Examining Algorithm Behavior using Recurrence Quantification and Landscape Analyses

Mario Andrés Muñoz
munoz.m@unimelb.edu.au
School of Computer and Information Systems
Parkville, VIC, Australia

ABSTRACT

Differences in performance between algorithms can be attributed to the interaction between their unique rule-sets and the characteristics of the instance's landscape. However, understanding this interaction can be difficult because algorithms are often composed of multiple elements, and in the worst cases are described using opaque notation and metaphors. In this paper, we introduce a methodology for the behavioral analysis of optimization algorithms, based on comparing algorithm dynamics in a given problem instance. At the methodology's core lays the hypothesis that if two algorithms, with the exact same initial conditions, have similar dynamics, then their rule-sets are also similar. An examination of Grey Wolf Optimization, shows that it exhibits bias leading to similar behavioral patterns regardless of the function.

CCS CONCEPTS

• **General and reference → Empirical studies**; **Experimentation**; • **Theory of computation → Bio-inspired optimization**; • **Computing methodologies** → *Randomized search.*

KEYWORDS

Black-box optimization, Dynamical Systems, Landscape Analysis, Instance Spaces, Recurrence quantification analysis

ACM Reference Format:
Mario Andrés Muñoz. 2022. Examining Algorithm Behavior using Recurrence Quantification and Landscape Analyses. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3520304.3534029

1 INTRODUCTION

An optimization algorithm uses a rule-set to iteratively identify better solutions for a problem instance. Consequently, algorithms can out-perform in some instances and under-perform in others, depending on the interaction between the rule-set and the characteristics of the instance's landscape. This bias can be difficult to understand, because algorithms are often composed of multiple interconnected pieces and, in the worst cases, are described using opaque notation and metaphors [16]. Therefore, experimental benchmarking is an essential part of the analysis of an optimization algorithm, as its aims are, among others, to provide an assessment of an algorithm's performance, and illustrate the search behavior [1].

Research into the control of their behavior has revealed that Evolutionary Algorithms (EA), due to the interactions between individual, create structures that resemble complex networks, which go through transitions between stochastic and deterministic chaotic regimes [22]. Moreover, EAs are highly non-linear, discrete dynamical systems, whose state is given by the location of each individual at a given iteration; hence, it is fully observable and recordable [22]. Therefore, techniques for non-linear dynamical systems' analysis could be useful for examining and comparing algorithm behavior in a given problem instance.

Based on these concepts, here we introduce a methodology for the behavioral analysis of optimization algorithms. We hypothesize that if two algorithms, acting on the same problem instance, starting from the same location and with the same random seed, have similar dynamics, then their rule-sets are also similar. To capture the dynamics into a set of features, we make use of Recurrence Quantification Analysis (RQA), a technique from chaotic systems recently employed by Vantuch et al. [17, 18] in the analysis of EAs. Using concepts derived from Instance Space Analysis (ISA) [11, 15], we construct a visualization that represents an algorithm/instance system as a point in a two-dimensional space. We test three algorithms, Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO) and Simple Genetic Algorithm (SGA). We combine these results with Landscape Analysis (LA) features to identify when and why our hypothesis is true. Our results show that, although PSO and GWO, with default parameters, do not show statistically significant similarities in behavior, GWO appears to exhibit systemic bias, leading to similar behavioral patterns regardless of the function, as long as this does not have ill-conditioning and multiple funnels.

The paper continues as follows: Section 2 presents the background of this work, focusing on RQA. Then, Section 3 presents our methodology and experimental settings. Section 4 presents and discusses our results, and Section 5 concludes our paper with some limitations and ways forward on this research.

2 BACKGROUND

2.1 Algorithm state

Let us define an algorithm state. Assuming minimization, an unconstrained black-box continuous optimization problem is a function f, defining a map between the *decision space*, $\mathcal{X} \subset \mathbb{R}^m$, and the *objective space*, $\mathcal{Y} \subset \mathbb{R}$, with $m \geq 1$. Let $\mathbf{x} \in \mathcal{X}$ be a *candidate* and $y \in \mathcal{Y}$ its *cost*. At each iteration i, an algorithm produces a matrix of

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9268-6/22/07…$15.00
https://doi.org/10.1145/3520304.3534029

candidates, $\mathbf{X}_i = \left[\mathbf{x}_{i,1} \ \dots \ \mathbf{x}_{i,k} \ \dots \ \mathbf{x}_{i,n}\right], \mathbf{X}_i \in \mathbb{R}^{m\times n}$, with costs, $\mathbf{y} \in \mathbb{R}^n$, where n is the number of individuals in the population. The vector $\mathbf{x}_{i,k}$ is the i-th *state* of an individual k, and the matrix $\mathbf{X}_i$ is the i-th *state* of the algorithm, which can be thought of as the result from a function g, that is:

$$\mathbf{X}_i = g\left(\mathbf{X}_{i-1}, \mathbf{y}_{i-1}\right)$$
$$\mathbf{y}_i = f\left(\mathbf{X}_i\right)$$

Let $\mathbf{x}_{(i)} = \left[\mathbf{x}_{i,1}^\top \ \dots \ \mathbf{x}_{i,k}^\top \ \dots \ \mathbf{x}_{i,n}^\top\right]^\top, \mathbf{x}_{(i)} \in \mathbb{R}^{mn}$ be a vector representation of the state, which is a point on the high-dimensional *phase space*. From its initial conditions, the state evolving over time traces a path in the phase space known as the *trajectory*, whose shape can elucidate qualities not obvious otherwise.

2.2 Recurrence Quantification Analysis

Recurrence Quantification Analysis (RQA) is a methodology used to characterize and detect transitions in the behavior of a dynamical system [9], and used by Vantuch et al. [17, 18] to demonstrate that swarm algorithms exhibit transitions between convergent (orderly) and non-convergent (chaotic) behavior, with these "phase-like" transitions being statistically different and dependent on the algorithm itself. Conceptually, RQA is based on two observations of dynamical systems: (a) similar states often evolve similarly; and (b) some states occur over and over again. Although RQA is based on principles from deterministic dynamics, the method makes no assumptions about the underlying system [10]. In the following sections, we will describe the basic concepts behind RQA.

