Which random is the best random? A study on sampling methods in Fourier surrogate modeling

Marco S. Nobile
Department of Industrial Engineering & Innovation Sciences
Eindhoven University of Technology
Eindhoven, The Netherlands
m.s.nobile@tue.nl

Simone Spolaor
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Milan, Italy
simone.spolaor@disco.unimib.it

Paolo Cazzaniga
Department of Human and Social Sciences
University of Bergamo
Bergamo, Italy
paolo.cazzaniga@unibg.it

Daniele M. Papetti
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Milan, Italy
d.papetti1@campus.unimib.it

Daniela Besozzi
Department of Informatics, Systems and Communication
University of Milano-Bicocca
Milan, Italy
daniela.besozzi@unimib.it

Daniel A. Ashlock
Department of Mathematics and Statistics
University of Guelph
Guelph, ON, Canada
dashlock@uoguelph.ca

Luca Manzoni
Department of Mathematics and Geosciences
University of Trieste
Trieste, Italy
lmanzoni@units.it

***Abstract*—Global optimization problems can be effectively solved by means of Computational Intelligence methods. However, there are several areas in which the effectiveness of these algorithms can be hampered by the computational costs of the fitness evaluations, or by specific features of the fitness landscape that can be characterized by noise and by the presence of several (even infinite) local optima. These issues bring about the necessity of defining specific techniques to replace the original problem with a surrogate representation. Fourier surrogate modeling represents a novel and effective approach to generate smoother, and possibly easier to explore, fitness landscapes, and to reduce the computational effort. Fourier surrogates require an initial sampling of the search space that must be performed to calculate the Fourier transforms. In this paper we investigate the impact on the quality of the surrogate models of the hyper-parameters of the methodology, and of several methods that can be employed for the initial sampling of the fitness landscape (i.e., pseudo-random numbers, low discrepancy sequences, a logistic map in chaotic regime, true random positions generated by a quantum computer, and point packing). Our results show that semi-structured approaches like quasi-random sequences and point packing can outperform the other sampling methods.**

***Index Terms*—Fuzzy Self-Tuning PSO, global optimization, surrogate modeling, Fourier transform, chaotic systems, logistic map, pseudo-random numbers generation, quasi-random numbers generation, quantum computing, point packing**

I. INTRODUCTION

Surrogate modeling is widely used in the context of global optimization, in order to define an approximate or simplified version of the fitness function, whose evaluation is (much) less expensive than the original version [1]. To this aim, various techniques have been employed, including genetic programming [2], artificial neural networks [3], polynomial regression [4], Kriging modeling [5], support vector regression [6], radial basis functions [7], or combinations of these methods to build local or global surrogate models [8]–[11]. Surrogate models are exploited in several engineering and scientific disciplines [1], [12]–[15], to reduce the relevant computational burden of optimization tasks, in which the evaluation of the fitness function represents the major bottleneck.

Surrogate models can be particularly useful when the optimization is carried out by means of Computational Intelligence methods, e.g., evolutionary computation or swarm intelligence algorithms [16]–[18], since these approaches typically require many evaluations of the fitness function to converge to an optimal solution. Surrogate models can also be useful for a second reason, i.e., filtering or smoothing out the fitness landscape. Taking into account both problems, we proposed surF [19], a method for generating smooth surrogate models of multimodal, rugged, and noisy fitness landscapes. surF leverages the Fourier transform to filter out the high frequency components of the landscape, in order to avoid the premature convergence of optimization algorithms into local optima. The surrogate fitness landscapes generated by surF preserve the advantage of being computationally less expensive to evaluate; at the same time, surF provides the user with the possibility of “tuning” the level of smoothness applied to the surrogate landscape by means of a specific hyper-parameter, corresponding to the number of low frequency spectral coefficients considered for the inverse Fourier transform. An additional hyper-setting of surF is the initial sampling of the search space that is used to create the surrogate.

In this paper we investigate the impact on the quality of the generated surrogate models of the hyper-parameters of surF, and of the sampling strategy employed. In particular, we test multiple sampling strategies, ranging from the absolute randomness provided by quantum random sampling, to the semi-regular coverage of the search space offered by entropic point packing algorithms. In order to investigate the impact

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

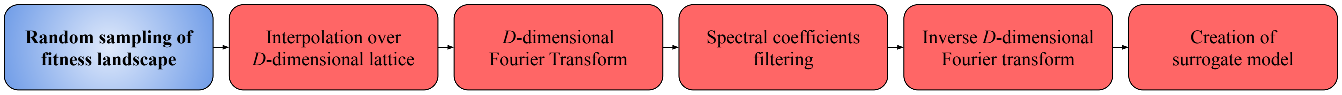
Random sampling of fitness landscape
Interpolation over D-dimensional lattice
D-dimensional Fourier Transform
Spectral coefficients filtering
Inverse D-dimensional Fourier transform
Creation of surrogate model

Fig. 1. Schematization of the surF surrogate modeling method: at the beginning of the procedure, a random sampling of the search space is applied (blue box). These samples are used to perform a Fourier filtering and create the surrogate model (red boxes), which can be later exploited to perform an optimization.

of these settings, we apply surF to three benchmark functions characterized by multi-modality and ruggedness: Ackley, Alpine and Rastrigin.

