S symbolizes a subset of size $|S|$ of players excluding player i (i.e. a subset of the set $F - i$). $v(S)$ is the game outcome for coalition S. The term $v(S \cup i)$ denotes the game outcome when player i joins coalition S. The Shapley value therefore captures the marginal contribution of player i across all possible coalitions, providing a basis for fair reward distribution based on each player's contribution to the total game outcome.

Transitioning from game theory to the domain of explainable Artificial Intelligence (XAI), Shapley values have found a novel application in quantifying the contribution of individual features in artificial intelligence (AI) models. In this context, features of the AI model are treated analogously to players in a game, each contributing to the model's output or decision [7]. The Shapley values offer an approach to quantify the impact of each feature, enabling a deeper understanding of model behavior and enhancing transparency.

Contrary to the implementation in game theory, many of the AI models cannot handle the absence of one of their input features. In other words, they have to be assigned a full set of input values to be able to make predictions. As a way around this, we define two different input values: the 'background point' and the 'point to be explained'. The absence of a player in a certain coalition is represented by using the background value as the value for this player. In that case, the Shapley method quantifies the contribution of each input feature in transitioning the model's output from the background value to the point to be explained [8].

To illustrate the computation of Shapley values, let us consider an AI model with three input features represented by the set $F = \{x_i \mid i \in \{1, 2, 3\}\}$. In the context of Shapley value calculations, each input feature x_i can take one of two possible values: the background value $x_i = x_{i,b}$ or the value corresponding to the point being explained $x_i = x_{i,e}$. This results in a set of 8 possible coalitions:

$$\begin{aligned} \text{coalitions} = \{&(x_{1,b}, x_{2,b}, x_{3,b}), \\ &(x_{1,e}, x_{2,b}, x_{3,b}), \\ &(x_{1,b}, x_{2,e}, x_{3,b}), \\ &(x_{1,b}, x_{2,b}, x_{3,e}), \\ &(x_{1,e}, x_{2,e}, x_{3,b}), \\ &(x_{1,e}, x_{2,b}, x_{3,e}), \\ &(x_{1,b}, x_{2,e}, x_{3,e}), \\ &(x_{1,e}, x_{2,e}, x_{3,e})\} \end{aligned} \tag{2}$$

The calculation of the Shapley values as indicated in Eq. (1) requires many evaluations of the model outcome for every possible coalitions (as given by Eq. (2) for the above example), which corresponds to the Figure-Of-Merit (FOM) within the optimization framework. More specifically, an optimization problem of N variables (i.e. players within the Shapley framework) would correspond to 2^N coalitions and accordingly 2^N function evaluations. This can present a problem when a large number of players exists for a specific problem. Therefore, in the context of our algorithm, we also propose a simplifying approximation in the calculation of Shapley values to decrease the number of required function evaluations.

The rest of this paper is structured as follows. We first introduce the Shapley-Guided Stochastic Optimization (SGSO) algorithm, which uses Shapley values in a novel way to guide the optimization process. We then apply the SGSO to some of the well-known benchmark functions like the Easom and the Ackley functions, to show how it handles common optimization challenges. In addition, we present a way to simplify the Shapley values calculations to reduce its computational cost, which is an important aspect of optimization methods. Then, we apply the SGSO to more complex problems in integrated photonics inverse design, using neural network models as surrogate models to efficiently explore the design space of a 3dB splitter [9,10] and a grating coupler [11]. We also explore the applicability of our algorithm in the inverse design problem of a broadband thin-film mirror for thermophotovoltaic applications [12]. In that case, the mirror is modeled using a dedicated electromagnetic solver for multilayer structures [13].

2. Shapley-guided stochastic optimization (SGSO) algorithm

The methodology employed in developing this algorithm can be summarized as follows. Different local optima have different important features that contribute to their FOM. Accordingly, the primary objective is to preserve these important features when moving in the optimization space. The role of the Shapley values is to quantify this importance and the relative contribution of the different features. It is important to realise that the significance of features is comparative. For instance, the salient features that distinguish local optimum A when contrasted with local optimum B may not be the same as when local optimum A is compared to an arbitrary point C that is not a local optimum. Comparing A to a non-optimal point like C might put all features of A in a favorable light due to the relative suboptimality of C. However, it is the comparison with another high-quality point, such as B, that truly highlights the distinctive and superior features of A. In summary, meaningful distinctions in features' importance are highlighted only when high-caliber points are evaluated against each other, rather than against lesser counterparts. This is one of the main differences between our procedure and what was reported in [6]. A similar concept was presented in [14], where surrogate backbones are identified among different high-quality solutions for combinatorial optimization problems. These backbones are modified to facilitate exploration of the optimization space. However, a key distinction in our approach is that we use Shapley values to identify the values of variables that are common across different local optima, and maintain these values during the optimization process.

The algorithm proceeds as follows. Initially, a starting point is chosen randomly. Although an informed choice for initialisation might expedite convergence, a random selection is generally sufficient or can be the most suitable option for complex optimization functions or when we have less information on the problem at hand. From this initial point, a local optimization method is employed to identify the nearest local optimum. Subsequently, a second random starting point is generated and subjected to the same local optimization process to find a second local optimum. Next, we apply the Shapley method with the best local optimum so far as the "point to be explained" and the second local optimum as the "background". Then, a new starting point is created by mixing these two optima. The degree to which these two points are combined is determined by their respective FOM and the calculated Shapley values. To further clarify, a new point is constructed by retaining the feature values from the best local optimum with the highest Shapley values, while inheriting the features with the lowest Shapley values from the background. This method ensures that the new starting point maintains the strengths of the best solution while integrating potentially beneficial features identified through the Shapley values analysis from the background. This newly generated point is used as a new starting point for the local optimizer to get the associated local optimum point which is appended to our list of the generated local optima. This newly generated local optimum point is potentially superior to the previous two, as it combines important features from different local optima. The steps above represent one algorithm iteration, which is repeated a number of times, keeping track of the best overall local optimum, and each time generating a separate new local optimum as the background for the Shapley calculation.

The pseudocode of this algorithm is shown in Algorithm 1. It is worth highlighting that SGSO can be cast in the framework of Basin Hopping, since they share a local optimization step. This could be achieved by setting the temperature in Basin Hopping to zero and the generation of the Shapley-guided suggestion in the inner loop of our algorithm as a custom step-taking routine [15,16].

