Dynamic Sine Benchmark

Almuth Meier

Computational Intelligence Group
Department of Computer Science
University of Oldenburg, Germany

Abstract. This document describes the theoretical background of our Dynamic Sine Benchmark and its algorithmic implementation. The code can be found at `TODO`.

1 Dynamic Sine Benchmark

In [2] we proposed the dynamic sine benchmark (DSB) to address the problem that many benchmarks only allow simple movements of their optima. For example, the previously discussed and frequently applied MPB benchmark only provides noisy linear movement, as well as the FPB benchmark does. Although the CEC competition test sets provide more advanced dynamics, they still are rather simple. For example the recurrent dynamics is defined by a single sine function.

Our DSB problems are quantifiable in their dynamics and are based on moving classic stationary fitness functions, e.g., Sphere or Rosenbrock, with parameterized trigonometric functions, see Figure 1. A related benchmark, that has found application in time series prediction but so far not in dynamic nature-inspired optimization, is the multiple superimposed oscillators (MSO) benchmark [7,6,1,3]. It consists of two or more additive sine functions to examine the performance in signals with large periodicity [4]. In contrast to this, DSB has multiplicative sine functions, which might lead to move complex dynamics, and offers quantifiable complexity

Although we already published the DSB generator [2], in the meantime we simplified the benchmark generator, solved some bugs and improved several details. This updated version is explained in the following.

1.1 Overall Approach

The underlying idea is that the optimum positions are generated by sampling trigonometric functions with a certain step size, see left part in Figure 2. In each dimension w of the solution space the optimum movement follows a separate trigonometric function

$$\zeta_w(c) = \tau + \alpha \cdot \prod_{i=1}^{\rho} \left(\iota_i \cdot \sin\left(\beta_i \cdot \kappa \cdot (c-1) + \gamma_i\right) \right) \tag{1}$$

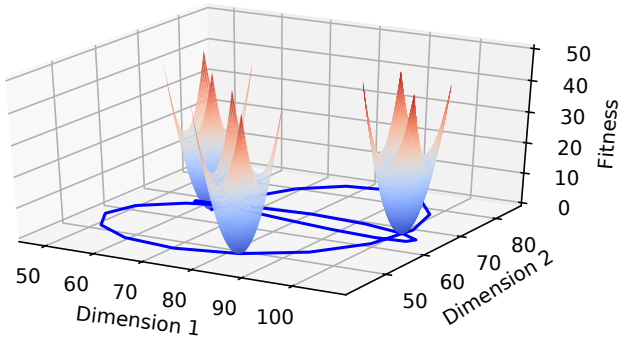

Fig. 1: Optimum position of Sphere function in a two-dimensional solution space at three points in time

which is composed by multiplying ρ sine functions, called component functions, with random amplitude ι_i, frequency β_i, phase shift γ_i, vertical movement τ, and overall scaling α. We call ζ_w composite function as well. In change period c the optimum is located at the sampled position

$$\mathbf{o}_c = [\zeta_1(c), \zeta_2(c), \ldots, \zeta_d(c)]\,, \tag{2}$$

which is visualized in the right part of Figure 2. To generate optimum positions for P change periods, each function ζ_w is evaluated for $c \in [1, \ldots, P]$. In ζ_w, step size κ determines the distance between sampling points. This can be illustrated with an alternative formulation where the optimum position is written as

$$\mathbf{o}_c = [\zeta'_1(\kappa \cdot (c-1)), \zeta'_2(\kappa \cdot (c-1)), \ldots, \zeta'_d(\kappa \cdot (c-1))]\,, \tag{3}$$

where the sine functions that ζ'_w consists of are written in the basic form with x as point in time at which ζ'_w is sampled:

$$\zeta'_w(x) = \tau + \alpha \cdot \prod_{i=1}^{\rho} \left(\iota_i \cdot \sin\left(\beta_i \cdot x + \gamma_i\right)\right) \tag{4}$$

From Equation (3) it can easily be obtained that the distance between successive points at that ζ'_w is sampled equals step size κ in each dimension:

$$\kappa \cdot ((c+1)-1) - (\kappa \cdot (c-1)) = \kappa \cdot c - \kappa \cdot c + \kappa = \kappa$$

In Equation (1), $c-1$ is used instead of c to begin the sampling at point 0 because c is defined with $c \geq 1$, see Equation (**??**). Step size κ requires a careful choice to cover all important parts of function ζ_w. Therefore, in the updated benchmark we set it automatically as described later.