The remainder of the paper is structured as follows. Section II provides background information on the surF method for surrogate modeling, and on the random sampling strategies exploited in this work. Section III presents a comparison on the quality of the surrogate models obtained with different hyper-parameters settings and different sampling strategies, and the application of our methodology to a real world problem. Finally, in Section IV we conclude with some discussions and future developments of this line of research.

II. Methods

Given an optimization problem, we define the search space as the sub-space $F \subseteq \mathbb{R}^D$ of the feasible candidate solutions. In what follows, we assume that the search space is bounded within the interval $[\beta^d_{\mathtt{min}}, \beta^d_{\mathtt{max}}]$, for each $d = 1, \dots, D$. Note that sampling techniques typically produce random values in the unitary hyper-cube, which can be easily remapped within the boundaries of the search space.

A. Fourier surrogate modeling (surF)

The *discrete Fourier transform* (DFT) is a ubiquitous and extremely useful mathematical tool for the processing of discretized signals and information [20]. Although DFT is generally exploited to manipulate periodic and oscillating signals [21], we propose a completely different application in the context of global optimization based on Computational Intelligence. Specifically, we introduced surF [19], a new approach for fitness landscape surrogate modeling by means of Fourier filtering.

The surF methodology works as schematized in Figure 1: the algorithm begins by randomly sampling σ points in the search space (blue box). These random samples are then used to generate a surrogate model (red boxes) by creating an interpolated D-dimensional grid—having ρ points on each dimension—that is fed to a DFT to convert the “signal” of the fitness landscape into its frequency-domain representation. Then, a low-pass filter retains the first γ low frequency coefficients and sets to zero the rest of the spectra. Finally, this filtered representation is fed back to an inverse Fourier transform, which eventually builds the surrogate model of the fitness landscape. surF works under the assumption that the high frequency components of the fitness landscape can be discarded, and the global optimum is preserved when a limited number of low-frequency components is maintained.

One important question is how to select the value of γ. Intuitively, the smaller the value of γ, the smoother the landscape. However, when γ is too small, there might not be enough coefficients to correctly preserve the position of the global optimum, thus moving the global optimum in the surrogate function far from the real one. Therefore, the optimal value of γ should correspond to the minimum number of coefficients needed to preserve the position of the global optimum in the surrogate function. We expect this approach to be effective when the landscape can be decomposed in a few low-frequency components plus a high-frequency noise: by keeping only the lower frequency, most of the local optima should be removed, while preserving the position of the global optima. Though, in general, the best value of γ will depend on the characteristics of the fitness landscape, which are usually unknown.

The goal of this paper is to investigate the impact of different sampling strategies of the search space on the quality of the surrogate model (i.e., the blue box in Figure 1), combined with the variation of the hyper-parameter γ. A thorough description of the surF surrogate algorithm can be found in [19].

B. Search space sampling methods

a) Uniform sampling with pseudo-random sequences: The simplest method to sample a D-dimensional search space consists in generating random vectors using a pseudo-random number generator (here denoted by PSEUDO), which produces a sequence of (apparently) random uncorrelated numbers following a uniform distribution. In this work, each d-th value of these vectors is randomly sampled from a uniform distribution in $[\beta^d_{\mathtt{min}}, \beta^d_{\mathtt{max}}]$, exploiting the random numbers generator offered by the Python numpy library that uses the Mersenne Twister algorithm [22]. Mersenne Twister is a well-known robust generator characterized by an extremely long period of $2^{19937} - 1$, which makes it the most widespread choice for pseudo-random numbers generation. Nevertheless, since computer programs are deterministic in nature, the pseudo-random numbers are correlated (e.g., it is possible to predict future values by looking at a sufficient number of iterations—624 in the case of Mersenne Twister), and depend on the chosen seed value.

b) Quantum random numbers generation: Among the alternative options to pseudo-random sequences to generate true random numbers, quantum computing is possibly one of the most effective solutions. Namely, it is possible to exploit superposition, i.e., to prepare the following quantum state using a Hadamard gate:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle . \quad (1)$$

If a downstream measurement of the quantum register is performed, the quantum waveform collapses and there is a 50% probability of measuring a 0, and 50% probability of measuring a 1. By iteratively repeating the process, truly random bits can be aggregated and interpreted as random numbers of arbitrary precision. In this work, this task was performed by using the qRNG library version 0.2.1. We also exploited the Qiskit library version 0.13.0 [23], configured to access the IBM Q Experience computing infrastructures. Specifically, we run the code on the following backends: the 1 qubit machine ibmq_armonk and the 5 qubits machine ibm_essex. It is worth noting that the random number generation task performed with these machines (here denoted by QUANTUM) is very time consuming; for this reason, in the results presented in Section III, we used actual quantum random generation only for $\sigma = 50$ and $\sigma = 500$, while for $\sigma = 5000$ we exploited the ibmq_qasm_simulator backend.