Algorithm 1. Shapley-Guided Stochastic Optimization

```
structures.append(optimize_locally(create_random_structure()))
**for** $i$ = 1 to $N_{alg}$ **do**
    optimized_random_structure = optimize_locally(create_random_structure())
    structures.append(optimized_random_structure)
    best_structure = structure_with_highest_FOM(structures)
    S_values = calculate_Shapley_values(structure_to_be_explained = best_structure,
                                        background = optimized_random_structure)
    # Create new structure starting from best one, replacing lowest contributors by optimized
random background structure.
    Shapley_guided_suggestion = best_structure
    $k$ = calculate_$k$(FOM(best_structure), FOM(optimized_random_structure))
    indices_of_worst_S_values = argpartition(S_values, $k$)
    Shapley_guided_suggestion[indices_of_worst_S_values]
                        = optimized_random_structure[indices_of_worst_S_values]
    structures.append(optimize_locally(Shapley_guided_suggestion))
**end for**
```

Finally, we want to clarify that the number of inherited parameters from the background k varies with each iteration of the algorithm and is given by:

$$k = \max\left(k_{\min}, \min\left(k_{\max}, k_{\max} - \left\lfloor \frac{FOM_{\text{best}} - FOM_{\text{background}}}{T_{\text{SGSO}}} \right\rceil\right)\right) \tag{3}$$

Here, k_{min} is the minimum number of parameters to be inherited which enforces a minimum rate of exploration. k_{max} is the maximum number of parameters that can be inherited, which limits the perturbation of the best obtained local optimum point. The parameter T_{SGSO} is inspired by other global optimization algorithms such as Simulated Annealing [2] or Basin Hopping [17]. It aims to restrict the inheritance process if the FOM of the best optimum point is much larger than that of the background. This is based on the interpretation that when the difference between the FOM values is large, there may be fewer important features in the background. Consequently, it is preferable to limit the extent of inheritance in such cases. In summary, we have three main hyberparameters that controls the SGSO's flow, k_{min}, k_{max}, T.

In addition, by going through the different iterations in the procedure, we are allowing the algorithm to learn about the different important features in the different local optima. Furthermore, the comparison with the best obtained optimum point and the proper adjustments of the number of the inherited parameters ensure that the most important features across the different local optima are given higher chance to remain in the newly generated points.

It should be noted that the correspondence between the optimization variables and the players for the Shapley values calculations can be chosen arbitrary. For instance, the simplest method involves treating each optimization variable as an individual player when forming all possible coalitions for the Shapley value calculations. Throughout this paper, we will refer to this approach as the exact method to calculate the Shapley values. However, in section 3.3, alternative choices that simplify the computation of Shapley values will be explored.

Also, it is important to emphasize again that when interpreting the Shapley method, higher values indicate a greater contribution of a variable in shifting the model output (i.e. the FOM) from the background to the point being explained. In the case of negative Shapley values, it implies that, on average, using the values of this variable from the background, rather than from the point being explained, leads to better performance (higher FOM). Furthermore, these interpretations are based on averages across all possible coalitions, not individual cases.

3. Results on Ackley and Eason benchmark functions

3.1. Benchmark definition

Our method underwent testing across a variety of optimization problems, each presenting a unique optimization landscape. We use the Easom and the Ackley functions [18], for their scalability to higher dimensions and the presence of numerous local optima within their optimization spaces. We flipped the sign of both functions to convert the problem to a maximisation one, as well as normalised them so that the global maximum value is 1. The maximization Easom function is given by

$$E(\mathbf{x}) = \prod_{i=1}^{n} \cos\left(f\left(x_i - \frac{\pi}{f}\right)\right) \cdot \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{\pi}{f}\right)^2\right) \tag{4}$$

This function possesses a global optimum at $\mathbf{x}^* = \left(\frac{\pi}{f}, \frac{\pi}{f}, \ldots, \frac{\pi}{f}\right)$. Manipulating the parameter f in Eq. (4) alters the location of this global optimum within the search space. It also affects the density of local optima, potentially increasing the problem difficulty. In addition, the Easom function possesses the unique feature of a relatively sharp global optimum, which makes it a rather difficult problem. Figure 1(a) illustrates a plot of this equation with the dimension of $\mathbf{x}$ set to two.

The normalised maximisation Ackley function is given by

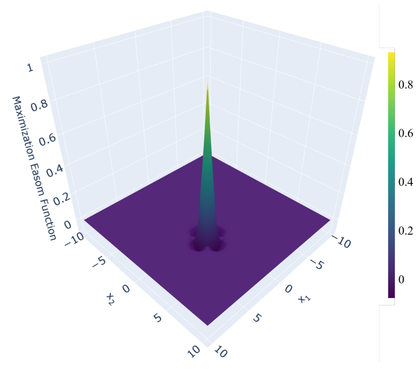
$$f(\mathbf{x}, n) = \frac{1}{a+e} \cdot \left(a \exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) + c \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(dx_i)\right)\right) \tag{5}$$

A plot of this equation is shown in Fig. 1(b) where the dimension of $\mathbf{x}$ is limited to two. It can be seen that this function features multiple local optima superimposed on a gradually ascending hill towards the maximum value.

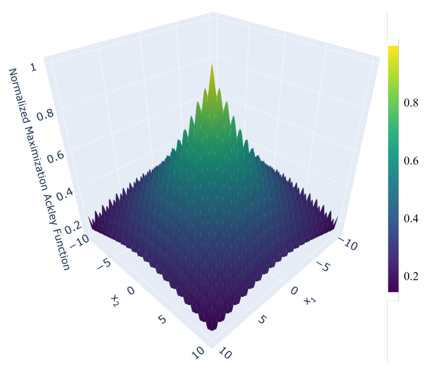
These characteristics make them robust tests for our algorithm's performance. For the testing of our various algorithms, we expanded the number of variables in the Easom and Ackley functions to ten, corresponding to a 10-dimensional optimization problem. It is also important to note that the difficulty of these problems is influenced by the limits of the search domain. In our results, we used a range of −10 to 10 for the Ackley function and −20 to 20 for the Easom function.

