mathematics

MDPI

Article

Dynamical Sphere Regrouping Particle Swarm Optimization Programming: An Automatic Programming Algorithm Avoiding Premature Convergence

Martín Montes Rivera [1,2,*], Carlos Guerrero-Mendez [1,*], Daniela Lopez-Betancur [1] and Tonatiuh Saucedo-Anaya [1]

[1] Unidad Académica de Ciencia y Tecnología de la Luz y la Materia, Universidad Autónoma de Zacatecas, Campus es Parque de Ciencia y Tecnología QUANTUM, Cto., Marie Curie S/N, Zacatecas 98160, Mexico; danielalopez106@uaz.edu.mx (D.L.-B.); tsaucedo@uaz.edu.mx (T.S.-A.)

[2] Research and Postgraduate Studies, Department of Universidad Politécnica de Aguascalientes, Aguascalientes 20342, Mexico

* Correspondence: martin.montes@upa.edu.mx (M.M.R.); guerrero_mendez@uaz.edu.mx (C.G.-M.)

Abstract: Symbolic regression plays a crucial role in machine learning and data science by allowing the extraction of meaningful mathematical models directly from data without imposing a specific structure. This level of adaptability is especially beneficial in scientific and engineering fields, where comprehending and articulating the underlying data relationships is just as important as making accurate predictions. Genetic Programming (GP) has been extensively utilized for symbolic regression and has demonstrated remarkable success in diverse domains. However, GP's heavy reliance on evolutionary mechanisms makes it computationally intensive and challenging to handle. On the other hand, Particle Swarm Optimization (PSO) has demonstrated remarkable performance in numerical optimization with parallelism, simplicity, and rapid convergence. These attributes position PSO as a compelling option for Automatic Programming (AP), which focuses on the automatic generation of programs or mathematical models. Particle Swarm Programming (PSP) has emerged as an alternative to Genetic Programming (GP), with a specific emphasis on harnessing the efficiency of PSO for symbolic regression. However, PSP remains unsolved due to the high-dimensional search spaces and local optimal regions in AP, where traditional PSO can encounter issues such as premature convergence and stagnation. To tackle these challenges, we introduce Dynamical Sphere Regrouping PSO Programming (DSRegPSOP), an innovative PSP implementation that integrates DSRegPSO's dynamical sphere regrouping and momentum conservation mechanisms. DSRegPSOP is specifically developed to deal with large-scale, high-dimensional search spaces featuring numerous local optima, thus proving effective behavior for symbolic regression tasks. We assess DSRegPSOP by generating 10 mathematical expressions for mapping points from functions with varying complexity, including noise in position and cost evaluation. Moreover, we also evaluate its performance using real-world datasets. Our results show that DSRegPSOP effectively addresses the shortcomings of PSO in PSP by producing mathematical models entirely generated by AP that achieve accuracy similar to other machine learning algorithms optimized for regression tasks involving numerical structures. Additionally, DSRegPSOP combines the benefits of symbolic regression with the efficiency of PSO.

Keywords: automatic programming; symbolic regression; particle swarm programming; particle swarm optimization

MSC: 68T20

check for updates

Citation: Montes Rivera, M.; Guerrero-Mendez, C.; Lopez-Betancur, D.; Saucedo-Anaya, T. Dynamical Sphere Regrouping Particle Swarm Optimization Programming: An Automatic Programming Algorithm Avoiding Premature Convergence. *Mathematics* **2024**, *12*, 3021. https://doi.org/10.3390/math12193021

Academic Editor: Ioannis Tsoulos

Received: 1 September 2024
Revised: 17 September 2024
Accepted: 21 September 2024
Published: 27 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Identifying the most appropriate model that matches the system's characteristics is essential for achieving accurate predictions, classifications, and simulations. Deterministic