c) Logistic maps: The logistic map is a relatively simple difference equation characterized by a complex chaotic behavior. The recurrence formula is defined as:

$$x_{n+1} = r x_n (1 - x_n), \quad (2)$$

with $r \in [0, 4]$ and $x_0 \neq 0$. The chaotic behavior arises for values of r greater than 3.56995. Due to its characteristics, the logistic map has been used to create random numbers generators [24]. In this paper, we employ a chaotic random numbers generator (denoted by CHAOS), using $r = 4$ and applying the following transformation (see [24] for additional information):

$$y_n = \frac{\arcsin(1 - 2x_n)}{\pi}. \quad (3)$$

d) Quasi-random low discrepancy sequences: Quasi-random number generators are designed to produce highly uniform samples of the unitary hyper-cube [25], [26]. Their goal is to minimize the discrepancy between the distribution of the sampled points, and a distribution with equal proportions of points in each sub-cube belonging to a uniform partition of the hyper-cube. Thanks to this low-discrepancy property, quasi-random numbers generators are able to more uniformly cover the search space than classic pseudo-random generators. In this work, we exploit the quasi-random generator based on the Sobol sequence (denoted by QUASI) offered by the Python package sobol_seq version 0.1.2.

e) Point packing: A point packing in the unit square is a placement of m points, aiming at the maximization of the geometric mean of the Shannon entropy [27]. This technique, which maximizes the evenness of marginal distributions in each coordinate, was used because it creates a good approximation to an evenly spaced lattice of points. The concept can be generalized to D-dimensional hyper-cubes, and multiple point packings are possible for the same hyper-space. In this work, we use a point packing algorithm (denoted by PPACK) whose representation exploits the Conway operator [28].

III. Results

Figure 2 shows some examples of surrogate models of the Ackley benchmark function (with $D = 2$), created by means of Fourier filtering. In the top-left figure we show the original fitness landscape, characterized by a single global optimum in $(0, 0)$ and several local minima. In the second column, we show the $\sigma = 50$ samples of the fitness landscape (red dots), which are used to create the lattice by means of the interpolation among the sampled values; in the background of these plots, the darker the color, the better the fitness. The five rows show the differences between the initial sampling methods, namely (top to bottom): QUANTUM, PSEUDO, CHAOS, QUASI, and PPACK. The sampling methods are sorted according to the "degree of randomness" of the produced points distribution. The lattice then undergoes Fourier filtering, where the high frequency spectral coefficients are removed and the surrogate model is created by means of inverse Fourier transform. The third, fourth and fifth columns show the surrogate models created using $\gamma = 3$, $\gamma = 5$ and $\gamma = 15$ coefficients, respectively. Overall, this figure clearly shows that different sampling methods can yield different surrogate models, by conveying different information into surF.

We observe that a more uniform coverage of the search space by means of QUASI and PPACK sampling allows for the creation of a surrogate model that faithfully approximates the original function, independently of the value of γ used. On the contrary, QUANTUM, PSEUDO and CHAOS strategies suffer from the low number of samples (i.e., $\sigma = 50$), especially in the case of 15 coefficients. In any case, the value of γ used to create the surrogate model has a strong effect on the final result.

We denote by f^o the original fitness function, and by $f^{\texttt{init}}_{\gamma,\sigma}$ the fitness function based on a surrogate model created by generating σ samples using the `init` sampling method (i.e., QUANTUM, PSEUDO, CHAOS, QUASI or PPACK), and considering the first γ spectral coefficients. For each combination of initial sampling method `init`, number of spectral coefficients γ, and number of initial samples σ, we calculate the average error with respect to the original function as follows:

$$error(\texttt{init}, \gamma, \sigma) = \frac{1}{R}\sum_{r=1}^{R} \left| f^o(\mathbf{x}_r) - f^{\texttt{init}}_{\gamma,\sigma}(\mathbf{x}_r) \right|, \quad (4)$$

where R is the number of random values taken from f^o and $f^{\texttt{init}}$ to compute the error. In all tests, we used $R = 10000$.

We calculated the error on three rugged and multi-modal benchmark functions, which represent problems that can benefit from the application of surF: Ackley, Alpine and Rastrigin. The hyper-parameters of surF that were tested are: $\gamma \in \{3, 15\}$ spectral coefficients, and $\sigma \in \{50, 500, 5000\}$ samples.

Although surF samples the fitness function f^o only σ times, the Fourier transform must be performed using a signal sampled at regular intervals. This is obtained by using the linear interpolation on a grid with ρ partitions per axis, leading