3.2. Performance and computational cost

The first aspect to be evaluated is whether the inclusion of Shapley values facilitates the algorithm's progression to more favorable positions within the optimization landscape. Later, we will consider



(a) 3D plot of the modified Easom function



(b) 3D plot of the modified Ackley function

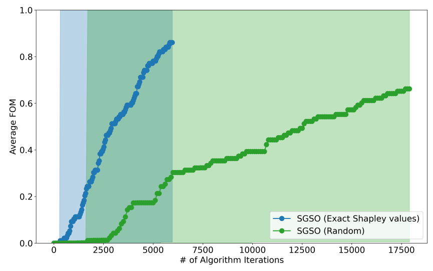
Fig. 1. 3D plot of the two benchmark functions, if necessary shifted such that the global optimum occurs at the origin. These two problems illustrate distinctly different landscapes. The Ackley function features multiple local optima superimposed on a gradually ascending hill towards the maximum value. In contrast, the Easom function has values close to zero almost everywhere, except for a sharp peak at the global optimum. Many local optima are present in the Easom function, but they are not visible in this zoomed-out plot due to their relatively low values.

the computational demands of calculating the Shapley values. In Fig. 2(a) and Fig. 2(b), we present a comparative analysis for the 10-dimensional Easom and the 10-dimensional Ackley functions, under two scenarios. The first one is the SGSO with guidance based on the Shapley values. The second scenario is the case where the Shapley values are replaced with random values, which serves as a baseline to compare with the performance obtained with the Shapley values. To generate these figures, we defined a permissible range for each optimization variable and generated a random starting point within this range, uniformly distributed. We then applied the optimization algorithm (with a certain number of algorithm iterations) and recorded the resulting FOM. This process was repeated 100 times, each time with a new starting point. The average FOM from these trials were displayed as convergence curves on a graph, with the upper and lower bounds of the obtained FOM illustrated as a shaded area around the curves. The x-axis in these two plots is used as the number of algorithm iterations (i.e. the jumps made in the optimization space). The upper and lower limits of the shaded areas around the average convergence lines in our plots effectively capture the variability of the results across multiple runs. Smaller differences between these upper and lower bounds indicate reduced variability, which corresponds to a plateau in the consistency curves described in [19], which suggests higher confidence in the performance of the respective algorithm.

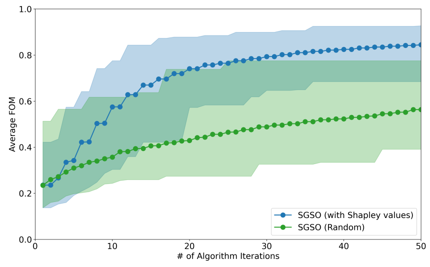
The obtained results in Fig. 2(a) and Fig. 2(b) clearly demonstrate the benefits of incorporating Shapley values, as evidenced by the improvement in convergence of the average FOM towards the global optimum of 1. For the Easom function, the curve does not saturate. So, the more function evaluations are allowed, the higher the probability to reach the global optimum point guided by the calculated Shapley values.

In the case of the Ackley function, while both our algorithm and the SSGO random baseline exhibit rapid convergence (note the difference in horizontal scale compared to Easom), the utilization of Shapley values still distinctly enhances performance.

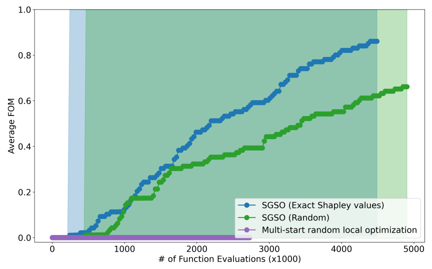
The results discussed here are generated using the COBYLA method [20] as a local optimizer. Additional results, which are detailed in Supplement 1 Section 6, utilize L-BFGS-B [21] as the local optimizer and show similar results. Additionally, the hyperparameters for these results and the subsequent results for the different examples are summarized in Supplement 1 Section 7.



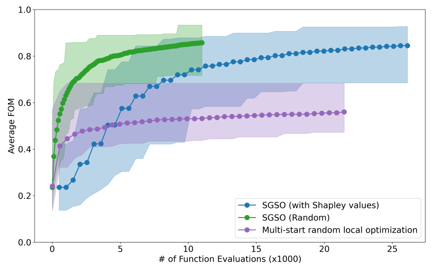
(a) Average FOM vs. number of algorithm iterations for the Easom function.



(b) Average FOM vs. number of algorithm iterations for the Ackley function.



(c) Average FOM vs. number of objective function evaluations for the Easom function.



(d) Average FOM vs. number of objective function evaluations for the Ackley function.

Fig. 2. Comparative analysis of the average FOM when the new point is based on Shapley values (SGSO - Exact Shapley values), and without it (SGSO - Random), in addition to a baseline method using multi-start random initialization followed by local optimization. For each x-value, these averages were calculated by applying the different algorithms to 100 randomly generated initial points for the 10-dimensional Easom and 10-dimensional Ackley test functions. (a) and (b) are plotted as a function of the number of algorithm iterations. This highlights the contribution of the Shapley values to the algorithm performance. (c) and (d) use the number of the objective function evaluations on the x-axis, which emphasises the effect of the computational cost associated with calculating the Shapley values on the algorithm convergence. The shaded regions represent the range of outcomes, outlining the maximum and minimum values obtained across the 100 random initial conditions.

It is important to note that although the ten variables are interchangeable in these two benchmark functions, their contributions to the objective function at a specific point are not necessarily equal. To illustrate this with a simple example, consider a function $f(x, y) = x + y$. While x and y are interchangeable by definition, their contributions to the function value at a specific point, such as $(5, 1)$, are unequal: $f(5, 1) = 6$, but x contributes five units, whereas y contributes one unit. This reasoning can be extended, albeit more crudely, to complex functions.

In the context of Shapley values, they are designed to capture the local importance or contribution of each variable at a specific point with respect to a given background point. This aligns with their role as a local explanation method in explainable AI (XAI), as opposed to global explanation methods. In XAI, the focus is on explaining why a model provides a specific prediction for a particular set of input values, rather than how the model behaves in general.

From an optimization perspective, we interpret Shapley values as providing a means to sense the surrounding landscape of the objective function locally. They capture the relationship between the point being evaluated and the background point, thereby guiding the optimization process in a favorable direction.

From the previous discussion, we can see that Shapley values help our algorithm reach better solutions. However, we need to consider whether the extra calculations for Shapley values are cost-effective. Could the effort be better spent elsewhere, such as allowing the algorithm more random search attempts? To tackle this question, we again turned to our two test functions, this time plotting the average FOM against the total number of objective function calls as shown in Fig. 2(c) and Fig. 2(d). These calls include all evaluations, whether for calculating Shapley values or for the optimization process itself, thus reflecting the overall computational expense. In addition, we added another baseline represented by a simple approach where we locally optimise a new random initial point each iteration and keep track of the best obtained FOM. We refer to this as multi-start random local optimization. This procedure favors optimization space exploration compared to the SGSO.