models can be inferred using mathematical and physical principles, while probabilistic approaches are employed when deterministic models are not available [1]. Machine Learning (ML) techniques offer several alternatives to optimize the numerical parameters in various models proposed by humans, such as in [2] using linear regression models for modeling urban freight generation, in [3] using Gaussian regression for prediction of product yields from lignocellulosic biomass, in [4] applying polynomial regression for environmental assessment of a diesel engine, in [5] obtaining the vertices in Fuzzy Inference Systems (FIS) for controlling a pneumatic positioning system, and in [6] finding the weights in Artificial Neural Networks (ANNs) for predicting money laundering. However, the optimization of numerical parameters in these models may restrict the range of potential behaviors that can be approximated to represent the system. Furthermore, advanced models such as deep ANNs are more challenging to interpret and difficult to share compared to mathematical models or equations due to their extensive numerical parameters [7]. Alternatively, it is feasible to employ ML algorithms to determine the appropriate structure or mathematical model that best fits the desired behavior through Automatic Programming (AP). AP facilitates the creation of program models or equations without requiring human intervention or models proposed by humans. Furthermore, if AP is utilized to produce mathematical models that illustrate a process, the outcomes can be interpreted in terms of symbolic regression. Since the final result is an equation model, it is also simpler to share than deep ANNs [1]. Genetic Programming (GP) is an algorithm proposed by John R. Koza in 1994 that has gained widespread use in the field of AP. GP applies mechanisms of evolutionary algorithms inspired by natural selection principles proposed by Charles Darwin. These mechanisms include a randomly generated population, fitness evaluation, selection of the fittest, crossover, and mutation. GP has proven to be a powerful tool for solving complex problems and generating models and structures in various application domains [1,8]. GP has been used to predict population dynamics [8], to model and optimize the removal of heavy metals [9], to solve lot-sizing and job shop scheduling problems [10], to predict COVID-19 using routine hematological variables [11], to measure the density and viscosity of carbon dioxide-loaded [12], to optimize job scheduling with linear representation [13,14], to perform semantic symbolic regression [15], to obtain models for sensor linearization [16], to generate models to describe forward kinematics in robots [17], to search phenotype trajectory networks [18] that produce new structures like spatial representation [19], to optimize dynamic scheduling with graph representation [20], to conduct automatic clustering [21], and to measure the impact of oil prices in the economy [22], among others. While GP has shown impressive results in a variety of applications, its computational demands are significant due to its inspiration from evolutionary algorithms, which involve selecting the most fit individuals, performing crossover operations, mutation, and various conversions to interpret the models [23,24]. Other researchers have made improvements to Koza's original GP proposal. These modifications include linear, Cartesian, and graph representations of the program that have shown promise in allowing for the creation of more efficient and effective programs [25,26]. However, GP still depends on the evolutionary mechanisms that increase complexity and computational demanding power. In contrast, Swarm Intelligence (SI) inspired by social learning mechanisms has demonstrated remarkable performance in numerical optimization, delivering outcomes comparable to evolutionary algorithms but with better parallelism and reduced computational expenses [27]. Additionally, SI delivers better results than evolutionary algorithms when the number of function calls is restricted [28]. Swarm Programming (SP) brings the benefits of SI to AP, offering less demanding computation and faster results than those obtained with GP. There are different SP alternatives according to the swarm algorithm basis, including Ant Programming (AnP), Particle Swarm Programming (PSP), Bee Swarm Programming (BSP), Herd Programming (HP), Artificial Fish Swarm Programming (AFSP), and Firefly Programming (FP), among others [28]. AnP is the most used SP technique and has been used in applied research for optimizing energy saving in a building management system [27], for industrial robot programming in a digital twin [29], and for effective optimal design of sewer networks [30].

GP is more complex and costly than PSP due to the inherited mechanisms in their numerical optimization counterparts. GP requires several computation loops used in the evolutionary algorithms to handle their mechanisms; it also has a lower level of parallelization than PSP since it works together as a population. On the other hand, PSP changes each particle's behavior independently by evaluating only two equations for determining the speed vector and position vector, allowing a high level of parallelization with fewer computation loops. Nevertheless, PSP inspired in the Particle Swarm Optimization (PSO) algorithm has been limited in applied research. The definitions of PSP can be traced back to the studies published in [31,32]. Ref. [31] definition describes PSP as the implementation of PSO to obtain tree representations in GP. Ref. [32] defines the PSP as a grammatical swarm, which the author defines as the generation of programs with PSO. The limited availability of applications in PSP can be attributed to its suboptimal performance, which stems from its reliance on PSO. PSO tends to encounter stagnation issues when handling intricate search spaces with numerous local optimal regions. The premature convergence in the conventional PSO occurs when all particles are attracted to a local optimum, resulting in stagnation as the velocity equation relies on both the global best position and particle best position to determine the direction of the speed vector. Consequently, when particles are concentrated near a local optimal region, they cannot escape; thus, all particles get stuck in the same position. This is especially problematic in complex and high-dimensional spaces where the likelihood of encountering local optimal regions is increased. The stagnation problem is particularly pronounced in AP, where high-dimensional search spaces with multiple local optimal regions are expected [33]. To overcome the limitations of PSO in PSP, researchers have explored the possibility of integrating PSO with Dynamic Programming (DP). This approach entails breaking down the search space into smaller, less complex entities using a divide-and-conquer strategy that optimizes partial results. By simplifying the search space in this way, the AP problem becomes more manageable and can be effectively solved [34]. DP has been used in generating structures for optimizing the Bellman equation with linear programming [35], in the optimization of the Mula river reservoir together with PSO [36], in identifying hydropower unit commitment together with genetic algorithms [37], and in the adaptative design of intelligent controllers [38], among others. On the other hand, in high-dimensional complex search spaces with several local optimal regions, such as those in AP, PSP could benefit from modifications to the original PSO that researchers have proposed to address issues with premature convergence and stagnation in local optimal regions. The PSO modifications include