2.2.1 Recurrence Plots (RP).

2.2.1 Recurrence Plots (RP). Introduced by Eckmann et al. [5], an RP is a visualization used to examine the recurrences of a trajectory in the phase space. In essence, it is a representation of a binary distance matrix between state vectors, $\mathbf{R}$, with its axes corresponding to time increasing from left to right and bottom to top. The elements of $\mathbf{R}$ are given by the equation:

$$\mathbf{R}_{i,j}(\varepsilon) = \mathbf{1}\left(\varepsilon - \left\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\right\|_2 \leq 0\right) \tag{1}$$

where $\mathbf{x}_{(i)}$ are the observed states of the system at times $\{i, j\} = 1, \dots, T$, ε is a threshold, $\mathbf{1}(\cdot)$ denotes the indicator function, and $\|\cdot\|_2$ denotes the Euclidean distance. The most important parameter is ε, which determines if two states are neighbors or not. If the value of ε is too small, almost no recurrence will be observed, limiting our ability to learn about the structure of the underlying system. On the other hand, if it is too large, almost every state will be considered a neighbor of another, leading to plots with significant artifacts [9]. A common rule of thumb is to use a value of ε equal to a percentage of the maximum phase space diameter. Figure 1 presents a distance matrix and the resulting RPs for a system composed of a 20-individual GWO algorithm exploring the two-dimensional Katsuura (f_{23}) function from COCO [6]. Five runs of 50 generations were started from different, randomly selected points, whereas the values of ε are equal to $\{5\%, 15\%, 25\%\}$ of the phase space diameter.

In general, an RP such as those shown in Figure 1 can present small-scale behavioral patterns, which can be of four types:

Isolated dots represent rare states that persist for a short time.

Diagonal lines represent an evolution process between states that is similar at different times. For example, for two starting times i and j, $\mathbf{x}_i \approx \mathbf{x}_j, \mathbf{x}_{i+1} \approx \mathbf{x}_{j+1}, \dots, \mathbf{x}_{i+\ell-1} \approx \mathbf{x}_{j+\ell-1}$, or more generally as $\mathbf{R}_{i+k,j+k} \equiv 1 \,|_{k=0}^{\ell-1}$, where ℓ is the length of the time of the interval in which the recurrence occurs. Diagonal line patterns can be formally described as:

$$\left(1 - \mathbf{R}_{i-1,j-1}\right)\left(1 - \mathbf{R}_{i+\ell,j+\ell}\right)\prod_{k=0}^{\ell-1}\mathbf{R}_{i+k,j+k} \equiv 1$$

Vertical or horizontal lines represent an interval where the state changes slowly or not at all. For example, for two starting times i and j, $\mathbf{x}_i \approx \mathbf{x}_j, \mathbf{x}_i \approx \mathbf{x}_{j+1}, \dots, \mathbf{x}_i \approx \mathbf{x}_{j+v-1}$, or more generally as $\mathbf{R}_{i,j+k} \equiv 1 \,|_{k=0}^{v-1}$. Vertical line patterns can be formally described as:

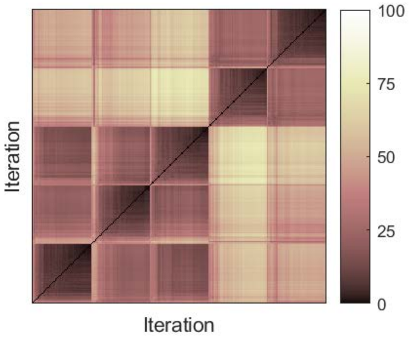
$$(1 - \mathbf{R}_{i,j-1})(1 - \mathbf{R}_{i,j+v})\prod_{k=0}^{v-1}\mathbf{R}_{i,j+k} \equiv 1$$

Bowed lines are patterns without constant slope or gradient, representing a similar evolution of the state but at different velocities, indicating changing system dynamics.

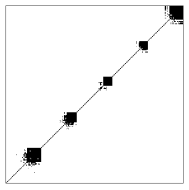
These small-scale patterns can result in large-scale ones, which can be of four other types:

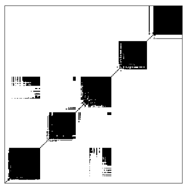
Homogeneous behavior or large recurrent states, which occur when the time taken to return to a steady state is shorter compared to the time spanned by the experiment.

Periodic and quasi-periodic behavior, represented by strong diagonal patterns.



(a)


(b) $\varepsilon = 5\%$


(c) $\varepsilon = 15\%$

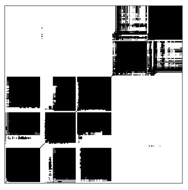

(d) $\varepsilon = 25\%$

Figure 1: State vectors represented as (a) a distance matrix, and recurrence plots for (b) $\varepsilon = 5\%$, (c) $\varepsilon = 15\%$, and (d) $\varepsilon = 25\%$ of the phase diameter, for a 20-individual GWO algorithm on the two-dimensional Katsuura (f_{23}) function from COCO [6] during five runs.

Drifting behavior, which occurs when the system has slowly varying parameters, represented by less recurrence as the states move further from the main diagonal.

Disrupted which have abrupt changes in dynamics, causing sections without recurrence.

For example, in Figure 1, the large black blocks represent homogeneous states, which indicate that the GWO algorithm quickly converges to a different local optima on each run. Moreover, these local optima form two groups, from which it is possible to escape into one of its composing optima, but not into another group. Each restart results in a disruption of the behavior. Moreover, the system does not present any periodical behavior, meaning that the individuals cannot continue exploring the problem after convergence.

2.2.2 Cross-Recurrence Plots (CRP). An extension of RP used to analyze two systems on the same phase space, CRPs aim to identify interactions [10, 20]. In essence, it is a visual representation of a binary distance matrix between the state vectors of two systems, $\mathbf{C}^{A,B}$, with axes corresponding to time. The elements of $\mathbf{C}^{A,B}$ are given by the equation:

$$\mathbf{C}_{i,j}^{A,B}(\varepsilon) = \mathbf{1}\left(\varepsilon - \left\|\mathbf{x}_{(i),A} - \mathbf{x}_{(j),B}\right\|_2 \leq 0\right)$$

where $\mathbf{x}_{(i),A}$ and $\mathbf{x}_{(j),B}$ are the observed states of systems $\{A, B\}$ at times $\{i, j\} = 1, \ldots, T$ respectively. As in Eq.(1), ε is a threshold, $\mathbf{1}(\cdot)$ denotes the indicator function, and $\|\cdot\|_2$ denotes the Euclidean distance. In a CRP, synchronization results in diagonal structures, whose distortion represents temporal dilation or compression.

2.2.3 RQA measures of complexity. To quantify the information presented in an RP and CRP, several measures of complexity have been proposed based on the point density and the diagonal and vertical line structures [9]. These are:

Recurrence rate (*RR*) is the simplest measure, corresponds to the percentage of recurrence points in an RP, i.e.,

$$RR = \frac{1}{T^2}\sum_{i,j=1}^{T} \mathbf{R}_{i,j}$$

As $T \to \infty$, *RR* corresponds to the probability that a state recurs to a ε-neighborhood in the phase space.