With the Easom function, the investment in computing Shapley values pays off, enhancing performance without increasing the overall computational cost. On the other hand, when applying our algorithm to the Ackley function, we observe a less efficient trend, with slower progress compared to the random version of the SGSO. This suggests that for the Ackley function, it seems better to skip the Shapley calculations in favor of increased random sampling. This effectiveness likely varies with the specific landscape of the objective function being optimized for the different problems. Furthermore, in both cases, the SGSO with its different versions outperforms the multi-start random local optimization. This highlights the value of mixing the local optima to generate new starting points.

Still, it should be noted that although calculating Shapley values increases the total count of function evaluations, these calculations are inherently parallelizable. This parallel processing is particularly advantageous when using analytical functions or surrogate models built on neural networks. In contrast, executing more random jumps in the algorithm is an inherently sequential process. Indeed, each jump relies on the position and information of the previous one, alongside the local optimization steps, making it a more time-intensive approach. However, we chose to adopt a worst-case scenario and keep the x-axis as total number of objective function evaluations, i.e. assuming that all the calculations, including those for the Shapley values, are done sequentially. The multi-start gradient methods can also be parallelized, which could reduce their computational time. However, as demonstrated in the results for the Easom and Ackley functions, the SGSO algorithm outperformed the multi-start gradient method in both cases. This suggests that even with equal parallel processing capabilities, the SGSO algorithm demonstrates superior performance compared to the multi-start approach under the same computational constraints.

Additionally, we will now discuss a technique to reduce the computational effort, by performing an approximation for the Shapley values.

3.3. Approximating the Shapley values

The formula given by Eq. (1) is a generic form to calculate Shapley values for a given game with a certain number of predefined players. However, with an increasing number of players, this can become computationally expensive. There exist different approximations for calculating Shapley values specifically designed for the AI models, like SHAP [8] and DeepLIFT [22], with reduced computation intensity. In this section, we introduce an alternative method: we use the same Eq. (1), but we modify the game and redefine the players for whom we are determining the importance in order to reduce the computational cost.

Within the context of optimization, we have some freedom in the definition of the players. For instance, nothing prevent us from grouping a certain number of optimization variables and assume that they will act as a single player for the Shapley value calculation. In that case, the Shapley values will correspond to the average contribution of this group as a whole. As an example, let us consider the Ackley function optimization problem as explained earlier. Suppose we assign a separate player to each of the 10 optimization variables. The number of different coalitions in that case is 2^{10}, which also corresponds to the number of function evaluations for the Shapley values calculations. Alternatively, to compute the Shapley value for a specific optimization variable, we can treat that variable as a single player in the game while grouping all other optimization variables together as a second player. This approach reduces the problem to a 2-player game, resulting in 2^2 possible coalitions. To determine the Shapley values for all the optimization variables, a separate 2-player game is constructed for each variable, where the selected variable acts as an individual player, and the remaining variables are grouped together. Each of these 2-player games generates 2^2 coalitions. However, since the background point and the point to be explained are shared across these games, the total number of unique coalitions is $2N + 2$, where N is the number of optimization variables. The additional 2 evaluations come from the FOM calculations at the background point and the point being explained. While these calculations might be considered redundant, as they were already performed during the local optimization step, we prefer to include them explicitly to highlight the computational cost associated with Shapley value calculations separately. Accordingly, the total number of evaluations, and thus the computational cost, is significantly reduced.

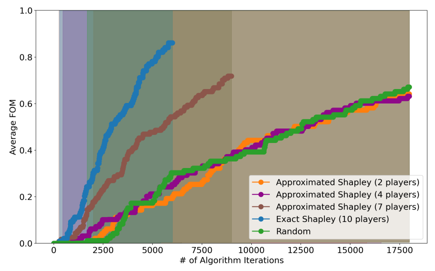
It is important to recognize that how we group certain players together as a single entity can significantly influence the resulting Shapley values. It might be reasonable to suggest that players who exhibit higher correlations should be grouped together. Although this effect might not be very pronounced in the Ackley and Easom functions, it could become more relevant in real-world scenarios, such as the inverse design photonics problem discussed later in this paper. However, such grouping depends heavily on the designer's understanding of the problem and their intuition about the interactions between the players.

Therefore, in this paper, we opted for the simpler choice where we grouped players based purely on their sequential order, assuming minimal prior knowledge or intuition about the interactions involved. This approach might seem less sophisticated, but it allows us to explore the baseline behavior without additional assumptions and it could also be considered as a first step to evaluate whether more sophisticated approaches are needed or not. As an example of how we performed player grouping, consider e.g. an optimization problem involving 10 variables $\{x_1, x_2, \ldots, x_{10}\}$ and 4 players $\{p_1, p_2, p_3, p_4\}$. If we want to determine the Shapley values for x_4, we treat the optimization variable x_4 as player p_1. The remaining variables are then sequentially distributed among the other players in roughly equal groups. Thus, we assign $p_2 = \{x_1, x_2, x_3\}$, $p_3 = \{x_5, x_6, x_7\}$, and $p_4 = \{x_8, x_9, x_{10}\}$.

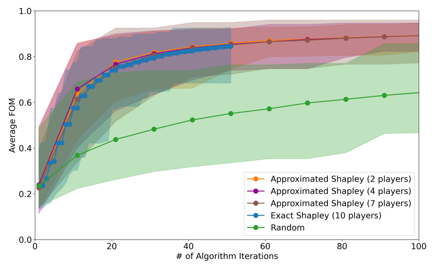
Let us first evaluate the Easom benchmark with a varying numbers of players. Figure 3(a) and Fig. 3(c) show that the results obtained with exact calculations of the Shapley values still outperform the algorithm with reduced number of players. This is true both as a function of the number of iterations and the number of function evaluations. It seems that the approximated Shapley values with fewer players provided a much cruder approximation, which significantly degraded the performance. Therefore, it appears that reducing the number of players in the algorithm slowed down its convergence, counteracting the benefits of lowered computational demands.

The situation is very different for the Ackley benchmark. Remember that using the exact Shapley values, there was no benefit in terms of number of function evaluations compared to random SGSO. Now however, Fig. 3(d) shows that reducing the player count even down to two decreases the computational cost significantly without sacrificing too much accuracy, resulting in enhanced performance relative to the random variant. However, if the plot were extended over a longer computational effort, the performance for the exact Shapley values would likely catch up and surpass the approximated version, albeit at a much higher computational cost.