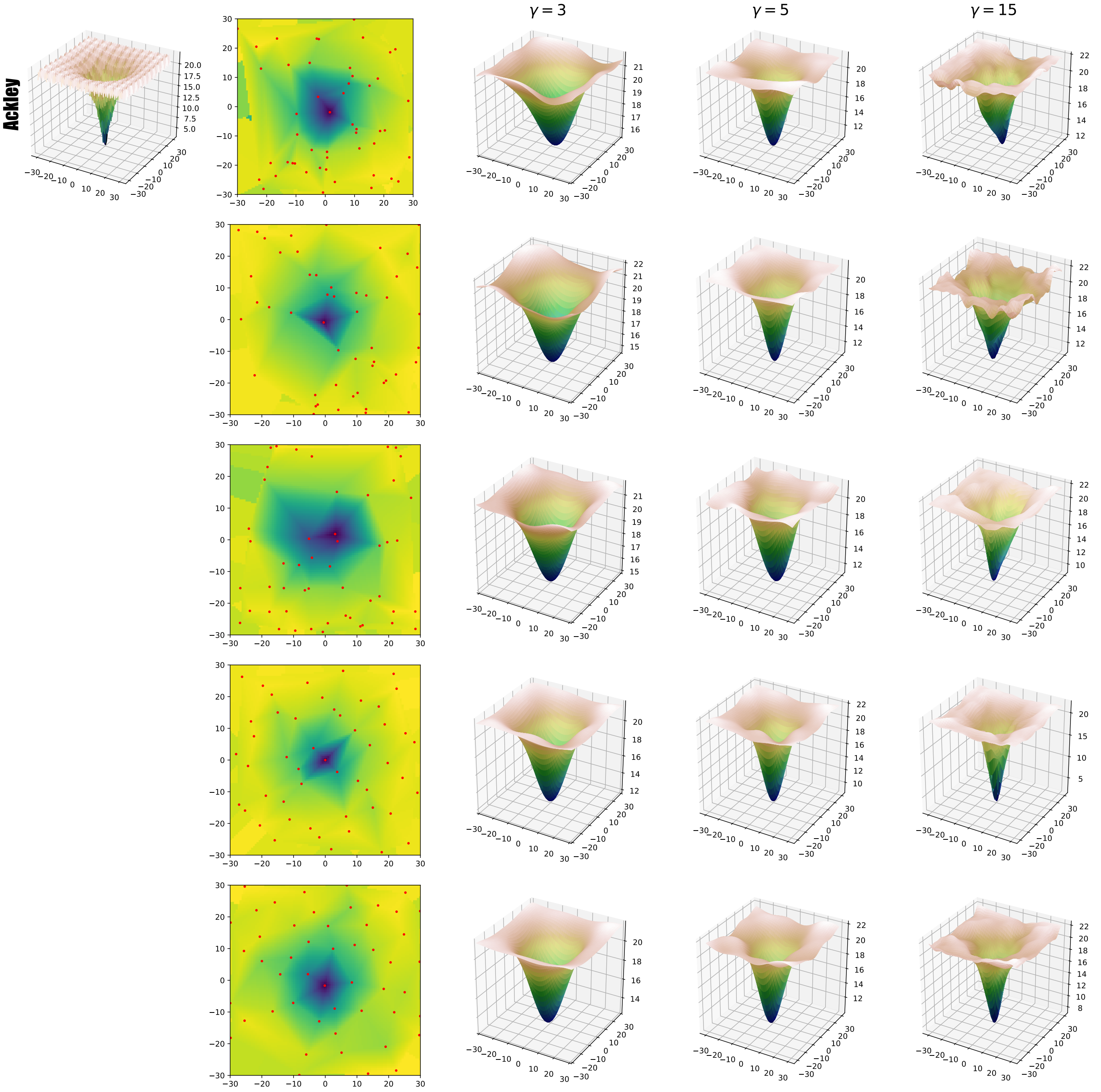

Fig. 2. Examples of Fourier surrogate models of the Ackley benchmark function created with Fourier filtering. The surrogate models are created using (from top to bottom) the QUANTUM, PSEUDO, CHAOS, QUASI, and PPACK sampling strategies. Example of $\sigma = 50$ sampled points are shown as red dots in the figures of the second column. The background color represents the linear interpolation of such points. The spectral coefficient used to create the surrogate models are $\gamma = 3$ (third column), $\gamma = 5$ (fourth column) and $\gamma = 15$ (fifth column).

to a count of ρ^D interpolated samples. Hence, the requirements in term of time and memory needed for storing the samples and computing the DFT increase *exponentially* with the number of dimensions D. Due to these circumstances, in all tests that follow we limited our investigation to $D = 5$. We will discuss this limitation in Section IV.

Figures 3 and 4 show the $error$ between the original fitness function and the surrogate models created by surF, considering the Ackley, Alpine and Rastrigin benchmark functions with $D = 2$ and $D = 5$ dimensions, respectively. These results show that the number γ of spectral coefficients considered for the creation of the surrogate model has a great impact on the error, especially when $\sigma = 500$ or $\sigma = 5000$. This is not unexpected, as the number γ must be high enough so that the position of the global optimum is preserved in the surrogate landscape. In addition, also the number σ of samples affects the $error$ value, which decreases from $\sigma = 50$ to $\sigma = 500$. On the contrary, increasing the number of samples to $\sigma = 5000$ allows to decrease the $error$ value only in a limited number of cases (see, e.g., the Rastrigin function with $\gamma = 15$ in Figure 3). Analysing these results from a different perspective, we observe that in the case of $\sigma = 500$ and $\sigma = 5000$, all sampling strategies allow to achieve similar $error$ values; conversely, a limited number of samples (i.e., $\sigma = 50$) may result in markedly different $error$ values, as in the case of the Ackley function with $\gamma = 15$ (see Figure 3, bottom left). Finally, by comparing the performance of the different sampling strategies over all benchmark functions with different hyper-parameters combinations, it appears that QUASI and PPACK methods generally obtain the best results. However, as we will discuss in Section IV, the application of the PPACK method may be problematic.

As a final test, we calculated the surrogate surface for a "real world" problem, i.e., the Parameter Estimation (PE) of a biochemical systems [29], in order to show a practical application of surF. The PE problem consists in finding the vector $\mathbf{x} \in \mathbb{R}^+$ of the (unknown) kinetic parameters of a mathematical model describing a biochemical system, which allows to reproduce its dynamic behavior. This problem can be re-stated as an optimization problem, where the distance between a target dynamics and the simulated dynamics has to be minimized. Here we consider a simplified version of the problem, in which we aim at minimizing the difference between the number of molecules appearing at any time point:

$$f_{bio}(\mathbf{x}) = |T(t) - S(\mathbf{x}, t)|, \tag{5}$$

where $T(t)$ and $S(\mathbf{x}, t)$ represent, respectively, the target number of molecules at time t, and the simulated number of molecules at time t obtained using the putative parameterization $\mathbf{x}$.