- GPSO, a modified version of PSO [39] that includes inertial behavior; this modification was proposed by the authors of the original PSO.
- GEPSO [40], which also uses inertial behavior but with three coefficients and a new speed equation that intends to improve the algorithm convergence.
- PSO with attractive search space border points [41], which shifts particles towards the search space boundaries to avoid stagnation; regrouping PSO that resets the position of particles when stagnation is perceived by sensing minimal distance among the particles' position [42].
- Canonical Deterministic PSO [43], which detects stagnation and reinvigorates the particle positions after iterations without improvement.
- IAPSO [44], which reinvigorates the swarm after detecting stagnation through entropy analysis.
- Multi-Swarm PSO alternatives that regulate exploration and exploration of the algorithm based on topological groups of particles [45].
- Dynamical Sphere Regrouping PSO (DSRegPSO), which is a modified version of PSO that introduces a sphere regrouping mechanism and momentum conservation effect [33].

These mechanisms continually reinvigorate the swarm, avoiding stagnation and obtaining remarkable results in large-scale global optimization of non-linear and non-separable subcomponents in functions with up to 1000 dimensions and several local optimal regions. The DSRegPSO algorithm capabilities could be integrated into AP as an alternative

for implementing PSP. This will allow larger entries for optimization in DP and even optimizing the entire structure in a single trial. Moreover, integrating DSRegPSO into PSP will enable it to have more applications beyond its theoretical implementation.

Our work introduces the Dynamical Sphere Regrouping PSO Programming (DSRegPSOP), a PSP implementation that employs the DSRegPSO algorithm with sphere regrouping and conservation momentum effects. This implementation is designed to handle large-scale search spaces that contain multiple local optimal regions, like when generating mathematical expressions with AP or symbolic regression. We evaluated our proposal by generating ten mathematical expressions for mapping points obtained from proposed functions representing various complexity levels akin to regression problems. In addition, we thoroughly assess DSRegPSOP and compare it with other algorithms found in the literature using real-world datasets. Our results demonstrate that we have effectively addressed the problem of PSP in practical applications, as stated in [31,32].

1.1. Contribution

The primary innovation in our work involves the implementation of the DSRegPSOP algorithm, which effectively incorporates dynamical sphere regrouping and conservation momentum effects into an automatic programming algorithm based on PSO (PSP). This addresses the limitations of PSP related to the PSO stagnation problem. As a result, our algorithm enables the derivation of intricate regression models from real-world datasets, producing results similar to those of other algorithms in the literature, such as artificial neural networks, while yielding symbolic regression equations. This level of performance in regression was not possible with previous PSP implementations.

1.2. Limitations

The effectiveness of DSRegPSOP in PSP was evaluated by comparing it with other machine learning algorithms in the regression of six datasets: Airfoil, Concrete Strength, Cooling Load, Yachts Sailing Resistance, Geographical Location of Music, and Tecator Fat of Meals. This approach was chosen because there is no established benchmark for symbolic regression algorithms, and comparisons are typically based on assessments in datasets. The six selected datasets found in the literature were used to compare various regression algorithms based on RMSE values; the datasets were available online.

To obtain optimal hyperparameters and train the algorithm using k-fold cross validation across six datasets, it was necessary to iterate several times. Consequently, we only compared DSRegPSOP with results from the literature. The algorithms we compared DSRegPSOP with include Partial Least Square Regression, Tree Regression, Random Forest, XGBoost, Ridge Regression, Generalized Additive Model, Position-gradient type fuzzy model, Neuro-fuzzy model based on extended RVM, NN model based on ELM, NN model based on IELM, NN model based on CIELM, ANN with L2 regularization, Linear Regression, Gaussian Regression, Multilayer Perceptron, SVM, ANN, ANFIS, Lasso Regression, and DNN. Although other algorithms could have been compared with DSRegPSOP, they have not been evaluated before using similar datasets available online.

However, we have proposed a benchmark with regression functions varying complexity levels achieved by increasing the problem dimensionality, inspired by the large-scale global optimization benchmark CEC'13. The benchmark involves multiple computations similar to the CEC'13 test, but we have only performed this evaluation within our algorithm, DSRegPSOP, since it requires several computations to be completed. We anticipate that other regression algorithms will adopt these functions as a standard regression test for comparison with DSRegPSOP.

The rest of the paper is organized as follows: Section 2 presents the DSRegPSOP algorithm, including the definitions, algorithms, transformations, and benchmarks required to introduce it. Section 3 presents the obtained results in all assessments conducted for the evaluation of DSRegPSOP, including statistical proofing to support that it obtains similar

efficiency to other algorithms in regression tasks. Finally, Section 4 presents the conclusions with the relevant findings in our proposal based on the results in Section 3.