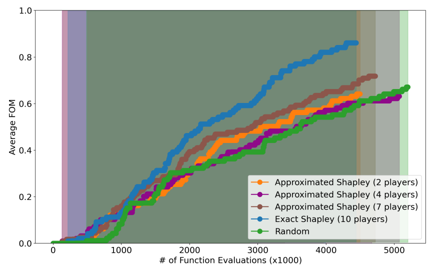
These contrasting outcomes illustrate that, while the proposed approximations may be effective for certain problems, they might not yield the best results for others. In summary, adjusting the number of players offers a practical approach to approximate Shapley values when computing the exact values is computationally expensive. Ideally, if the computational resources allow, we



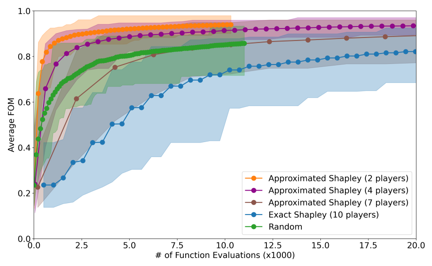
(a) Average FOM vs. number of algorithm iterations for the Easom function.



(b) Average FOM vs. number of algorithm iterations for the Ackley function.



(c) Average FOM vs. number of the objective function evaluations for the Easom function.



(d) Average FOM vs. number of the objective function evaluations for the Ackley function.

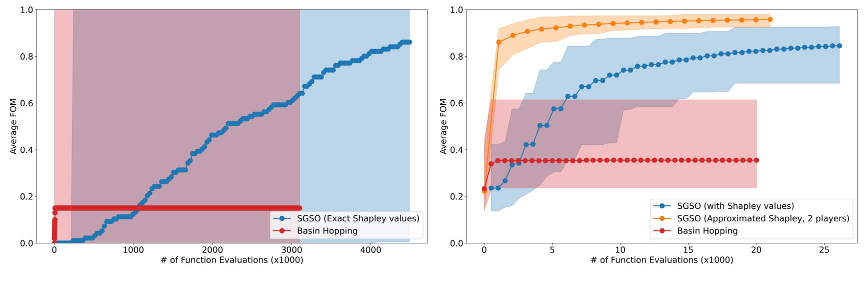
Fig. 3. Convergence for varying player counts for Shapley value calculation (Approximated Shapley (xx players)) in addition to the random version of the SGSO (Random). (a) and (b) depict convergence comparisons with the *x*-axis denoting algorithm iterations for Easom and Ackley functions respectively. (c) and (d) plot the average FOM against the total number of function evaluations. The results indicate that for the Easom function, the approximation inaccuracies result in a degraded performance. However, for the Ackley function, we observe a significant reduction in computational cost without compromising convergence efficiency.

recommend using the exact Shapley values (i.e., the maximum number of players) to ensure the most accurate results.

3.4. Comparison with Basin Hopping

For a thorough assessment of our algorithm, it was also benchmarked against Basin Hopping, a commonly used optimization technique. Detailed information regarding the selection and configuration of parameters for Basin Hopping, including the setup of its local optimizer, can be found in Supplement 1 Section 3. We experimented with various temperatures and step sizes within the Basin Hopping algorithm, which are pivotal in balancing the trade-off between exploring new areas and refining existing local optima. Here in the main text, we will only present the results when the COBYLA was used as a local optimizer. Results with the L-BFGS-B method as a local optimizer are similar and are presented in Supplement 1 Section 6. Using these specific hyperparameters and the COBYLA optimizer, the comparative performance for our two benchmark functions is presented in Fig. 4.

It is important to acknowledge that the hyperparameters selected for Basin Hopping in this study may not be optimal, and the results presented here, as well as in subsequent sections, reflect these specific conditions. The same applies to SGSO and its hyperparameters, as further optimization could be pursued. Generalizing the conclusions across all possible hyperparameter configurations is however nearly impossible. While we followed a methodology to select the hyperparameters for Basin Hopping, any further refinement would essentially introduce a new



(a) (b)

Fig. 4. Performance comparison of SGSO against Basin Hopping. (a) Easom function (The results for the approximated Shapley values were omitted here since they did not provide any enhancement) (b) Ackley function.

optimization problem layered on top of the original one. This challenge is inherent to heuristic optimization algorithms, making it difficult to fairly compare different methods under varying conditions.

For the Easom function, the average FOM for Basin Hopping was only 0.2. In contrast, the SGSO algorithm reached 0.8. The limited success for Basin Hopping might be explained as follows: if the initial point falls closer to the global optimum, Basin Hopping manages to reach this global optimum after a few jumps. However, if it starts from a point far away, then there is a smaller probability that it will be able to reach this global optimum with random jumps. Therefore, we would expect higher probability of convergence if the search domain were to be reduced. Indeed, more results for Basin Hopping with a smaller search domain are presented in Supplement 1 Section 5, which show better performance with a smaller search range.

The results for the Ackley function, depicted in Fig. 4(b), highlight that the SGSO using approximate Shapley values with two players achieved the fastest convergence and highest FOM compared to Basin Hopping and the SGSO with the exact Shapley method. However, none of the algorithms reached the global optimum of 1, as indicated by the upper bounds of their shaded areas. Still, the shaded area for the SGSO with approximate Shapley values exceeds a FOM of 0.98, suggesting a convergence to local optima very close to the global. This suggests that with additional time, it might converge to the global optimum.

Finally, in Fig. 5 we plot the FOM histogram after approximately 5000 evaluations of the Ackley function. This histogram can be considered as representing the probability distribution function of reaching the global optimum point for the different algorithms. We can see that the

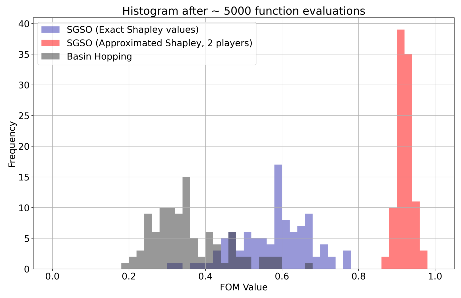


Fig. 5. Histogram for the obtained FOM for Ackley function after approximately 5000 function evaluations. The plot shows a comparison for the SGSO with exact and approximated Shapley in addition to Basin Hopping.