Due to the non-linear and non-convex nature of this problem, the PE is a particularly challenging task to solve; this issue can be exacerbated if the biochemical system is analyzed by means of stochastic approaches, which are suitable when the molecular amounts are low and the role played by noise cannot be neglected [30]. Several algorithms for the stochastic simulation of biochemical models exist [30]–[32]. In this work, we leverage Gillespie's Stochastic Simulation Algorithm (SSA); specifically, the direct method implemented in the StochPy library [33]. Due to stochasticity, two SSA simulations run using the same model parameterization can yield different dynamics. As a result, the fitness function in Equation 5 is noisy, i.e., it can yield different values for the same parameterization **x**, possibly misleading the optimization algorithm. In this context, surF can be helpful to mitigate the noise, yet preserving the general structure of the fitness landscape while simplifying the optimization problem.

To investigate the effectiveness of this approach, we exploit a simple stochastic model describing the transcription of a gene [33], a process that is well-known to be affected by stochastic fluctuations [34]. Transcription is a cellular process in which the information encoded in a gene is copied into a messenger RNA by an enzyme called RNA polymerase. This process can be formalized with the following reactions:

- polymerase $\xrightarrow{k_{ini}}$ polymeraseMoving;
- polymeraseMoving $\xrightarrow{k_{tra}}$ mRNA + polymerase;
- mRNA $\xrightarrow{k_{deg}} \lambda$,

where polymerase and polymeraseMoving denote the inactive and active RNA polymerase, respectively, mRNA denotes the messenger RNA, and λ denotes the degradation of the reactant. A stochastic parameter is associated with each reaction: k_{ini} denotes the rate of activation of the RNA polymerase, k_{tra} denotes the rate of transcription of the gene, and k_{deg} denotes the rate of degradation of the mRNA. Stochastic parameters are exploited by SSA to assess the probability of each reaction to occur [30].

In this test, the goal of the PE was to fit an amount of mRNA equal to 80 molecules at $t = 500$ minutes. The value of k_{tra} was fixed to 0.1 min^{-1}, while k_{ini} and k_{deg} were unknown. It is worth noting that, despite the small number of parameters, their estimation can be a very complex task, because the fitness landscape is characterized by noise and multi-modality. We show in Figure 5 a comparison of the original fitness landscape (blue surface) and the surrogate model created by surF (orange surface). In this test, we used the QUASI initialization method, $\gamma = 8$, $\sigma = 500$, $\rho = 100$.

Thanks to the application of surF, the plethora of local minima characterizing the original fitness landscape is removed in the surrogate model, but its general features are preserved. For instance, in both cases the fitness value increases when the degradation constant k_{deg} decreases, since this causes the accumulation of mRNA molecules. It is also important to highlight that the fitness evaluations performed with the surrogate model were computationally less expensive than the SSA runs necessary to compute the corresponding fitness values of the original model. The generation of the surface related to the original fitness landscape in Figure 5 required 156 seconds, while the surface of the surrogate fitness landscape required 0.08 seconds, corresponding to approximately a 2000× speedup.