2. Materials and Methods

2.1. Symbolic Regression with AP Mathematical Expressions for Position of Particles

The implementation of AP for generating mathematical models can be done using various alternatives, such as the tree (green), graph (cyan), cartesian (yellow), and linear representations (orange) shown in Figure 1 [25,26].

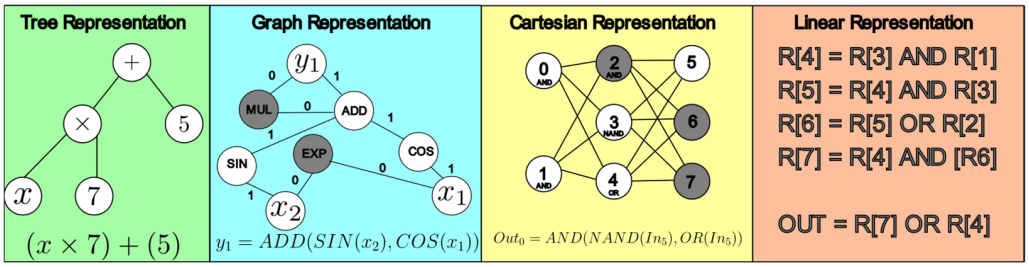


Figure 1. Different AP representations for the generation of mathematical expressions.

DSRegPSOP applies a string transformation defined as TS for $\overrightarrow{X}_i$ (the position of every i particle of the n particles and d dimensions) to generate $\overrightarrow{M}_i$, with n_l linear mathematical representations; each one is contained in a register m_j; i.e., for all the particles the transformation is $TS : X \in R^{n \times d} \mapsto M \in S^{n \times ds}$, with ds as the total number of characters in each m_j. Every m_j is an equation that depends on previous registers, with the final output equation being the last register ($j = n_l$). In Figure 2, a visual representation is presented, showcasing the spatial distribution of 100 particles, each characterized by 38 dimensions. Furthermore, the figure displays the equation and surface pertaining to the best particle in that iteration. This particle has two equation registers, i.e., $n_l = 2$.

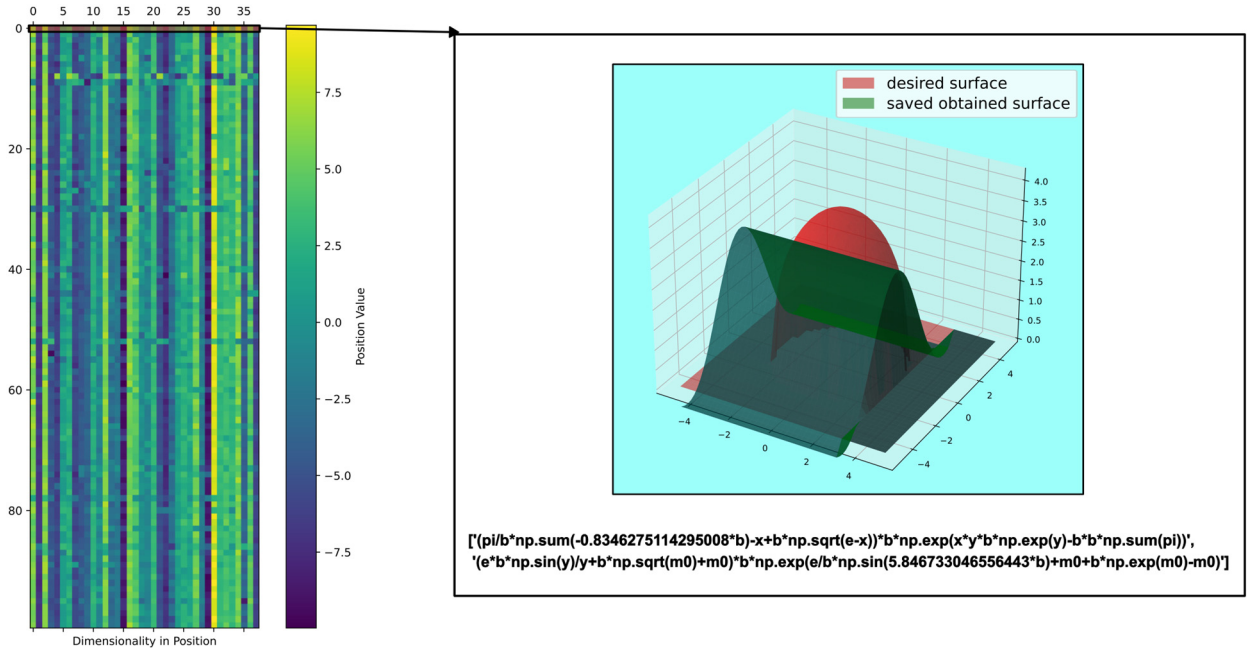


Figure 2. Position of particles for generation of mathematical expressions in AP.

The complexity of a mathematical expression in a DSRegPSOP depends on several factors, including the operators, coefficients, variables, groups, and levels. These elements can be affected if a model is well-fitted and sub- or over-parameterized.