SGSO with the approximate Shapley values falls almost completely above 0.9, while the results for the other two are scattered across the FOM range. (As a side note for the Easom function, the histogram was not shown since it does not contain any interesting distribution: the FOM either takes 1 or values very close to 0 in the histogram.) The figure also shows that also the SGSO with exact Shapley values is more likely to converge to higher quality local optima than Basin Hopping.

4. Application to integrated photonic inverse design

We extended our testing to practical scenarios within the domain of integrated photonic inverse design. We chose three inverse design challenges: a grating coupler, an integrated 3dB splitter and broadband mirrors for thermophotovoltaics applications. The broadband mirror is simulated with the transfer matrix method [13]. Evaluating the performance of the first two structures, the grating coupler and the 3dB splitter, can be performed using finite-difference (FD) simulations implemented using commercial software like Lumerical [23] or an open-source tool like EMopt [24]. However, the need for multiple FOM evaluations to compute Shapley values means that FD simulations can become too time-intensive, particularly when testing various algorithm configurations. To avoid this, we developed surrogate models employing neural networks to simulate these two photonic problems. These surrogate models dramatically reduced the algorithm's runtime from several hours to few seconds, greatly enhancing the practicality and efficiency of our approach. However, this improvement comes at the expense of reduced accuracy in simulating the actual performance of the device since we employed relatively simple neural network architectures. More advanced approaches like physics-informed neural networks [25] could further enhance the accuracy, but are outside the scope of this paper.

4.1. Inverse design applied to a grating coupler's surrogate model

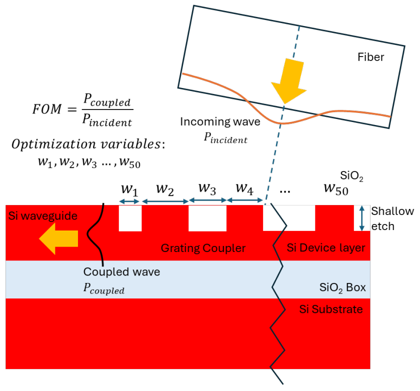
The grating coupler is commonly used to facilitate light transfer between a photonic chip and an optical fiber, featuring a periodic structure. However, challenges such as mode mismatch, substrate coupling, and wavevector matching into the reflected mode can significantly reduce its efficiency [26]. As a result, optimization techniques are frequently used to enhance the grating coupler's performance, aiming to balance these various factors for optimal coupling.

The schematic representation of the considered grating coupler is shown in Fig. 6, where the widths of the Si teeth and the gaps of the SiO_2 correspond to various optimization variables. In this case, there are 25 pairs of Si and SiO_2, resulting in a total of 50 optimization variables, corresponding to a 50-dimensional optimization space. Exactly calculating the Shapley values for the grating coupler's 50 design variables would require an infeasible 2^{50} function evaluations, so we resorted to the two-player approximation described above. The FOM is determined by the efficiency of the coupling from the input waveguide to the optical fiber, and is calculated using a surrogate model based on a machine learning model. This model is based on a simple machine learning architecture that could capture the general trends in the optimization landscape of the original problem, but which isnot as accurate as the conventional FD simulations methods. More details on this model are given in Supplement 1 Section 2.

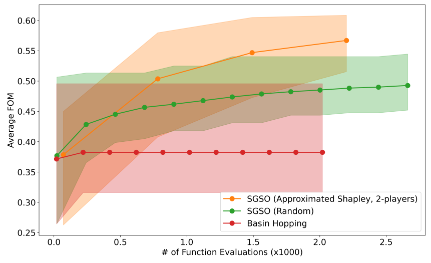
The results for the different algorithms are shown in Fig. 6(b). The two-player SSGO showed superior performance compared to both the random version of the SGSO and the Basin Hopping algorithm. For instance, SGSO achieved an average FOM of >0.55, compared to 0.5 for its random version and 0.4 for basin hopping. Such differences can be particularly significant in the context of complex photonic integrated circuits.

4.2. Inverse design applied to 3dB splitter's surrogate model

A 3dB splitter divides an incoming light wave into two equal parts. Its operational principle often relies on multimode interference, which adds complexity to the design process. We parameterized



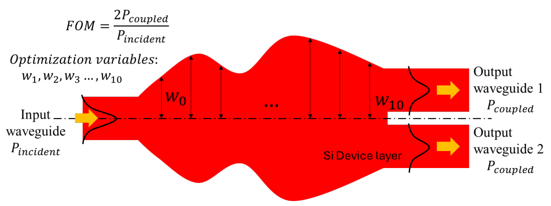
(a) Integrated grating coupler



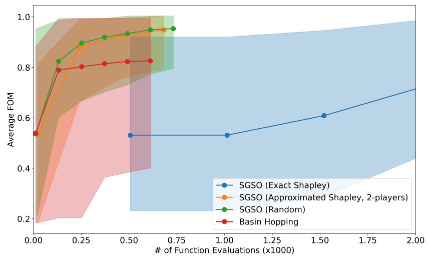
(b)

Fig. 6. (a) Schematic for the grating coupler showing its optimization parameters and the FOM. The incident wave coming from the fiber is transformed by the silicon teeth (whose widths are the optimization parameters) into a propagating waveguide mode. The efficiency of the coupling process represents the FOM. Twenty-five grating periods were used, corresponding to a 50-dimensional optimization space. (b) Performance comparison of SGSO with its random version and Basin Hopping applied to the grating coupler's surrogate model. The exact evaluation the Shapley values is omitted due to the infeasibility of calculating it with a large number of variables.

the 3dB splitter structure with 10 optimization variables as shown in Fig. 7(a). The FOM is defined as the coupling from the input waveguide to the output waveguides. A surrogate model based on neural networks was developed for the 3dB splitter to predict the FOM given the values of the different widths. Detailed information about this model is available in Supplement 1 Section 1. In this case, the calculation of the exact Shapley values is feasible with $2^{10} = 1024$ function evaluations.



(a) Integrated 3dB Splitter



(b)

Fig. 7. (a) Integrated 3dB splitter with its associated parameters and FOM. The drawing shows an incident light wave from the input waveguide on a silicon slab with varying geometry. This wave should be split into two waves that are coupled into the two output waveguides with minimal reflections and radiation losses. The geometry was parameterized using 10 different widths, using linear interpolation and smoothing. (b) Performance comparison of SGSO with its random version and Basin Hopping for 3dB splitter's surrogate model.