Determinism (*DET*) is the percentage of points forming diagonal lines in an RP, i.e.,

$$DET = \frac{\sum_{\ell=\ell_{\min}}^{T} \ell P(\ell)}{\sum_{\ell=1}^{T} \ell P(\ell)}$$

where $\ell_{\min}$ is a user-defined parameter. Processes with uncorrelated or weakly correlated, stochastic or chaotic behavior cause none or very short diagonals, while deterministic processes cause longer diagonals. Therefore, *DET* is a measure of the predictability of the system.

Divergence is the inverse of the longest diagonal line found on the RP, $\ell_{\max}$, which is related to the exponential divergence of the phase space trajectory, i.e., shorter diagonal lines represent faster divergence of the trajectory segments.

Entropy (*ENTR*) of the probability of finding a diagonal line of exactly length ℓ in the PR, i.e.,

$$ENTR = -\sum_{\ell=\ell_{min}}^{T} p(\ell)\ln p(\ell)$$

Therefore, *ENTR* captures the complexity of the diagonal line structure of the RP.

Laminarity (*LAM*) is the measure analogous to *DET* but on vertical lines, and corresponds to the percentage of points forming a vertical line on an RP, i.e.,

$$LAM = \frac{\sum_{v=v_{\min}}^{T} vP(v)}{\sum_{v=1}^{T} vP(v)}$$

where $v_{\min}$ is a user-defined parameter, and often equal to $\ell_{\min}$. This measure represents the occurrence of laminar states in the system, and will decrease if the RP consist of more single points.

Trapping time (*TT*) corresponds to the average length of a vertical line structure, i.e.,

$$TT = \frac{\sum_{v=v_{\min}}^{T} vP(v)}{\sum_{v=v_{\min}}^{T} P(v)}$$

That is, *TT* identifies how long a system will be trapped in a given state.

These empirical measures, although helpful for finding various transitions in dynamical systems, have as a drawback is their lack of invariance with parameters, such as ε [9].

2.3 Landscape Analysis

Landscape Analysis (LA) are data-based methods for measuring characteristics of a problem, i.e., modality, smoothness, global structure, variable scaling and separability [13], which generate one or more *features* describing particular characteristics. LA has been successfully used for identifying strengths and weaknesses of algorithms [8], automatic algorithm selection [7], and per instance algorithm configuration [2] methods, and benchmark construction techniques [12]. For this work, we employed 31 LA features, which we used on previous work [11].

2.4 Algorithms

We focus our analysis on Particle Swarm Optimization (PSO), Grey Wolf Algorithm (GWO) and Simple Genetic Algorithm (SGA), with the PSO and GWO sharing conceptual similarities worthy of analysis, and SGA serving as a conceptually different control. Swarm algorithms are a natural fit for RQA because they model individuals moving across the landscape, drawing a trajectory. For SGA, we assume that no offspring is created, but instead each individual has changed its state [22], allowing our methodology to be applicable to other population-based algorithms.

PSO models the social interactions between bird flocks or fish schools. In its canonical version [14], each individual $i = 1, \ldots, n$ is represented by a position, $\mathbf{x}_{i,k}$, and a velocity, $\mathbf{v}_{i,k}$, vectors of size m. Both vectors are initialized at random, and updated at each iteration using the equations:

$$\mathbf{v}_{i+1,k} = \omega\mathbf{v}_{i,k} + \gamma_1\mathbf{r}_1 \odot \left(\mathbf{p}_{i,k} - \mathbf{x}_{i,k}\right) + \gamma_2\mathbf{r}_2 \odot \left(\mathbf{l}_{i,k} - \mathbf{x}_{i,k}\right)$$
$$\mathbf{x}_{i+1,k} = \mathbf{x}_{i,k} + \mathbf{v}_{i+1,k}$$

where $\{\mathbf{r}_1, \mathbf{r}_2\}$ are vectors of uniformly distributed random numbers between $[0, 1]$; $\{\gamma_1, \gamma_2\}$ are acceleration coefficients; ω is the inertia weight; $\left\{\mathbf{p}_{i,k}, \mathbf{l}_{i,k}\right\}$ are the personal best and local (for a group of neighboring particles) best positions at the current iteration; and $\odot$ indicates the point-wise product between vectors.

GWO models the hierarchy and hunting behaviors of a wolf pack. Camacho-Villalon et al. [4] considers GWO to be a PSO algorithm in which three best particles are used to bias the movement of the others, as follows:

$$\begin{aligned}
\mathbf{s}_{i,1} &= \mathbf{g}_{i,1} - (2\gamma_i - 1)\,\mathbf{r}_{i,1}\left|2\mathbf{q}_{i,1} \odot \mathbf{g}_{i,1} - \mathbf{x}_{i,k}\right| \\
\mathbf{s}_{i,2} &= \mathbf{g}_{i,2} - (2\gamma_i - 1)\,\mathbf{r}_{i,2}\left|2\mathbf{q}_{i,2} \odot \mathbf{g}_{i,2} - \mathbf{x}_{i,k}\right| \\
\mathbf{s}_{i,3} &= \mathbf{g}_{i,3} - (2\gamma_i - 1)\,\mathbf{r}_{i,3}\left|2\mathbf{q}_{i,3} \odot \mathbf{g}_{i,3} - \mathbf{x}_{i,k}\right| \\
\mathbf{x}_{i+1,k} &= \left(\mathbf{s}_{i,1} + \mathbf{s}_{i,2} + \mathbf{s}_{i,3}\right)/3
\end{aligned}$$

where $\left\{\mathbf{g}_{i,1}, \mathbf{g}_{i,2}, \mathbf{g}_{i,3}\right\}$ are the position of the three best particles at the current iteration; $\left\{\mathbf{r}_{i,j}, \mathbf{q}_{i,j}\right\}, j = 1 \ldots, 3$ are vectors of uniformly distributed random numbers between $[0, 1]$; and γ_j is a decreasing acceleration coefficient that goes from 2 to 0.

SGA is an implementation of a genetic algorithm, using two-individual tournament selection, exponential crossover and polynomial mutation. During selection, two individuals are drawn at random, and the one with the lowest fitness is selected. At crossover, a random position is selected in the parent chromosome and then inserts, on each successive gene, the partner values with a user-given probability. Finally, during mutation, a sample centered around each gene is taken from the *polynomial* distribution $Pr(\delta) = 0.5\,(n + 1)\,(1 - |\delta|)^n$, with δ being drawn uniformly from $(-1, 1)$.