The coefficients (a, b, c, π, and e) and variables (x, y, and z) use operators like $+$, $-$, $\times$, $\div$, $\wedge$, $sin(\ )$, and $cos(\ )$, $(,)$, among others, in the mathematical expressions. Groups are sets of operations enclosed in parentheses and separated by operators. Each group forms a mathematical expression with operators, coefficients, and variables. The mathematical expressions have n_l levels—the registers containing m_j mathematical expressions with n_g groups, every group with n_v variables and coefficients, and n_o operators. The output model is the final m_j function in the last level (n_l) that depends on previous registers, like in Figure 3 with $n_l = 4$, $n_g = 2$, $n_o = 2$, and $n_v = 3$.

Registers or levels Groups Coefficients Operators

```
out = [m0 = (b-b*np.sqrt(pi-e))/(e*b*np.sum(2.652675820816853*b)-0.35510508673154817*b),
       m1 = (b+(m0)-e)/(x+b*np.sin(pi)*pi),
       m2 = (m1*b/(pi))/(x+e**m0),
       m3 = (m0+b*np.sin(m0)+m2)/(m2-b*np.sum(m0)*pi)]
```

Figure 3. Example of mathematical expression generated with DSRegPSOP.

The dimensionality required for the position of particles relies on the number of operators (n_o), coefficients and variables (n_v), groups (n_g), and levels (n_l) according to Equation (1).

$$d = n_l \cdot \left[n_g \cdot (n_o + n_v) + (n_g - 1) \right] \tag{1}$$

To convert the particle's position into equations, we transform the position into two integer matrices ($Z^{n \times d}$). One matrix contains all the particles of the indexes for operators ($M_o \in Z^{n \times d}$), and the other one includes the variables and coefficients ($M_v \in Z^{n \times d}$). Equations (2) and (3) allow for obtaining those indexes for every particle. These matrices are the indexes that allow re-mapping of the particle's position to the corresponding operators saved in L_o or the variables saved in L_v.

$$\vec{M}_{o,i} = \frac{\vec{X}_i - L_l}{L_u - L_l} \cdot (n_o - 1) \tag{2}$$

$$\vec{M}_{v,i} = \frac{\vec{X}_i - L_l}{L_u - L_l} \cdot (n_v - 1 + n_c) \tag{3}$$

where n_c is the number of positions in the list of variables for coefficients. Given that each variable has an equal probability of being selected (i.e., there is one space in the list for each variable), this approach enables the control of the number of positions that are allocated to a numeric coefficient.

After obtaining $\vec{M}_{v,i}$ and $\vec{M}_{o,i}$, m_j adds the corresponding $L_v\left(\vec{M}_{v,i}\right)$ or $L_o\left(\vec{M}_{o,i}\right)$ depending on whether the dimension is a pair, the assignation takes the index from $\vec{M}_{v,i}$ for odds and $\vec{M}_{o,i}$ for pairs. Thus, each group in the equation has $n_v + n_o$ dimensions. The groups are delimited by parentheses and operators determined by the particles' positions. Then, every level uses $n_g \cdot (n_v + n_o) + (n_o - 1)$ dimensions of $\vec{X}_i$ containing n_g groups. However, n_v increases with every new m_j, since it adds the variable that represents the previous register; thus, $\vec{M}_{v,i}$ is re-calculated.

When the index of variables is greater than its number ($\vec{M}_{v,i} > n_v$) a numerical coefficient is determined instead of a variable. Then, for that dimension, Equations (4) and (5) obtain the lower (L_{lc}) and the upper (L_{uc}) limits for re-mapping the coefficient to the search space range $[L_l, L_u]$. Equation (6) obtains the numerical coefficient (c).

$$L_{lc} = L_l + n_v \frac{L_u - L_l}{n_v + n_c} \tag{4}$$

$$L_{uc} = L_u \tag{5}$$

$$c = (L_u - L_l)\frac{\vec{X}_{i,l} - L_{lc}}{L_{uc} - L_{lc}} - L_u \tag{6}$$

Where L_l and L_u are the search space limits in the variant of the PSO algorithm, particularly in this work, DSRegPSO.

Finally, for every particle, each level m_j is added to $\vec{M}_i$—the mathematical model for prediction or regression represented with the position of the particle.

The transformation proposed required for the DSRegPSOP is in Algorithm 1.