DSB can be combined with any stationary fitness function f_s. While shape and level of fitness function f_s remain unchanged, function ζ_w defines the location of the landscape in the solution space. It determines the position of the global

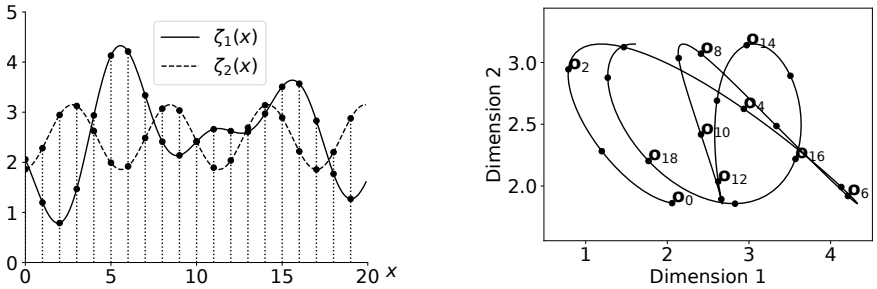

Fig. 2: Left: sampling of ζ_w at 20 points in time. Right: corresponding optimum positions $\mathbf{o}_c$ in the solution space

optimum as the fitness landscape's anchor, see Figure 1. If f_s has multiple global optima, one of them has to be chosen as anchor.

The fitness for individual $\mathbf{x}$ in change period c is

$$f(\mathbf{x}, c) = f_s(\mathbf{x} - (\mathbf{o}_c - \mathbf{o}_{f_s}))\text{where} \tag{5}$$

where $\mathbf{o}_{f_s}$ denotes the global optimum of the unmoved function f_s.

In order to quantify the benchmark's complexity, we introduce the concepts curviness and velocity. Curviness $C \in \mathbb{N}_{>0}$ specifies the number of extremes ζ_w has within 100 samples at successive equidistant points. Thus, curviness determines how many changes of the optimum's movement direction take place on the first 100 optimum positions. Since trigonometric functions are periodic, the overall number of extremes can be deduced based on the curviness. By this means, the curviness indicates the difficulty for a prediction model to track the optimum. Velocity $V \in \mathbb{R}_{>0}$ is the median distance between successive optimum positions and represents the problem difficulty for the optimization algorithm. In case the velocity is much higher than the mutation strength, an ES without prediction might need many more generations to find the new optimum position. To generate a DSB instance with specified velocity and curviness, scaling factor α and frequencies β_i have to be chosen appropriately. The generation process and the mathematical relations are described in the next paragraphs.

In a d-dimensional DSB instance, for dimensions $w \in [1, \ldots, d]$ the functions ζ_w are generated with the same curviness and velocity in order to ensure that the complexity of DSB instances with different dimensionality but equal curviness and velocity only depends on the number of dimensions. Figure 3 visualizes DSB instances with different parameterizations for velocity and curviness. Apparently, curviness influences the shape of ζ_w, while velocity influences the vertical scaling. For example, the range of DSB instances with $V = 1.0$ is twice as large as the range of instances with $V = 0.5$.

1.2 Algorithmic Implementation

The DSB generator provides parameters that have to be set to construct a DSB instance with desired properties. They are listed in the following.

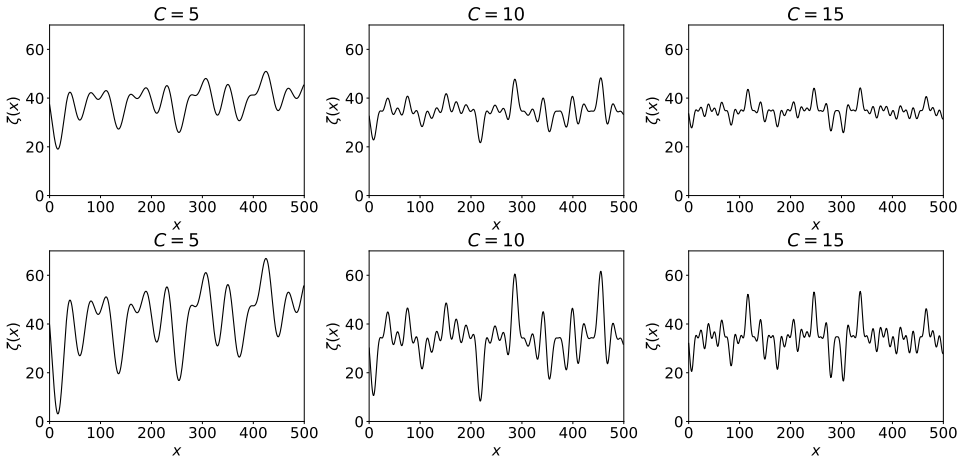

Fig. 3: 500 optimum positions of 1-dimensional DSB instances ($\rho_{\max} = 4$) with varying curviness and velocity. First row: $V = 0.5$, second row: $V = 1.0$. From left to right: $C \in \{5, 10, 15\}$