3 METHODOLOGY

Our methodology aims to identify the conditions in which two algorithms behave similarly, as evidence that their rule-sets are similar in practice. That is, the algorithms are not 'complementary,' a condition necessary for the construction of well-performing algorithm portfolios [7]. At the core of our methodology are four hypotheses: For different initial conditions, if the same algorithm independently runs over the same problem instance twice, then the dynamics of the system over the two runs should have similarities. Similarly, assuming the same initial conditions, if the same algorithm runs over two different instances, two different algorithms run over the same instance, or two different algorithms run over two different instances, then the dynamics of the system should have differences. We distinguish between shared and specific initial conditions, where the former refers to those comparable between algorithms, i.e., the starting point and random seed, and the latter refers to those which are incomparable, i.e., the acceleration coefficients in PSO and GWO, and other algorithm-specific parameters. The methodology has four steps:

Collecting the data includes selecting both the benchmark instances and the algorithms under study. Then, running the algorithms for an equal number of function evaluations with the same shared initial conditions.

Extracting the RQA measures of complexity from (a) the RP resulting from comparing the state of the algorithm, $\mathbf{x}_{(i)}$, and (b) the CRPs resulting from comparing the state of each individual, $\mathbf{x}_{i,k}$, against all others, which for n individuals results in $n\,(n - 1)/2$ CRPs. As we are agnostic on the settings for ε and $\ell_{\min}$, the two user-defined parameters from RQA, we define a grid with $\varepsilon = \{5\%, 10\% \ldots, 25\%, 30\%\}$ of the phase space diagonal, and $\ell_{\min} = \left\{2^1, \ldots, 2^4\right\}$, resulting in 24 combinations. With six RQA measures per combination, we have 144 measures resulting from the RP and each CRP. To summarize the results from the individuals' comparison, we take the average measures across all CRPs. In total, we obtain 288 features. Simultaneously, we extract the LA features from a suitable dataset, this often being an independent experiment using Latin Hypercube or a similar sampling method.

Visualization is achieved by projecting the RQA measures into a two-dimensional *Behavioral Space*. For this purpose, we use Principal Component Analysis (PCA) to reduce from 288 features into K components which capture over 95% of the variance. Finally, these K components are further reduced into a two-dimensional space, $\mathbf{Z}$, using t-Stochastic Network Embedding (t-SNE).

Statistical analysis helps us determine whether the behavior of the algorithms are similar or not, by testing the hypotheses:

(1) For two independent sets of runs of an algorithm on an instance, we use multivariate one-way analysis of variance (MANOVA1) to determine whether the means are co-located in $\mathbf{Z}$, meaning that the behavior is not significantly different. Then, we split the instances in two groups, those with co-located means and those without. We use a two-sample t-test (t-test2) to check whether the means of the LA features from these two groups are equal or not. If a feature has means that are not equal, then we take this as evidence that this feature affects the consistency between runs of an algorithm.

(2) For two sets of runs of an algorithm on two instances, we use MANOVA1 to determine for which combinations of instances the means are co-located in $\mathbf{Z}$. If they are, then we use t-test2 to check whether the means of LA features from these two instances are equal, in which case we take this as evidence that instances with this feature result in similar behaviors from the algorithm. As we are interested on the majority trend, if a feature has more than 60% of its tests with the null hypothesis rejected, then we consider that this feature to affect the behavior of an algorithm.

(3) For independent sets of runs of two algorithms on the same instance, we use MANOVA1 to determine for which instances the means are co-located in $\mathbf{Z}$, meaning that both algorithms behave similarly. Then, we split the instances into two groups, those with co-located means and those without. We use t-test2 to check whether the means of the LA features from these two groups are equal or not. If a feature has means that are not equal, then we take this as evidence that the behavior is similar between the algorithms for problems with this feature.

(4) For independent sets of runs of two algorithms on two instances, we use MANOVA1 to determine for which combinations of algorithms and instances the means are co-located in **Z**, meaning that both algorithms behave similarly. If they are, then we use t-test2 to check whether the means of LA features from these two instances are equal, in which case we take this as evidence that instances with this feature result in similar behaviors from the algorithms. As we are interested on the majority trend, if a feature has more than 60% of its tests with the null hypothesis rejected, then we consider that this feature to affect the behavior of an algorithm.

3.1 Experimental settings

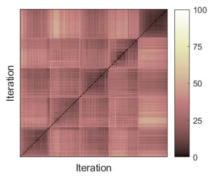
As benchmark instances, we use the 2010 BBOB scenario from COCO [6] at $m = 2$, i.e., the first fifteen instances from each of the 24 functions, resulting in 360 instances for analysis. As algorithms, we use the PSO, GWO and SGA implementations from pygmo, the Python interface of the pagmo2 library [3], with default parameters. The environment is Anaconda Python 3.7.6, with pygmo 2.16.0 and COCO 2.4. Each algorithm is run ten times with a population of 20 individuals for 50 generations, resulting in 10^3 function evaluations per run, or 10^4 evaluations per algorithm/instance combination. Each run has a random seed taken from a set, which in turn was generated by setting the random seed to 10, and then drawing ten integers from a uniform distribution between $[1, 1000]$. At each iteration i, we collect the state, $\mathbf{x}_{(i)}$, and cost, $\mathbf{y}_i$, vectors.

To extract the RQA measures, we reorganize the state vector data into three matrices, with rows corresponding to generations. Two of them containing either the first or the last five runs, $\left\{\mathbf{X}_{(A)}, \mathbf{X}_{(B)}\right\}$, and one containing all the ten runs, $\mathbf{X}_{(C)}$. The former two are used to test Hypothesis 1, whereas the latter one is used for testing Hypotheses 2 to 4. Using Yang's implementation of the RQA measures [19], we estimate $\{RR, DET, \ell_{max}, ENTR, LAM, TT\}$ for the RP and CRPs as described above, for each of $\left\{\mathbf{X}_{(A)}, \mathbf{X}_{(B)}, \mathbf{X}_{(C)}\right\}$, resulting in three datasets, all of them with 288 features for 360 instances. Simultaneously, we extract the 31 LA features by generating a sample, $\mathbf{X}_{\text{LHS}}$, of size 2×10^3 using Latin hypercube sampling. Then, we evaluate $\mathbf{X}_{\text{LHS}}$ on each instance from the COCO benchmark to obtain a sample, $\mathbf{Y}_{\text{LHS}}$. By sharing $\mathbf{X}_{\text{LHS}}$ across instances, the differences observed in the features are not due to sampling, and the computational cost is reduced. Visualization and statistical analysis are carried out in MATLAB r2020a Update 7 using the inbuilt libraries. The significance level α is set to 0.01, as to reduce the probability of false positives. Code and data are available at github.com/andremun/rqa_gecco.