The outcomes for the photonic design problems are displayed in Fig. 7(b). Incorporating Shapley values did not yield any performance gains, with results matching those of the random approach, even when the approximate Shapley values were utilized. This indicates that the

Shapley values may not effectively identify the important features for the 3dB splitter problem, or the simplifications made to the Shapley value calculations compromised their accuracy. Therefore, precise calculations might be necessary to maintain accuracy, but this approach adds a high computational cost that outweighs simply exploring the optimization space. The results for the 3dB splitter are not surprising, as we do not anticipate our algorithm to outperform every other algorithm for every problem, in line with the so-called 'No Free Lunch' theorem [27]. Still, we can observe that the SGSO either with random Shapley values or the approximated version is doing better than Basin Hopping with its hyberparameters as described in the supplementary material. This difference (>0.15) between the SGSO and Basin Hopping represents a significant performance improvement, particularly in scenarios involving multiple 3dB splitters within a complex photonic integrated circuit.

4.3. Inverse Design applied to transfer matrix model of broadband mirrors for thermophotovoltaics applications

Our next step involved employing our algorithm to design a broadband mirror for thermophotovoltaic applications. These mirrors must provide high reflectivity across a broad wavelength spectrum and consist of a stack of layers made from different materials with alternating refractive indices as shown in Fig. 8. The thickness of each layer serves as an optimization variable to achieve the required high reflectivity over an extensive wavelength range. A greater number of layers introduces more degrees of freedom, potentially increasing the achievable reflectivity. We adhered to the problem description/settings from [12], but we omitted the use of a gold substrate to have a larger range of variation in the obtained FOM, to help us in the comparison between the different algorithms. We tested the SGSO algorithm in two scenarios: one with six Si and SiO_2 layers, and another with ten layers. For these experiments, we utilized an electromagnetic solver based on the semi-analytical transfer matrix method, as described in [13]. The local optimizer within the SGSO and Basin Hopping was based on the gradient calculation method from the same paper, and facilitated faster local optimizations relative to finite difference gradients evaluations. Further details on parameter adjustments for Basin Hopping are discussed in Supplement 1 Section 3.

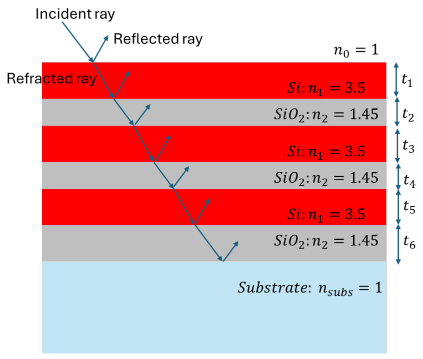


Fig. 8. Schematic representation of a broadband multilayer mirror composed of alternating Si/SiO_2 layers, with air as both the substrate and the incident medium. The thickness of each layer is an optimization parameter designed to enhance the mirror's reflectivity across its operational bandwidth. Incident rays from the air are both refracted and reflected at the initial layer interface. Subsequent layers induce multiple reflections and refractions at various interfaces. The phase differences between the reflected rays, governed by layer thickness, result in either constructive or destructive interference at different wavelengths. By precisely adjusting these thicknesses, high reflectivity can be achieved across a broad wavelength range and for various angles of incidence.

The results for the two scenarios with different numbers of layers are depicted in Fig. 9. We chose to employ only the SGSO algorithm with approximate Shapley values to keep the optimization time within reasonable bounds. The SGSO clearly outperforms both its random counterpart and the Basin Hopping algorithm. In the 6-layer scenario, after 2400 function evaluations, the SGSO achieves approximately 1.5% higher reflectivity than its random version and about 3.7% higher than Basin Hopping. For the 10-layer case, the improvements are around 1% and 2.7% over the random version and Basin Hopping, respectively.

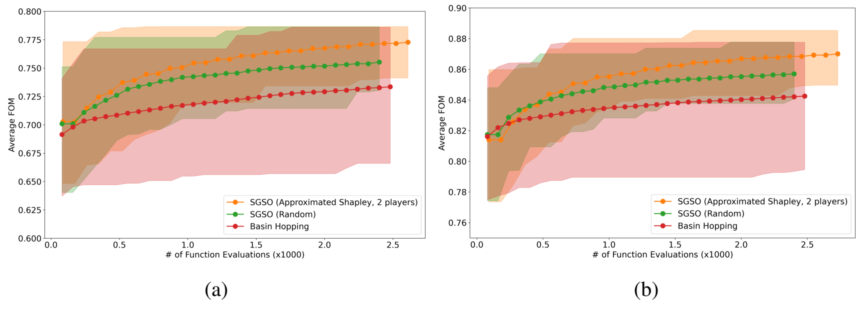


Fig. 9. Comparison of the performance of SGSO (using approximate Shapley values) against its random counterpart and Basin Hopping for optimizing a multilayer broadband mirror. (a) presents results for configurations with 6 layers (6-dimensional optimization problem), and (b) for 10 layers (10-dimensional optimization problem). The SGSO consistently outperforms the other two algorithms. In the case of 10 layers, the FOM is higher because the larger number of degrees of freedom.

While the performance differences between the algorithms are less pronounced, the SGSO algorithm still demonstrated faster convergence. The practical significance of these differences is highly application-dependent. In some scenarios, even fractional improvements in the FOM can be crucial. For example, in a ring laser gyroscope, achieving mirrors with extremely high reflectivity (>0.999) is essential to mitigate the lock-in phenomenon [28,29], which restricts the measurement of minimal angular rotations. In such cases, optimizations that deliver even slight performance enhancements can have a substantial impact on overall system functionality and precision.

5. Conclusions

In this study, we introduced a novel optimization algorithm named SGSO, which incorporates the concept of Shapley values from game theory. Shapley values can be particularly useful for pinpointing important features at local optima which can be exploited throughout optimization framework. The SGSO is related to evolutionary algorithms, but equipped with Shapley values to decide which features to retain in subsequent generations, in addition to employing local search from Basin Hopping to reach local optima with relatively high-quality features. We assessed our algorithm's performance by comparing it with Basin Hopping, a conventional method for global optimization. Our evaluations were conducted using two well-known benchmark functions and three problems in integrated photonics inverse design. Two of the inverse design problems, the grating coupler and the 3dB splitter, were addressed through neural networks as surrogate models. The third example, the broadband mirror, was modeled using an electromagnetic solver based on the transfer matrix method. The SGSO ultimately demonstrated superior performance, exhibiting faster convergence and a higher likelihood of reaching better solutions in four out of these five cases—the Easom function, the Ackley function, the grating coupler and the broadband mirror (in both cases, 6 and 10 layers). The SGSO also outperformed Basin Hopping on the fifth problem, the 3dB splitter. However, this could be due to the algorithm's procedure itself rather than the insights provided by the Shapley values, since it did not surpass its random version. This variance is anticipated, as no single optimization algorithm is expected to be universally effective across all problems. Additionally, we proposed and tested a simplification of the Shapley value calculations. This modification proved crucial for certain problems, such as the Ackley function, the grating coupler and the broadband mirror. However, it did compromise the accuracy of the Shapley values in the case of the Easom function.