$d \in \mathbb{N}_{>0}$	dimensionality
$b^l, b^u \in \mathbb{R}$	lower and upper bounds for range of ζ_w; correspond to bounds of solution space
$P \in \mathbb{N}_{>0}$	number of samples; equal to number of change periods P and optimum positions
$C \in \mathbb{N}_{>0} \wedge C < 100$	curviness
$V \in \mathbb{R}_{>0}$	velocity
$\rho_{\max} \in \mathbb{N}_{>0}$	maximum number of functions that are composed to ζ_w

Furthermore, the seed of the random number generator can be set to allow for reproducibility.

In our definition, curviness counts the extremes of ζ_w per 100 samples. The minimum curviness realizable is one, since the generator can only count discrete numbers of extremes on a given set of samples. Theoretically, it might be interesting to examine benchmarks with lower curviness values, e.g., $C = 0.6$. To construct such an instance, the number of samples which curviness refers to, must be increased in the generator's implementation. For the given example, 500 samples would be appropriate since $0.6 \cdot 5 = 3$ would result in a discrete number of extremes on 500 points. However, if the number of reference samples is changed, it should be mentioned in the description of the experiments to enable comparability with other studies.

The benchmark generation process works as follows; pseudocode is provided in Algorithm 1. Based on the user-defined specification, parameters are set and upper bounds $\iota_{\max}$ and $\beta_{\max}$ for parameters ι and β of the functions ζ_w are determined (Lines 4-7). Then, functions ζ_w are preliminarily parameterized (Lines 9-12). The chosen parameter values are corrected after all functions have been

initialized in order to comply with the specification (Lines 14-16.) Justification for the upper bounds and explanation of the parameter correction is given in the next paragraph.

Algorithm 1 DSB generator

1: **given** $d, b^l, b^u, \rho_{\max}, P, C, V$
2: **for** $w = 1, \ldots, d$ **do**
3: # set constants and upper bounds for parameters
4: $\rho \sim \mathcal{U}(1, \rho_{\max})$
5: $\iota_{\max} = \left\lfloor \sqrt[\rho]{\frac{b^u - b^l}{2}} \right\rfloor$
6: $\kappa = 1$
7: $\beta_{\max} = \frac{1}{2 \cdot \kappa}$
8: # parameterize function ζ_w randomly
9: **for** $i = 1, \ldots, \rho$ **do**
10: $\iota_i \sim \mathcal{U}(-\iota_{\max}, \iota_{\max}) \wedge \iota_i \neq 0$
11: $\beta_i \sim \mathcal{U}(-\beta_{\max}, \beta_{\max}) \wedge \beta_i \neq 0$
12: $\gamma_i \sim \mathcal{U}\left(0, \frac{2\pi}{\beta_i}\right)$
13: # adapt parameterization
14: $\beta_i \leftarrow$ correct curviness
15: $\alpha \leftarrow \frac{V}{V_{\zeta_w}}$ # V_{ζ_w}: velocity realized by ζ_w
16: $\tau \leftarrow$ correct range

1.3 Mathematical Justification

Roughly following the order of operations in Algorithm 1, we describe in this paragraph how the upper bounds for parameters are determined and how the generator fulfills the specifications regarding curviness, velocity and range.

Number of Component Functions The maximum number $\rho_{\max}$ of component functions forming ζ_w can be chosen arbitrarily. However, our studies showed that $\rho_{\max} < C$ is sufficient. Otherwise, interference becomes likely so that ζ_w exhibits extreme peaks or flat regions, see Figure 4 for $\rho_{\max} = 14$, which might be unusual in real-world applications. In our applications, $\rho_{\max} = 4$ was a reasonable choice independent of C. In each function ζ_w, we randomly chose ρ (Line 4) to allow for more diverse function shapes.

Amplitudes The upper bound $\iota_{\max}$ for amplitudes ι_i of component functions (Line 5) depends on range[1] $[b^l, b^u]$ allowed for composite function ζ_w. In order to

[1] Without loss of generality, we depict the benchmark generator with equal bounds b^l and b^u for all ζ_w, while different bounds are applicable as well.