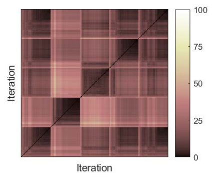
4 RESULTS

4.1 Recurrence Matrices

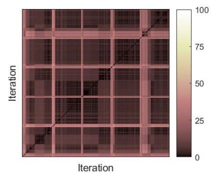
Figure 2 illustrates the distance matrix between states of the first five runs from each of the three algorithms under study, on three functions from the COCO, i.e., Ellipsoidal, f_2, Step Ellipsoidal, f_7, and Katsuura, f_{23}. On f_2, the three algorithms have very distinct behavior, i.e., PSO always converges to different states, GWO converges on some runs to the same state, and SGA quickly converges on all runs to the same state. On f_7, PSO and GWO converge to



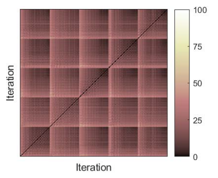
(a) PSO - f_2



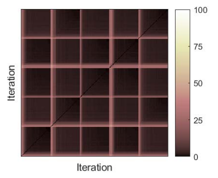
(b) GWO - f_2



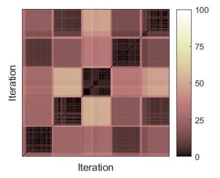
(c) SGA - f_2



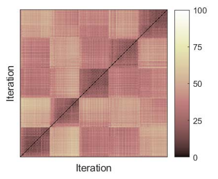
(d) PSO - f_7



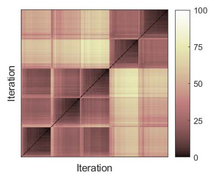
(e) GWO - f_7



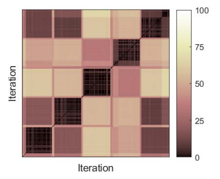
(f) SGA - f_7



(g) PSO - f_{23}



(h) GWO - f_{23}



(i) SGA - f_{23}

Figure 2: Distance matrices between states of the first five runs from PSO, GWO and SGA, on the Ellipsoidal, f_2, Step Ellipsoidal, f_7, and Katsuura, f_{23} functions from the COCO.

similar states on the five runs, as can be observed by the checkered structure of the matrix, while SGA converges to very distinct states, as the lighter areas of the matrix show. On f_{23}, both GWO and SGA show similarities, as they converge to similar states on each run, whereas PSO always converges to a different state on each run.

4.2 Behavioral Space

Figure 3 presents the *Behavioral Space*, a two-dimensional projection of the RQA measures, where each mark is an algorithm/instance combination. The color represent the algorithm. We can observe two well-separated clusters. The first corresponding to SGA, located in the top left corner, and the second corresponding to PSO and GWO, located in the bottom center. Arguably, this second cluster could be further split, with some PSO results having a behavior closer to that of GWO. However, the figure shows differences between the swarm algorithms and the control.

Figure 4 illustrates the two RQA measures with the highest correlation with the space axes, i.e., the percentage of points forming diagonal lines with a minimum length of 16, DET ($\varepsilon = 5\%$, $\ell_{\min} = 16$), and the average percentage of points forming vertical lines with a minimum length of 8 for the individuals, $\overline{LAM}$ ($\varepsilon = 5\%$, $\ell_{\min} = 8$). Broadly, the first measure separates PSO and SGA from GWO. This implies that GWO, which has larger values of this measure, tends to be a process with a more deterministic behavior, resulting in a higher *consistency*, potentially making it more predictable. On the other hand, the second measure separates PSO and GWO from SGA. This implies that the individuals of SGA, which

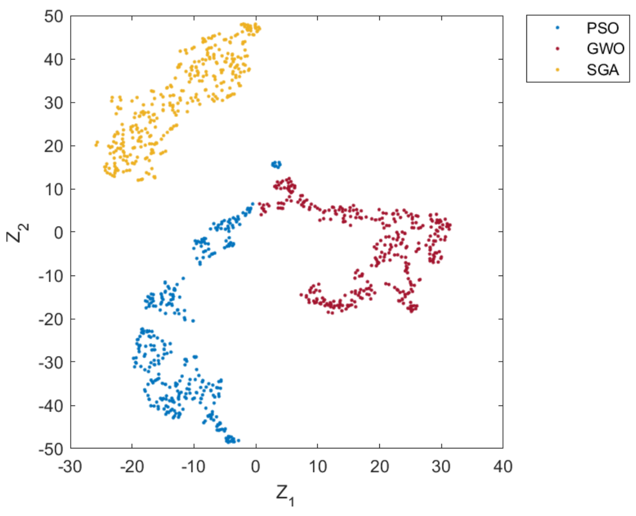
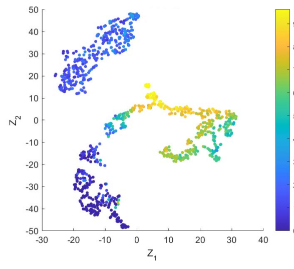
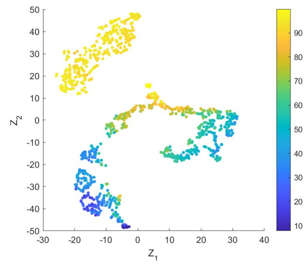


Figure 3: Behavioral Space obtained by projecting the RQA measures, extracted from the ten run data, into two dimensions using PCA and t-SNE. Color marks indicate the algorithm, i.e., PSO (●), GWO (●) and SGA (●).



(a) DET (ε = 5%, $\ell_{\min}$ = 16)



(b) $\overline{LAM}$ (ε = 5%, $\ell_{\min}$ = 8)

Figure 4: RQA measures with the highest correlation with the two-dimensional space axes.

has larger values of this measure, tend to stay at steady states more consistently than the swarm algorithms.

Figure 5 shows the COCO benchmarks divided into four groups, (a) $f_1 - f_6$, (b) $f_7 - f_{12}$, (c) $f_{13} - f_{18}$ and (c) $f_{19} - f_{24}$, giving us finer grain information on the location of each function. We focus the results from PSO and GWO around the origin coordinates, (0.0, 0.0), where most similarities appear to exist. We observe that there is no overlap on the location of functions, e.g., $\{f_5, f_9, f_{14}, f_{19}\}$ have some instances located nearby the origin. However, for f_5, the centroid is located approximately on (3.32, 15.63) for PSO (+) and (25.26, −17.47) for GWO (×); for f_9, it is located approximately on (−18.19, −24.85) for PSO (+) and (5.97, 9.89) for GWO (×); for f_{14}, it is located approximately on (−4.61, 0.16) for PSO (+) and (8.70, −5.86) for GWO (×); and for f_{19}, it is located approximately on (−16.31, −38.22) for PSO (+) and (10.45, 5.60) for GWO (×). These results indicate that it is unlikely that statistically significant differences will be observed for Hypotheses 3 and 4.