As a final remark, it is worth noting that, when using a few

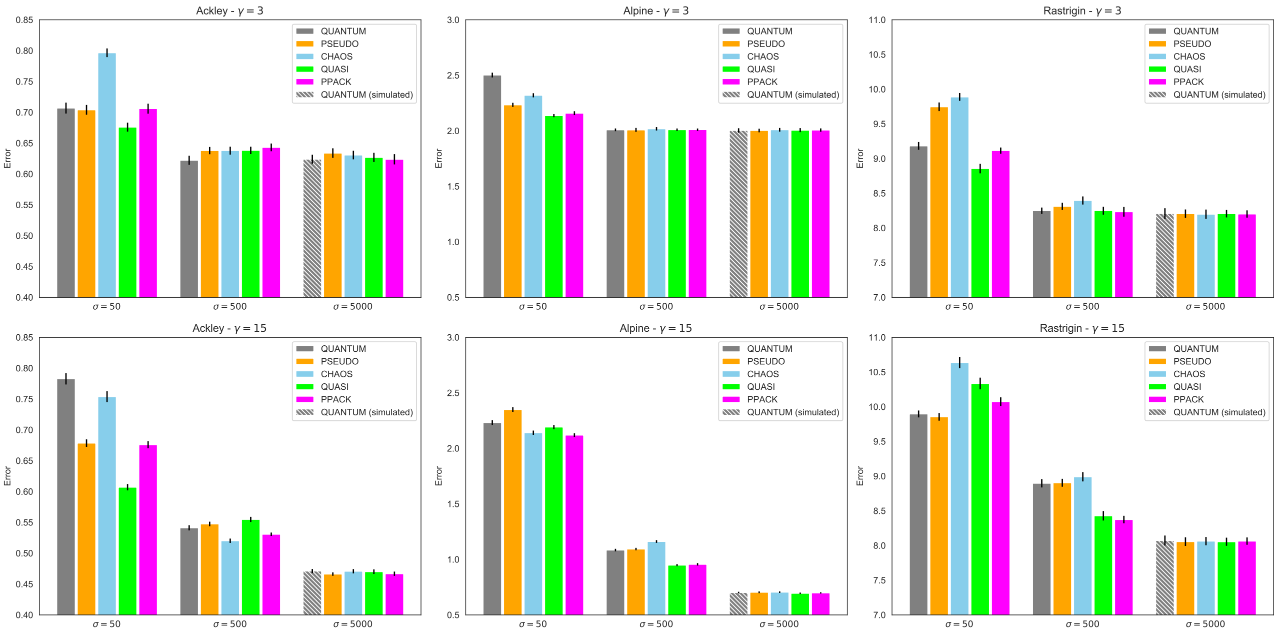

Fig. 3. Error between the original fitness function and the surrogate models created by surF. The first, second and third columns correspond to the 2D versions of the Ackley, Alpine and Rastrigin functions, respectively. The upper and lower rows correspond to the cases $\gamma = 3$ and $\gamma = 15$, respectively. In each figure, the histograms are divided in three groups corresponding (left to right) to the number of initial samples used: $\sigma = 50$, $\sigma = 500$, and $\sigma = 5000$.

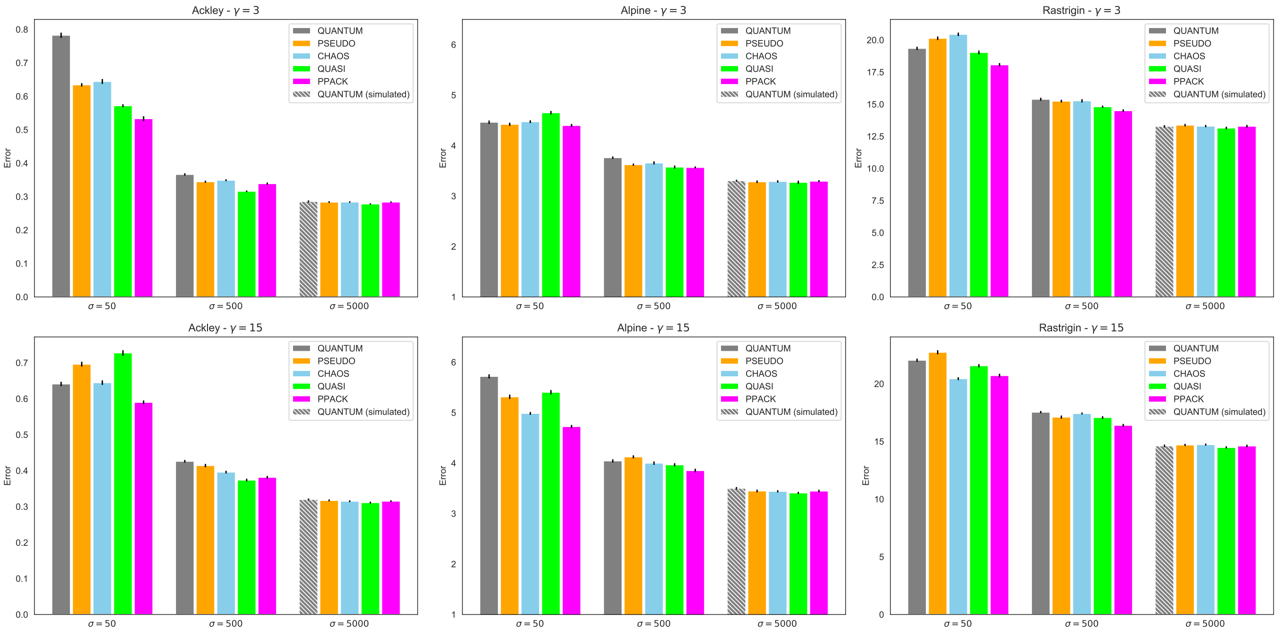

Fig. 4. Error between the original fitness function and the surrogate models created by surF. The first, second and third columns correspond to the 5D versions of the Ackley, Alpine and Rastrigin functions, respectively. The upper and lower rows correspond to the cases $\gamma = 3$ and $\gamma = 15$, respectively. In each figure, the histograms are divided in three groups corresponding (left to right) to the number of initial samples used: $\sigma = 50$, $\sigma = 500$, and $\sigma = 5000$.

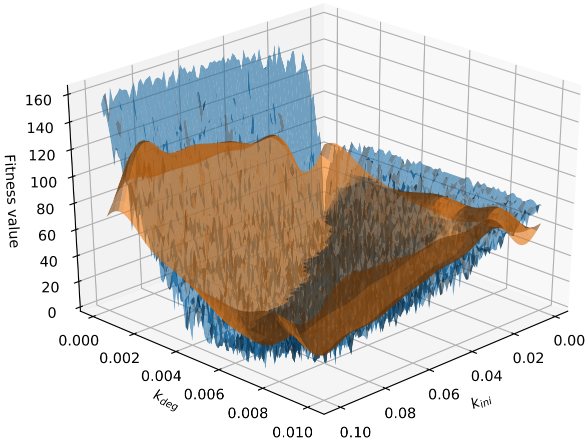

Fig. 5. Comparison of the fitness landscape of the PE problem of a stochastic biochemical model. The blue and orange surfaces represent the original fitness landscape and the surF surrogate model.

samples, surF can be more effective than alternative surrogate modeling methods. For example, Figure 6 shows four Kriging surrogate models of the Ackley function, created with SMT [35] (using default settings), and using $\sigma = 50$ pseudo-random samples. Compared to the surrogate model produced by surF (see Figure 2, second row), the Kriging surrogates appear to be more approximate and noisy. The average errors calculated for the four examples in Figure 6 are 1.04, 1.5, 1.2 and 1.3 (from top to bottom, left to right), which are always higher than the $error$ values obtained with surF (see Figure 3, first column).