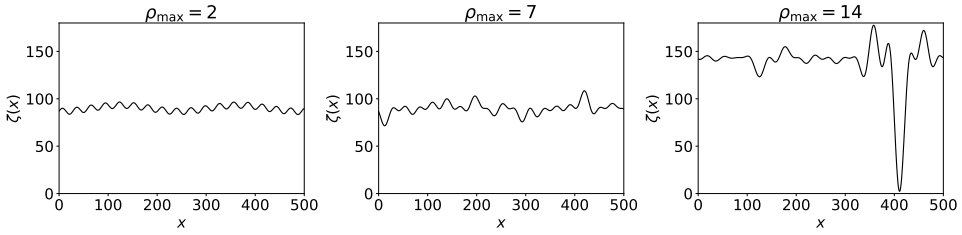

Fig. 4: 500 optimum positions of 1-dimensional DSB instances ($C = 7, V = 0.5$) with varying $\rho_{\max}$. From left to right: $\rho_{\max} \in [2, 7, 14]$

match it, the amplitude of ζ_w must be less or equal to $\frac{b^u - b^l}{2}$ since the maximum range of a trigonometric function is twice its amplitude. The amplitude of a function consisting of ρ multiplied trigonometric functions is the product of the component amplitudes ι_i which leads to the condition $\prod_{i=1}^{\rho} \iota_i \leq \frac{b^u - b^l}{2}$. Applying the ρ-th root results in the upper bound for amplitudes ι_i. For simplification, the result is rounded to an integer value:

$$\iota_{\max} = \left\lfloor \sqrt[\rho]{\frac{b^u - b^l}{2}} \right\rfloor \tag{6}$$

Step Size As described earlier, the optimum positions are samples from functions ζ_w. A proper choice for step size κ is necessary to sample all important parts of the function ζ_w. The Nyquist-Shannon sampling theorem [5] defines the relation $\kappa < \frac{1}{2 \cdot \beta_i}$. We chose $\kappa = 1$ (Line 6) as fixed step size and parameterize the component functions ζ_w with frequencies compatible with κ, see next paragraph. Any other value would be possible for κ as well, the possible specifications, e.g., for curviness, would be the same.

Frequency (Curviness) Frequency and curviness are directly related. Relying on the largest possible frequency $\beta_{\max}$ that follows from the Nyquist-Shannon theorem (Line 7), the maximum curviness $C_{\max}$ that ζ_w theoretically can realize is

$$C_{\max} = \sum_{i=1}^{\rho} |50 \cdot \beta_{\max}|. \tag{7}$$

Value $C_{\max}$ is the sum of the largest possible curviness values that the ρ component functions can achieve. The maximum curviness of a component function is deduced as follows. Any sine function consists of 2 extremes within each period. If the function is sampled with the upper bound for the step size according to the Nyquist-Shannon theorem, one period is covered by four sampling points:

$$\kappa < \frac{1}{2 \cdot \beta_i} = \frac{1}{2 \cdot \frac{2\pi}{L}} = \frac{L}{4\pi} \tag{8}$$

with period length L as multiple of π and the known relationship for trigonometric functions $\beta_i = \frac{2\pi}{L}$. Thus, not more than two extremes can lie within four sampling points, which is visualized in Figure 5. Our generator specifies the curviness in relation to 100 samples. Thus, any ideally sampled function has $\frac{100}{4}$ periods per 100 samples and thus $\frac{100}{4} \cdot 2 = 50$ extremes within 100 samples. Therefore, $C = 50$ is the largest realizable curviness for this and any other function. From this follows that sine functions with $\beta_i \leq \beta_{\max}$ provide $50 \cdot \beta_i$ extremes per 100 samples. Summing up the component curviness values results in the theoretical curviness of ζ_w:

$$C_{\text{theo}} = \sum_{i=1}^{\rho} |50 \cdot \beta_i| \tag{9}$$

This leads to the upper bound $C_{\max}$ in Equation (7) if every β_i equals $\beta_{\max}$. In practice, C_{theo} is not necessarily equal to the curviness C_{ζ_w} realized by ζ_w due to two reasons. First, the component functions might not be ideally sampled because of the phase shift. Second, we compute curviness C_{ζ_w} empirically. It is determined by computing the number of sign changes of the differences $\mathbf{o}_c - \mathbf{o}_{c-1}$ for all $c \in \{1, \ldots, P\}$. Therefore peaks located at a border cannot be detected, for example the extreme at point 4π in Figure 5.

In the implementation, we restrict C to be less than 100, since 100 is the number of samples the definition of C refers to, and it is not possible to have more extremes than samples. However, we allow $C > C_{\max}$, even if it is only realizable by violating the Nyquist-Shannon condition for β_i, since in some applications the resulting benchmark might be useful though the functions are not ideally sampled.