Table 1: Results for testing whether two independent sets of runs of an algorithm on the same instance are co-located (Hypothesis 1)

(a) P-values of MANOVA1. In bold are values less than 0.01.

Function	PSO	GWO	SGA	Function	PSO	GWO	SGA
f_1	0.637	0.992	**0.007**	f_{13}	0.699	0.707	**0.000**
f_2	0.635	0.219	**0.000**	f_{14}	**0.000**	0.709	0.082
f_3	0.720	0.033	**0.001**	f_{15}	0.190	0.042	**0.002**
f_4	0.077	0.827	0.020	f_{16}	0.674	0.892	0.052
f_5	0.086	**0.005**	0.143	f_{17}	0.696	0.774	**0.002**
f_6	**0.004**	0.973	**0.000**	18	0.468	0.982	**0.000**
f_7	0.950	0.210	**0.000**	f_{19}	0.564	0.054	**0.000**
f_8	0.225	0.909	**0.000**	f_{20}	**0.000**	0.042	**0.000**
f_9	0.011	**0.001**	**0.001**	21	0.562	0.924	**0.003**
f_{10}	0.967	0.116	**0.000**	f_{22}	0.557	0.991	**0.001**
f_{11}	0.447	0.964	**0.001**	f_{23}	0.678	0.225	**0.000**
f_{12}	0.814	0.328	**0.000**	f_{24}	0.030	0.159	**0.000**

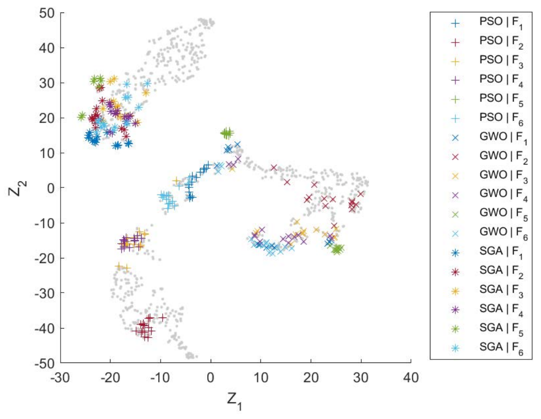
(b) P-values of t-test2. In bold are values greater than 0.01.

Feature	PSO	GWO	SGA	Feature	PSO	GWO	SGA
FDC	**0.017**	0.007	**1.000**	$\epsilon_{\max}$	0.000	0.001	**0.015**
$DISP_{1\%}$	0.000	0.004	**0.460**	M_0	0.000	0.000	**0.884**
R^2_L	0.000	0.001	**0.154**	EL_{10}	**0.409**	0.000	0.004
R^2_{LI}	0.000	0.000	**0.563**	EQ_{10}	0.000	0.000	**0.109**
R^2_{QI}	0.000	0.000	**0.589**	ET_{10}	0.000	0.000	**0.297**
R^2_Q	0.000	0.006	**0.761**	LQ_{10}	**0.503**	0.000	0.000
β_{min}	**0.400**	**0.495**	**0.616**	EL_{25}	0.000	0.000	**0.046**
β_{max}	**0.278**	**0.378**	**0.518**	EQ_{25}	0.000	**0.288**	**0.209**
CN	0.000	**0.962**	**0.796**	ET_{25}	0.000	0.000	**0.429**
$H(Y)$	**0.268**	**0.154**	0.000	LQ_{25}	**0.018**	0.000	**0.014**
$\xi^{(N)}$	0.000	0.000	0.000	EL_{50}	0.000	**0.040**	**0.464**
$\xi^{(1)}$	**0.011**	0.000	0.000	EQ_{50}	0.000	0.000	**0.683**
$\xi^{(2)}$	0.000	0.000	0.000	ET_{50}	0.000	0.000	**0.804**
$\gamma(Y)$	0.009	**0.363**	**0.102**	LQ_{50}	**0.967**	0.002	**0.245**
$\kappa(Y)$	**0.993**	**0.640**	**0.142**	PKS	0.000	0.000	**0.929**
H_{max}	0.000	0.000	**0.133**				

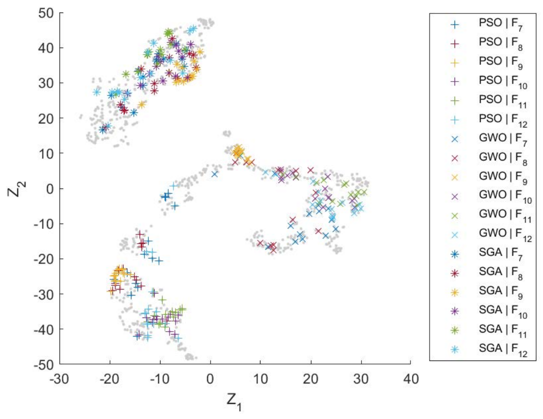
4.3 Statistical analysis

4.3.1 Hypothesis 1. Table 1a shows the p-values of MANOVA1 for testing whether two independent sets of runs of an algorithm on an instance are not significantly different. These results indicate that both PSO and GWO tend to converge similarly across runs; hence, they are more *consistent* and could be more predictable on the same instance. On the other hand, SGA is more *discrepant*, i.e., it is more dependent on the starting point and random seed. Table 1b shows the p-values for ttest2 for verifying which LA features are influential. We observe that for SGA, features derived from Information Significance are not influential, which may indicate that variable dependencies do not result in discrepancies. On the other hand, features that attempt to measure ill-conditioning, such as $\beta_{\max}$ and $H(Y)$, are influential for the consistency of both PSO and GWO.