Algorithm 1 Transformation $TS : X \in R^{n\times d} \mapsto M \in S^{n\times ds}$

Data: $X, n_l, n_g, n_o, n_s, L_l, L_u, L_v, L_o$
Result: M
$d = n_l \cdot \left[n_g \cdot (n_o + n_v) + (n_g - 1)\right]$;
$M = []$;
for $i \leftarrow 1$ **to** n **do**
  $\vec{M}_i = []$;
  for $j \leftarrow 1$ **to** n_l **do**
    $m_j = []$;
    $\vec{M}_{o,i} = \frac{\vec{X}_i - L_l}{L_u - L_l} \cdot (n_o - 1)$;
    $\vec{M}_{v,i} = \frac{\vec{X}_i - L_l}{L_u - L_l} \cdot (n_v - 1 + n_c)$;
    for $k \leftarrow 1$ **to** n_g **do**
      if $k = 0$ **then**
        $m_j \leftarrow add('(')$;
      end
      else if $m_j\,[k-1] \neq$ '(' **then**
        $m_j \leftarrow add('(')$;
      end
      for $l \leftarrow k \cdot \frac{\frac{d}{n_l}}{n_g} + j \cdot \frac{d}{n_l}$ **to** $(k+1) \cdot \frac{\frac{d}{n_l}}{n_g} + j \cdot \frac{d}{n_l}$ **do**
        if $(l+1) + mod(k,2) + mod(j,2) = 0$ **then**
          $m_j \leftarrow add\left(L_o\left(\vec{M}_{o,i=l}\right)\right)$;
        end
        else then:
          if $\vec{M}_{v=i,l} < n_v$ **then**
            $m_j \leftarrow add\left(L_v\left(\vec{M}_{v,i=l}\right)\right)$;
          end
          else then:
            $L_{lc} = L_l + n_v \frac{L_u - L_l}{n_v + n_c}$;
            $L_{uc} = L_u$;
            $c = (L_u - L_l)\frac{\vec{X}_{i,l} - L_{lc}}{L_{uc} - L_{lc}} - L_u$;
            $m_j \leftarrow add(c)$;
          end
        end
      end
      $m_j \leftarrow add(')')$;
    end
    $\vec{M}_i \leftarrow add\left(m_j\right)$;
  end
  $M \leftarrow add\left(\vec{M}_i\right)$;
end

2.2. Particle Swarm Optimization with Inertial Coefficient (GPSO)

The PSO algorithm randomly initializes a numeric array of positions $\vec{X}_i \in R^{1\times d}$ with $i = 1, \ldots, n$ where n is the number of particles and d is the search space dimensionality. Every iteration $\vec{X}_i$ is updated depending on the speed array $\vec{V}_i \in R^{1\times d}$. The updating equations depend on the PSO version. In GPSO, the most utilized PSO with an inertial coefficient was proposed in 1998. The updating equations for speed and position are in Equations (7) and (8), respectively [39].

$$\vec{V}_i = w \cdot \vec{V}_i + c_1 \cdot \vec{R}_1 \cdot \left(\vec{P}_G - \vec{X}_i\right) + c_2 \cdot \vec{R}_2 \left(\vec{P}_i - \vec{X}_i\right) \tag{7}$$

$$\vec{X}_i = \vec{X}_i + \vec{V}_i \tag{8}$$

where w is the inertial coefficient in ranges of $[0.9,\ 1.2]$ as in the original algorithm [29]. c_1 and c_2 are personal and social coefficients, and their values are typically set to 2. $\vec{P}_G$ is the global best position known, and $\vec{P}_i$ is the best position known for each particle. $\vec{R}_1$ and $\vec{R}_2$ are random arrays in ranges $[0, 1]$ with size $n \times d$. The position $\vec{X}_i$ is updated every iteration using Equation (8) while being constrained within the lower position limit (L_l) and the upper position limit (L_u) [33,39].

Through multiple iterations of updating $\vec{V}_i$ and $\vec{X}_i$ using Equations (7) and (8) depending on a cost function $f\left(\vec{X}_i\right)$, $\vec{P}_i$ is drawn towards $\vec{P}_G$. Algorithm 2 describes the implementation of PSO with an inertial coefficient as in [39].

Algorithm 2 GPSO

Data: $d, n, f\left(\vec{X}_i\right), L_l, L_u, LS_l, LS_u, c_1, c_2, w$
Result: $\vec{P}_G$
$\vec{X}_i = \vec{\Psi}_i \in [L_i, L_u]$;
$\vec{V}_i = 0$;
$C_i = f\left(\vec{X}_i\right)$;
$G_{P,i} = C_i$;
$G_B = min\left(G_{P,i}\right)$;
$\vec{P}_i = \vec{X}_i$;
$\vec{P}_G = \vec{X}_{argmin\left(f\left(\vec{X}_i\right)\right)}$;
$it = 0$;
while $k < k_{max}$ **or** $G_B \leq C_d$ **do**
  $\vec{V}_i = w \cdot \vec{V}_i + c_1 \cdot \vec{R}_1 \cdot \left(\vec{P}_G - \vec{X}_i\right) + c_2 \cdot \vec{R}_2\left(\vec{P}_i - \vec{X}_i\right)$;
  $\vec{V}_i = \vec{V}_i \in [LS_l, LS_u]$;
  $\vec{X}_i = \vec{X}_i + \vec{V}_i$;
  $\vec{X}_i = \vec{X}_i \in [L_l, L_u]$;
  $C_i = f\left(\vec{X}_i\right)$;
  for $i \leftarrow 1$ **to** n **do**
    if $G_{P,i} < C_i$ **then**
      $G_{P,i} = C_i$;
      $\vec{P}_i = \vec{X}_i$;
    end
  end
  $G_B = min\left(G_{P,i}\right)$;
  $\vec{P}_G = argmin\left(f\left(\vec{P}_i\right)\right)$;
end