IV. Conclusion

In this paper we investigated the impact of different sampling methods and of the hyper-parameters of surF on the generation of surrogate models of fitness landscapes. surF exploits the Fourier transform to perform a low-pass filtering of the fitness landscape, producing a smooth surrogate model characterized by a "tunable" level of ruggedness. Notably, the surrogate model is also computationally less expensive than the original fitness function. According to our results, the sampling of the fitness landscape performed with purely random approaches (e.g., QUANTUM, PSEUDO, CHAOS) is outperformed by more "structured" methods (e.g., QUASI, PPACK), which grant a more uniform coverage of the search space, even with a reduced number of samples. Interestingly, the best sampling algorithm seems to be problem-dependent in the case of low values of σ, while the differences are limited if σ is higher.

The main drawback of the surF methodology concerns the interpolation of the random samples of the fitness function using a D-dimensional lattice. This phase—which is necessary to evaluate the Fourier transforms at regular intervals—is characterized by a very high time and space complexity: if ρ partitions are considered for each dimension of the search space, then a grid of ρ^D interpolated points must be calculated. Hence, the first implementation proposed in [19] should be enhanced to solve optimization problems with a high number of dimensions. We will investigate new approaches for the effective assessment of Fourier transforms over high-dimensional fitness landscapes—as in the case of real-world problems—using a less-than-exponential complexity algorithm in which the ρ hyper-parameter is no longer necessary.

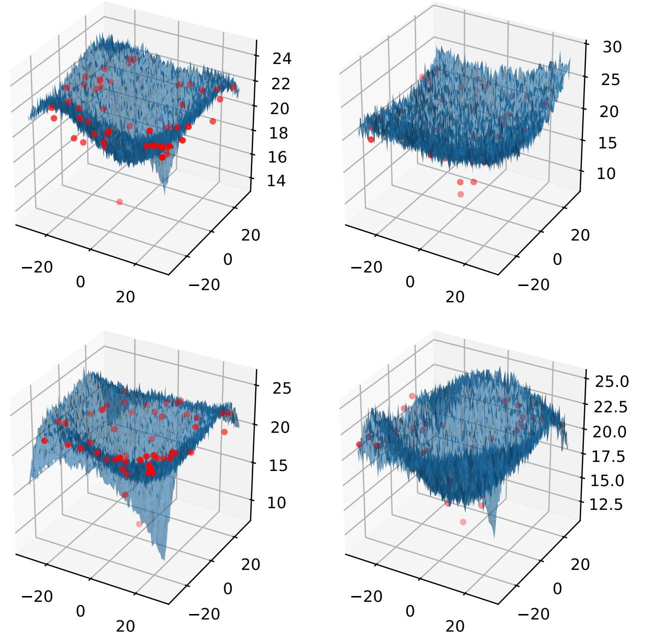

Fig. 6. Surrogate models of the Ackley function created with the Kriging method. The red dots denote the 50 random samples used for the generation of each surrogate model.

A second issue highlighted in this work regards the huge effort spent by the QUANTUM and PPACK methodologies in generating the random samples:

- in the case of QUANTUM, by exploiting multiple qubits quantum machines (e.g., the ibmq_essex backend that we leveraged in part of our tests), several measurements can be performed during the same experiment (e.g., "in parallel"). However, due to the scarcity of quantum computers in the planet, we frequently ended up queuing and waiting minutes for a single experiment to be scheduled. Such experiments generate just a subset of the 64 bits that are necessary to build a double precision floating point random number. Hence, paradoxically, the execution of single-qubit experiments on the ibmq_armonk backend resulted more efficient, since there was almost no queue on this specific machine. Due to this circumstances, and the technical delay introduced by the communication with backends, the use of QUANTUM initialization for surF resulted extremely time consuming. We expect that, with the advancement of the technology (i.e., more quantum computers, equipped with more qubits), this methodology will eventually become feasible for surF;

- in the case of PPACK, the existing algorithms are designed to place as many points as possible while preserving the minimum distance between any pair of points. This is actually the opposite of what is necessary to run surF, i.e., the optimal packing of exactly σ points in a D-dimensional search space. Nevertheless, the correct placement can be achieved by tuning, by means of a trial-and-error approach, the minimum distance parameter until the number of points reaches the desired value. In order to make surF completely automatic, we will investigate a novel variant of the point packing algorithm, able to estimate the minimum distance to solve the inverse problem of optimally packing σ points.

To conclude, the current default setting for surF is the QUASI sampling strategy, which provides an excellent trade-off between efficiency and surrogate model error.