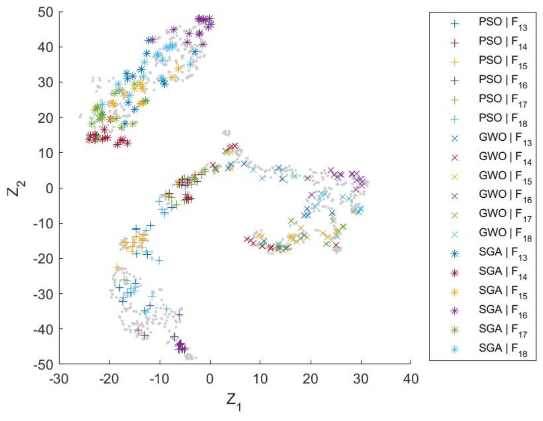
4.3.2 Hypothesis 2. Figure 6 illustrates the combination of functions on which the same algorithm produces similar convergence behavior. A color mark is used to signify that the combination produces a p-value of MANOVA1 less than 0.01. This effect is more common on GWO with 25.0% of the combinations, then SGA with 14.9% of the combinations and finally PSO with 8.0% of the combinations. These results align with our observations on Figure 4, which showed that GWO tends to behave more consistently across



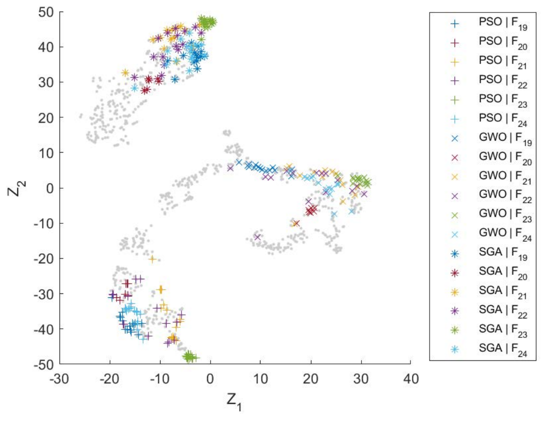
(a) $f_1 - f_6$



(b) $f_7 - f_{12}$



(c) $f_{13} - f_{18}$



(d) $f_{19} - f_{24}$

Figure 5: Location of the COCO benchmarks, where color and mark combinations represent an algorithm/instance system.

functions. In other words, GWO tends to produce similar behavioral patterns, regardless of the function being explored, which may indicate a systemic bias. This is likely to be a disadvantage, as it shows that the algorithm does not adapt to new environments as well as PSO or SGA. Testing this observation is left for further research. The figure also shows pairs of functions where all algorithms have similar convergence behavior, e.g., (f_1, f_{14}), (f_3, f_4), (f_7, f_{18}), (f_8, f_{13}), (f_{10}, f_{12}), (f_{11}, f_{21}), and (f_{13}, f_{18}).

Table 2 shows the percentage of ttest2 which resulted in their p-values being higher than 0.01, that is, the mean of the feature was not significantly different between a pair of functions. This is a measure of the shared features that produced consistent results for an algorithm across pairs. Broadly, $\{EL_{10}, EL_{25}, EL_{50}, LQ_{50}\}$ are features that indicate multi-modality. Observing the pairs of functions listed above, in most cases these are functions with single funnels. Exploring this observation further is left for future research.

4.3.3 Hypotheses 3 and 4. MANOVA1 found that the means are not co-located, meaning that independent runs of two different algorithms, on either the same or different instances, do not share statistically significant similarities in behavior. Given the results observed for Hypothesis 1 and 2 above, which indicated that PSO and GWO are more consistent across ill-conditioned instances, with GWO potentially suffering from systemic bias, and Figure 4, which indicated that GWO is more consistent across functions, it could be possible to find a set of parameters for which GWO mimics PSO's behavior. However, testing this observation is left for further research.

5 CONCLUSION

We have presented an experimental benchmarking methodology for the behavioral analysis of optimization algorithms for black-box optimization, based on comparing algorithm dynamics in a given problem instance. The results from comparing PSO, GWO and SGA show that, although we did not observe statistically significant behavioral similarities on the experimental conditions tested, it is likely that there is a set of conditions in which similarities could appear. In particular, if there is a set of parameters, or instances with certain characteristics, that would collapse the behavior of one

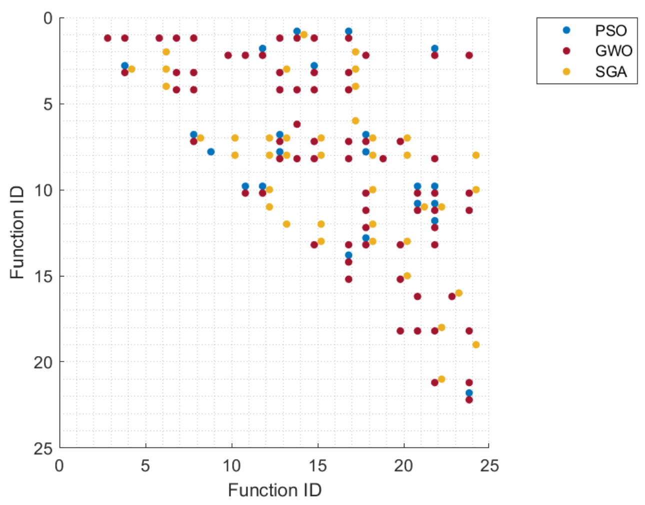


Figure 6: Results from MANOVA1 for testing whether two sets of runs of an algorithm on two functions are co-located (Hypothesis 2). Color marks indicate the combination and the algorithm for which their p-value is less than 0.01, i.e., PSO (●), GWO (●) and SGA (●).

Table 2: Percentage of t-test2 that rejected the null hypothesis for testing whether two sets of runs of an algorithm on two functions are co-located (Hypothesis 2). In bold are percentage values higher than 50%.

Feature	PSO	GWO	SGA	Feature	PSO	GWO	SGA
FDC	31.8%	36.2%	36.6%	ϵ_{max}	18.2%	4.3%	4.9%
$DISP_{1\%}$	31.8%	24.6%	26.8%	M_0	27.3%	17.4%	17.1%
R^2_L	31.8%	44.9%	48.8%	EL_{10}	**86.4%**	**78.3%**	**78.0%**
R^2_{LI}	27.3%	31.9%	29.3%	EQ_{10}	27.3%	30.4%	34.1%
R^2_{QI}	4.5%	11.6%	17.1%	ET_{10}	18.2%	7.2%	17.1%
R^2_Q	18.2%	15.9%	26.8%	LQ_{10}	22.7%	27.5%	26.8%
β_{min}	18.2%	13.0%	14.6%	EL_{25}	**59.1%**	**65.2%**	**63.4%**
β_{max}	22.7%	11.6%	12.2%	EQ_{25}	22.7%	27.5%	34.1%
CN	27.3%	24.6%	34.1%	ET_{25}	22.7%	11.6%	19.5%
$H(Y)$	13.6%	11.6%	2.4%	LQ_{25}	27.3%	24.6%	31.7%
$\xi^{(N)}$	13.6%	20.3%	9.8%	EL_{50}	**54.5%**	**53.6%**	**58.5%**
$\xi^{(1)}$	27.3%	18.8%	19.5%	EQ_{50}	22.7%	20.3%	29.3%
$\xi^{(2)}$	13.6%	20.3%	9.8%	ET_{50}	31.8%	33.3%	36.6%
$\gamma(Y)$	13.6%	11.6%	9.8%	LQ_{50}	50.0%	**52.2%**	**56.1%**
$\kappa(Y)$	9.1%	10.1%	7.3%	PKS	22.7%	20.3%	22.0%
H_{max}	9.1%	10.1%	4.9%				

algorithm into the behavior of another. The first focus of our future work is to investigate these conditions. The results presented above provide us with some clues. More importantly, we observed that GWO appears to exhibit systemic bias, leading to similar behavioral patterns regardless of the function, as long as this does not have ill-conditioning and multiple funnels.