During the generation process, first all frequencies are chosen from interval $[-\beta_{\max}, \beta_{\max}]$ regardless of the realized curviness, see Line 11 in Algorithm 1. After the component functions' parameterization was completed by setting the phase shift, the frequencies are adapted so that ζ_w complies with the specified curviness (Line 14). This works as follows.

All frequencies that do not become too large by this operation are multiplied with factor $\frac{C}{C_{\zeta_w}}$. In case the specified curviness can only be realized by violating the bounds of all β_i, all frequencies are multiplied with $\frac{C}{C_{\zeta_w}}$. The correction is done iteratively since multiple steps might be necessary to achieve C. If the correction was not successful for a specified number of iterations, e.g., when the frequencies are increased and decreased by the same factor, only a random subset of frequencies is updated. By this means, it becomes more probable to find a proper parameterization. After the correction procedure, some β_i might exceed $\beta_{\max}$. Thus, the sampling would not be mathematically exact, but for dynamic optimization as application context it should be sufficient.

Phase Shift Phase shift γ_i is chosen from $[0, \frac{2\pi}{\beta_i})$, where $\frac{2\pi}{\beta_i}$ is the period length of the corresponding component function (Line 12). This interval is sufficient because sine functions are periodic so that phase shifts with larger or smaller

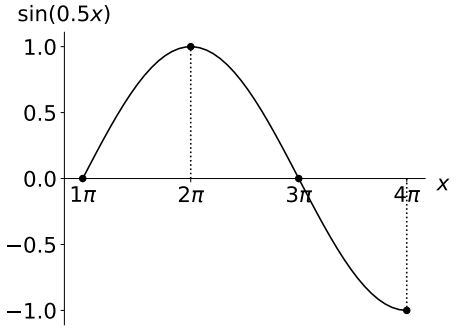

Fig. 5: Ideal sampling of a period of an arbitrary sine function. Black dots mark sampling points

values can be expressed by values within this interval as well. It is reasonable to parameterize the component functions with different phase shifts in order to lower the risk of interference and to increase the number of possible functions.

Scaling (Velocity) The overall scaling factor α influences the velocity. It is set to $\frac{V}{V_{\zeta_w}}$ to achieve the specified velocity (Line 15). Here, V_{ζ_w} is the velocity realized by ζ_w and is equal to the median distance of the differences $\mathbf{o}_c - \mathbf{o}_{c-1}$ for all $c \in \{1, \ldots, P\}$. The scaling factor is computed after curviness has been corrected, since changes in the frequencies might change the velocity again.

Vertical Movement (Range) The term τ moves ζ_w vertically such that the range covered by ζ_w matches the specified interval $[b^l, b^u]$ (Line 16). It might be the case, that no such movement can be found as the range of ζ_w is too wide. In this case ζ_w cannot be corrected because adapting the scaling α in turn would violate the velocity specification. The only solution would be that the user selects a larger range.

2 Update September 2019

Changes to original version:

- simplified correction procedure to comply with desired velocity (scaling factor)
- bugfix: comply with range (vertical movement was not correct in every case)
- automatic computation of some parameters (amplitude, step size)
- curvature referes now to fixed number samples (100), not to concrete interval [0,2π)
- new generation process for frequencies and additional correction procedure (did not exist before)
- concrete value for κ is irrelevant, so any value < 100 is possible for C

– phase shift γ_i depends now on β_i
– allowed negative amplitudes and frequencies
– abolished some restrictions, e.g. regarding $\rho_{\max}$ and C

References

1. D. Koryakin and M. V. Butz. Reservoir sizes and feedback weights interact non-linearly in echo state networks. In *International Conference on Artificial Neural Networks (ICANN)*, pages 499–506, 2012.
2. A. Meier and O. Kramer. Predictive uncertainty estimation with temporal convolutional networks for dynamic evolutionary optimization. In *International Conference on Artificial Neural Networks (ICANN)*, pages 409–421, 2019.
3. S. Otte, P. Rubisch, and M. V. Butz. Gradient-based learning of compositional dynamics with modular RNNs. In *International Conference on Artificial Neural Networks (ICANN)*, pages 484–496, 2019.
4. B. Roeschies and C. Igel. Structure optimization of reservoir networks. *Logic Journal of the IGPL*, 18(5):635–669, 2010.
5. C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, Jan 1949.
6. J. J. Steil. Several ways to solve the MSO problem. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 489–494, 2007.
7. D. Wierstra, F. J. Gomez, and J. Schmidhuber. Modeling systems with internal state using evolino. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 1795–1802, 2005.