2.3. Dynamical Sphere Regrouping Particle Swarm Optimization (DSRegPSO)

DSRegPSO algorithm is a variant of PSO, inspired by PSO by bird flocking or fish schooling. In both algorithms, birds search for the best food source (best cost known) in a defined area (search space). During the search, birds adjust their position by following the current best-known position or the one nearest to the food. However, DSRegPSO introduces the situation of birds exhausting food when there is no improvement in the global best cost, forcing particles to travel to other locations to explore the search space while changing their speed and inertial behavior (dynamical sphere behavior). Additionally, DSRegPSO considers the physical principle of conservation momentum when the particles cross the frontier of the search space, as detailed in [33].

The DSRegPSO starts by obtaining the ω with Equation (9), which distributes the maximum inertial momentum (M_{max}) across $\frac{\delta}{\delta_{max}}$ and $\frac{S_s}{S_{max}}$ that control the inertial effect [33].

$$\omega = \frac{M_{max}}{2} \tag{9}$$

The speed of DSRegPSO is in Equation (10). It has three components: inertial $\left(\left(\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}\right) \cdot \omega\right)$, social $\left(c_1 \cdot \vec{R}_1 \cdot \left(\vec{P}_G - \vec{X}_i\right)\right)$, and personal $\left(c_2 \cdot \vec{R}_2 \left(\vec{P}_i - \vec{X}_i\right)\right)$. The inertial component contains the dynamical sphere regrouping mechanism controlled by the hyper-sphere diameter δ and its expansion speed S_s. Social and personal components are the same as in the original PSO [33].

$$\vec{V}_i = \left[\left(\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}\right) \cdot \omega \cdot \vec{V}_i\right] + c_1 \cdot \vec{R}_1 \cdot \left(\vec{P}_G - \vec{X}_i\right) + c_2 \cdot \vec{R}_2 \cdot \left(\vec{P}_i - \vec{X}_i\right) \tag{10}$$

The speed limits in $\vec{V}_i$ change across iterations. LS_u is the upper speed limit, and LS_l is the lower speed limit. They are initialized depending on the initial upper speed limit (LS_{ui}). As in Equations (11)–(13) [33].

$$LS_{ui} = \lambda(L_l - L_u) \tag{11}$$

$$LSu = LS_{ui} \tag{12}$$

$$LS_l = -LS_u \tag{13}$$

where λ is a coefficient that allows for the modification of the starting speed limits depending on the size of the search space.

After obtaining $\vec{V}_i$ of the i particle and retaining them under its limits, Equation (14) updates its position [33].

$$\vec{X}_i = (1 - U_i) \cdot \left(\vec{X}_i + \vec{V}_i\right) + U_i \cdot \vec{\Psi}_i \tag{14}$$

$\vec{\Psi}_i$ is a random array with positions between the lower position limit (L_l) and the upper position limit (L_u). $\vec{\Psi}_i$ changes the position of particles too near to $\vec{P}_G$, based on $U_i(\delta_i)$ that depends on the hyper-sphere diameter δ, as in Equation (15).

$$U_i(\delta_i) = \begin{cases} 1, & \delta_i \leq \delta \\ 0, & \delta_i > \delta \end{cases} \tag{15}$$

The distance δ_i is determined with Equation (16).

$$\delta_i = \Sigma \left| \vec{X}_{i,d} - \vec{P}_G \right| \tag{16}$$

The sphere diameter δ is updated using Equation (17) if there is insufficient improvement (ζ) in global best cost (G_B), i.e., $|G_B(k) - G_B(k-1)| \leq \zeta \cdot G_B(k)$. Similarly, S_s obtains its value according to Equation (18). When there is enough improvement, $\delta = \delta_{min}$ and $S_s = S_{min}$ to maximize exploitation.

$$\delta = \begin{cases} \delta + \delta_{max} \cdot S_s, & \delta < \delta_{max} \\ \delta_{min}, & else \end{cases} \tag{17}$$

$$S_s = \begin{cases} S_s + S_{min}, & S_s < S_{max} \\ S_{min}, & else \end{cases} \tag{18}$$

The conservation of momentum of DSRegPSO in Equation (19) maximizes exploration by returning particles to different positions in an opposite direction if the speed takes them beyond the search space boundaries.