Finally, other techniques from complex systems' analysis may be useful for extending the Behavioral Space. For example, Zelinka et al. [21] proposes representing an algorithm as a Coupled Map Lattice (CML) systems, opening the opportunity for calculating features such as degree centrality, mean neighbor degree, community is done among the others.

ACKNOWLEDGMENTS

This work has been funded by the Australian Research Council through grants No.: FL140100012 and IC200100009. We thank Hao Li and Shing Hei Zhan for their input in earlier stages of this project.

REFERENCES

[1] T. Bartz-Beielstein, C. Doerr, D. van den Berg, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, W. La Cava, M. Lopez-Ibanez, K.M. Malan, J.H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, and T. Weise. 2020. Benchmarking in Optimization: Best Practice and Open Issues. arXiv:2007.03488 [cs.NE]

[2] N. Belkhir, J. Dréo, P. Savéant, and M. Schoenauer. 2017. Per instance algorithm configuration of CMA-ES with limited budget. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM. https://doi.org/10.1145/3071178.3071343

[3] F. Biscani and D. Izzo. 2020. A parallel global multiobjective framework for optimization: pagmo. *J. Open Source Softw.* 5, 53 (2020), 2338. https://doi.org/10.21105/joss.02338

[4] C.L. Camacho-Villalón, T. Stützle, and M. Dorigo. 2020. Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty. In *ANTS 2020*. Springer International Publishing, 121–133. https://doi.org/10.1007/978-3-030-60376-2_10

[5] J.-P Eckmann, S. Oliffson Kamphorst, and D Ruelle. 1987. Recurrence Plots of Dynamical Systems. *Europhys. Lett.* 4, 9 (nov 1987), 973–977. https://doi.org/10.1209/0295-5075/4/9/004

[6] N. Hansen, D. Brockhoff, O. Mersmann, T. Tusar, D. Tusar, O. Ait ElHara, P.R. Sampaio, A. Atamna, K. Varelas, U. Batu, D. Manh Nguyen, F. Matzner, and A. Auger. 2019. *COmparing Continuous Optimizers: numbbo/COCO on Github*. https://doi.org/10.5281/zenodo.2594848

[7] P. Kerschke and H. Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evol. Comput.* 27, 1 (mar 2019), 99–127. https://doi.org/10.1162/evco_a_00236

[8] K.M. Malan and A.P. Engelbrecht. 2014. Characterising the searchability of continuous optimisation problems for PSO. *Swarm Intell.* 8, 4 (2014), 1–28. https://doi.org/10.1007/s11721-014-0099-x

[9] N. Marwan, M. Carmen Romano, M. Thiel, and Jügen Kurths. 2007. Recurrence plots for the analysis of complex systems. *Phys. Rep.* 438, 2007 (2007), 2370329.

[10] N. Marwan, M. Thiel, and N. R. Nowaczyk. 2002. Cross recurrence plot based synchronization of time series. *Nonlinear Process. Geophys.* 9, 3/4 (2002), 325–331. https://doi.org/10.5194/npg-9-325-2002

[11] M.A. Muñoz and K. Smith-Miles. 2017. Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evol. Comput.* 25, 4 (2017), 529–554. https://doi.org/10.1162/EVCO_a_00194

[12] M.A. Muñoz and K.A. Smith-Miles. 2020. Generating new space-filling test instances for continuous black-box optimization. *Evol. Comput.* 28, 3 (2020), 379–404. https://doi.org/10.1162/evco_a_00262

[13] M.A. Muñoz, Y. Sun, M. Kirley, and S.K. Halgamuge. 2015. Algorithm selection for black-box continuous optimization problems: a survey on methods and challenges. *Inform. Sciences* 317 (2015), 224–245. https://doi.org/10.1016/j.ins.2015.05.010

[14] Y. Shi and R.C. Eberhart. 1998. A modified particle swarm optimizer. In *IEEE CEC '98*. 69–73. https://doi.org/10.1109/ICEC.1998.699146

[15] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis. 2014. Towards objective measures of algorithm performance across instance space. *Comput. Oper. Res.* 45 (2014), 12–24. https://doi.org/10.1016/j.cor.2013.11.015

[16] K. Sörensen. 2015. Metaheuristics— the metaphor exposed. *Int. T. Oper. Res.* 22, 1 (Jan. 2015), 3–18. https://doi.org/10.1111/itor.12001

[17] T. Vantuch, I. Zelinka, A. Adamatzky, and N. Marwan. 2018. Phase Transitions in Swarm Optimization Algorithms. In *UCNC 2018*. Springer International Publishing, 204–216. https://doi.org/10.1007/978-3-319-92435-9_15

[18] T. Vantuch, I. Zelinka, A. Adamatzky, and N. Marwan. 2019. Perturbations and phase transitions in swarm optimization algorithms. *Nat. Comput.* 18, 3 (may 2019), 579–591. https://doi.org/10.1007/s11047-019-09741-x

[19] H. Yang. 2011. Multiscale Recurrence Quantification Analysis of Spatial Cardiac Vectorcardiogram Signals. *IEEE Trans. Biomed. Eng.* 58, 2 (2011), 339–347. https://doi.org/10.1109/TBME.2010.2063704

[20] J.P. Zbilut, A. Giuliani, and C.L. Webber. 1998. Detecting deterministic signals in exceptionally noisy environments using cross-recurrence quantification. *Phys. Lett. A* 246, 1-2 (1998), 122–128. https://doi.org/10.1016/s0375-9601(98)00457-5

[21] I. Zelinka. 2015. A survey on evolutionary algorithms dynamics and its complexity – Mutual relations, past, present and future. *Swarm Evol. Comput.* 25 (2015), 2–14. https://doi.org/10.1016/j.swevo.2015.06.002

[22] I. Zelinka, L. Tomaszek, P. Vasant, T. Trong Dao, and Duy Vo Hoang. 2018. A novel approach on evolutionary dynamics analysis – A progress report. *J Comput. Sci.* 25 (2018), 437–445. https://doi.org/10.1016/j.jocs.2017.08.010