The three integrated photonics inverse design problems showed relatively promising performance for the SGSO, highlighting its potential applicability in real-world inverse design challenges. However, the optimization was conducted using surrogate models based on neural networks in the cases of the grating coupler and the 3dB splitter. Consequently, the accuracy of these models may be questionable, which can limit the direct use of the generated optimal designs without further tweaking. Nevertheless, the landscape of the optimization space of the developed surrogate models is likely to retain some characteristics of the original space of the actual optimization problem. From a purely optimization perspective, the SGSO managed to achieve better outcomes across these varied landscapes. Additionally, the inverse design problem of the broadband mirror presents another demonstration where the surrogate model, based on an EM solver, yields more precise results. This serves as a more tangible example of the effectiveness of our method in the context of integrated photonics inverse design.

Funding. Flemish Fund for Scientific Research (FWO EOS GOH1422N); HORIZON EUROPE Framework Programme (Prometheus, Respite).

Acknowledgments. We are thankful to Xinling Yu for his constructive feedback on the surrogate model built for the grating coupler. Additionally, we thank Muhammed Gouda for his constructive feedback and discussions during the development phases of this algorithm.

Disclosures. The authors declare no conflicts of interest.

Data Availability. Data underlying the results presented in this paper are not publicly available at this time but may be obtained from the authors upon reasonable request.

Supplemental document. See Supplement 1 for supporting content.

References

1. K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the third international conference on Genetic algorithms*, (1989), pp. 42–50.
2. S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," Science **220**(4598), 671–680 (1983).
3. S. So, T. Badloe, J. Noh, *et al.*, "Deep learning enabled inverse design in nanophotonics," Nanophotonics **9**(5), 1041–1057 (2020).
4. L. S. Shapley, "A value for n-person games," (1953).
5. E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," Knowl. Informat. Sys. **41**(3), 647–665 (2014).
6. C. Yeung, D. Ho, B. Pham, *et al.*, "Enhancing adjoint optimization-based photonic inverse design with explainable machine learning," ACS Photonics **9**(5), 1577–1585 (2022).
7. B. Rozemberczki, L. Watson, P. Bayer, *et al.*, "The shapley value in machine learning," arXiv (2022).
8. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," Advances in neural information processing systems **30**, 1 (2017).
9. L. B. Soldano and E. C. Pennings, "Optical multi-mode interference devices based on self-imaging: principles and applications," J. Lightwave Technol. **13**(4), 615–627 (1995).
10. A. Michaels, M. C. Wu, and E. Yablonovitch, "Hierarchical design and optimization of silicon photonics," IEEE J. Sel. Top. Quantum Electron. **26**(2), 1–12 (2020).
11. R. Marchetti, C. Lacava, A. Khokhar, *et al.*, "High-efficiency grating-couplers: demonstration of a new design strategy," Sci. Rep. **7**(1), 16670 (2017).
12. Z. Omair, S. Hooten, V. Menon, *et al.*, "Broadband mirrors for thermophotovoltaics," Opt. Express **32**(7), 11000–11009 (2024).
13. Z. Omair, S. Hooten, and E. Yablonovitch, "Broadband mirrors with> 99% reflectivity for ultra-efficient thermophotovoltaic power conversion," in *Energy Harvesting and Storage: Materials, Devices, and Applications XI*, vol. 11722 (SPIE, 2021), pp. 13–18.

14. M. Mohseni, D. Eppens, J. Strumpfer, *et al.*, "Nonequilibrium monte carlo for unfreezing variables in hard combinatorial optimization," arXiv (2021).
15. P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," Nat. Methods **17**(3), 261–272 (2020).
16. D. J. Wales and J. P. Doye, "Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms," J. Phys. Chem. A **101**(28), 5111–5116 (1997).
17. Z. Li and H. A. Scheraga, "Monte carlo-minimization approach to the multiple-minima problem in protein folding," Proc. Natl. Acad. Sci. **84**(19), 6611–6615 (1987).
18. M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," Int. J. Math. Model. Numer. Optimisation **4**(2), 150–194 (2013).
19. P. Bennet, D. Langevin, C. Essoual, *et al.*, "Illustrated tutorial on global optimization in nanophotonics," J. Opt. Soc. Am. B **41**(2), A126–A145 (2024).
20. M. J. Powell, "A view of algorithms for optimization without derivatives," Mathematics Today-Bulletin of the Institute of Mathematics and its Applications **43**, 170–174 (2007).
21. C. Zhu, R. H. Byrd, P. Lu, *et al.*, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," ACM Trans. Math. Softw. **23**(4), 550–560 (1997).
22. A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International conference on machine learning*, (PMLR, 2017), pp. 3145–3153.
23. "Page title," https://www.lumerical.com. Accessed: 2024-04-10.
24. A. Michaels, "emopt: Electromagnetic optimization software," https://github.com/anstmichaels/emopt (2024).
25. G. E. Karniadakis, I. G. Kevrekidis, L. Lu, *et al.*, "Physics-informed machine learning," Nat. Rev. Phys. **3**(6), 422–440 (2021).
26. D. Taillaert, F. Van Laere, M. Ayre, *et al.*, "Grating couplers for coupling between optical fibers and nanophotonic waveguides," Jpn. J. Appl. Phys. **45**(8R), 6071 (2006).
27. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," IEEE Trans. Evol. Computat. **1**(1), 67–82 (1997).
28. M. Faucheux, D. Fayoux, and J. Roland, "The ring laser gyro," J. Opt. **19**(3), 101–115 (1988).
29. D. Yang, Y. Jiang, J. Zhao, *et al.*, "Measurement of low loss and mirrors' reflectivity using cavity ring down spectroscopy with high accuracy," in *2nd International Symposium on Advanced Optical Manufacturing and Testing Technologies: Optical Test and Measurement Technology and Equipment*, vol. 6150 (SPIE, 2006), pp. 35–41.