Acknowledgment

We acknowledge the use of the IBM Q Experience. Part of this work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

References

[1] A. Bhosekar and M. Ierapetritou, "Advances in surrogate based modeling, feasibility analysis, and optimization: A review," *Computers & Chemical Engineering*, vol. 108, pp. 250–267, 2018.

[2] T. Lew, A. Spencer, F. Scarpa, K. Worden, A. Rutherford, and F. Hemez, "Identification of response surface models using genetic programming," *Mechanical Systems and Signal Processing*, vol. 20, no. 8, pp. 1819–1831, 2006.

[3] J. Eason and S. Cremaschi, "Adaptive sequential sampling for surrogate model generation with artificial neural networks," *Computers & Chemical Engineering*, vol. 68, pp. 220–232, 2014.

[4] G. E. Box and N. R. Draper, *Empirical Model-Building and Response Surfaces.* John Wiley & Sons, 1987.

[5] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical Science*, pp. 409–423, 1989.

[6] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[7] Z. Wang and M. Ierapetritou, "A novel feasibility analysis method for black-box processes using a radial basis function adaptive sampling approach," *AIChE Journal*, vol. 63, no. 2, pp. 532–550, 2017.

[8] A. Samad, K.-Y. Kim, T. Goel, R. T. Haftka, and W. Shyy, "Multiple surrogate modeling for axial compressor blade shape optimization," *Journal of Propulsion and Power*, vol. 24, no. 2, pp. 301–310, 2008.

[9] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.

[10] F. A. Viana, R. T. Haftka, and L. T. Watson, "Efficient global optimization algorithm assisted by multiple surrogate techniques," *Journal of Global Optimization*, vol. 56, no. 2, pp. 669–689, 2013.

[11] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 1, pp. 66–76, 2006.

[12] A. I. Forrester and A. J. Keane, "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, vol. 45, no. 1-3, pp. 50–79, 2009.

[13] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[14] B. Liu, Q. Zhang, and G. G. Gielen, "A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 180–192, 2013.

[15] Y. Yang, W. Zeng, W.-s. Qiu, and T. Wang, "Optimization of the suspension parameters of a rail vehicle based on a virtual prototype Kriging surrogate model," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 230, no. 8, pp. 1890–1898, 2016.

[16] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[17] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 644–660, 2017.

[18] Y. Tang, J. Chen, and J. Wei, "A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions," *Engineering Optimization*, vol. 45, no. 5, pp. 557–576, 2013.

[19] L. Manzoni, D. M. Papetti, P. Cazzaniga, S. Spolaor, G. Mauri, D. Besozzi, and M. S. Nobile, "Surfing on fitness landscapes: A boost on optimization by Fourier surrogate modeling," *Entropy*, vol. 22, no. 3, 2020.

[20] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[21] M. S. Nobile and H. Iba, "A double swarm methodology for parameter estimation in oscillating gene regulatory networks," in *2015 IEEE Congress on Evolutionary Computation (CEC).* IEEE, 2015, pp. 2376–2383.

[22] M. Matsumoto and T. Nishimura, "Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

[23] H. Abraham, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, and G. A. *et al.*, "Qiskit: An open-source framework for quantum computing," 2019.

[24] S. Phatak and S. S. Rao, "Logistic map: A possible random-number generator," *Physical Review E*, vol. 51, no. 4, p. 3670, 1995.

[25] I. M. Sobol', "On the distribution of points in a cube and the approximate evaluation of integrals," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 7, no. 4, pp. 784–802, 1967.

[26] H. Niederreiter, "Low-discrepancy and low-dispersion sequences," *Journal of Number Theory*, vol. 30, no. 1, pp. 51–70, 1988.

[27] R. Brown and D. Ashlock, "Parameter tuning of a peak fitting algorithm with an evolved experimental design," in *Proceedings of the 2019 IEEE Congress on Evolutionary Computation*, 2019, pp. 2379–2386.

[28] D. Ashlock, S. K. Houghten, J. A. Brown, and J. Orth, "On the synthesis of DNA error correcting codes," *Biosystems*, vol. 110, no. 1, pp. 1–8, 2012.

[29] A. Tangherloni, S. Spolaor, P. Cazzaniga, D. Besozzi, L. Rundo, G. Mauri, and M. S. Nobile, "Biochemical parameter estimation vs. benchmark functions: A comparative study of optimization performance and representation design," *Applied Soft Computing*, vol. 81, p. 105494, 2019.

[30] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[31] ——, "The chemical Langevin equation," *The Journal of Chemical Physics*, vol. 113, no. 1, pp. 297–306, 2000.

[32] M. S. Nobile, P. Cazzaniga, D. Besozzi, D. Pescini, and G. Mauri, "cuTauLeaping: A GPU-powered tau-leaping stochastic simulator for massive parallel analyses of biological systems," *PLOS One*, vol. 9, no. 3, 2014.

[33] T. R. Maarleveld, B. G. Olivier, and F. J. Bruggeman, "StochPy: a comprehensive, user-friendly tool for simulating stochastic biological processes," *PLOS One*, vol. 8, no. 11, 2013.

[34] A. Eldar and M. B. Elowitz, "Functional roles for noise in genetic circuits," *Nature*, vol. 467, no. 7312, pp. 167–173, 2010.

[35] M. A. Bouhlel, J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J. R. R. A. Martins, "A Python surrogate modeling framework with derivatives," *Advances in Engineering Software*, p. 102662, 2019.