$$\vec{X}_{i,d} = \begin{cases} max\left(L_l, L_u + \left(L_u - \vec{X}_{i,d}\right)\right), & \vec{X}_{i,d} > L_u \\ min\left(L_u, L_l + \left(L_l - \vec{X}_{i,d}\right)\right), & \vec{X}_{i,d} < L_l \end{cases} \tag{19}$$

δ_{max} and δ_{min} vary depending on $\vec{P}_G$, progressively reducing the size of the search space around the best position across iterations. The speed of refinement is controlled with fd_{max} and fd_{min}, as in Equations (20) and (21).

$$\delta_{max} = max\left(\vec{P}_G\right) \cdot fd_{max} \tag{20}$$

$$\delta_{min} = min\left(\vec{P}_G\right) \cdot fd_{min} \tag{21}$$

Finally, Equations (22) and (23) update the speed limits for each iteration while reaching the desired number of iterations ($k < k_{max}$) or a desired cost value ($G_B \leq C_d$).

$$LS_u = \frac{\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}}{2} \cdot LS_{ui} \tag{22}$$

$$LS_l = -LS_u \tag{23}$$

The DSRegPSO implementation is presented in Algorithm 3.

Algorithm 3 DSRegPSO

Data: $d, n, f\left(\vec{X}_i\right), L_l, L_u, c_1, c_2, M_{max}, \lambda, fd_{max}, fd_{min}, S_{max}, S_{min}, \zeta$

Result: $\vec{P}_G$

$LS_{ui} = \lambda(L_l - L_u)$;
$LSu = LS_{ui}$;
$LS_l = -LS_u$;
$\omega = \frac{M_{max}}{2}$;
$\vec{X}_i = \vec{\Psi}_i \in [L_i, L_u]$;
$\vec{V}_i = 0$;
$C_i = f\left(\vec{X}_i\right)$;
$G_{P,i} = C_i$;
$\vec{P}_i = \vec{X}_i$;
$G_B = min\left(G_{P,i}\right)$;
$\vec{P}_G = argmin\left(f\left(\vec{X}_i\right)\right)$;
$S_s = S_{min}$;
$\delta_{max} = max\left(\vec{P}_G\right) \cdot fd_{max}$;
$\delta_{min} = min\left(\vec{P}_G\right) \cdot fd_{min}$;
$\delta = \delta_{min}$;
$it = 0$;

Algorithm 3 *Cont.*

while $k < k_{max}$ **or** $G_B \leq C_d$ **do**
$\vec{V}_i = \left[\left(\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}\right) \cdot \omega \cdot \vec{V}_i\right] + c_1 \cdot \vec{R}_1 \cdot \left(\vec{P}_G - \vec{X}_i\right) + c_2 \cdot \vec{R}_2 \cdot \left(\vec{P}_i - \vec{X}_i\right)$;
$\vec{V}_i = \vec{V}_i \in [LS_l, LS_u]$;
$\delta_i = \Sigma\left|\vec{X}_{i,d} - \vec{P}_G\right|$;
$U_i(\delta_i) = \begin{cases} 1, & \delta_i \leq \delta \\ 0, & \delta_i > \delta \end{cases}$;
$\vec{X}_i = (1 - U_i) \cdot \left(\vec{X}_i + \vec{V}_i\right) + U_i \cdot \vec{\Psi}_i$;
$\vec{X}_{i,d} = \begin{cases} max\left(L_l, L_u + \left(L_u - \vec{X}_{i,d}\right)\right), & \vec{X}_{i,d} > L_u \\ min\left(L_u, L_l + \left(L_l - \vec{X}_{i,d}\right)\right), & \vec{X}_{i,d} < L_l \end{cases}$;
$\vec{X}_i = \vec{X}_i \in [L_l, L_u]$;
$C_i = f\left(\vec{X}_i\right)$;
for $i \leftarrow 1$ **to** n **do**
if $C_i < G_{P,i}$ **then**
$G_{P,i} = C_i$;
$\vec{P}_i = \vec{X}_i$;
end
if $G_{P,i} < G_B$ **then**
$G_B = G_{P,i}$;
$\vec{P}_G = \vec{P}_i$;
end
end
$it = it + 1$;
if $|G_B(k) - G_B(k-1)| \leq \zeta \cdot G_B(k)$ **then**
if $\delta < \delta_{max}$ **then**
$\delta = \delta + \delta_{max} \cdot S_s$;
else
$\delta = \delta_{min}$;
if $S_s \geq S_{max}$ **then**
$S_s = S_{min}$;
else
$S_s = S_s + S_{min}$;
end
end
else
$\delta = \delta_{min}$;
$S_s = S_{min}$;
end
$\delta_{max} = max\left(\vec{P}_G\right) \cdot fd_{max}$;
$\delta_{min} = min\left(\vec{P}_G\right) \cdot fd_{min}$;
$LS_u = \frac{\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}}{2}$;
$LS_l = -LS_u$;
end

2.4. Dynamical Sphere Regrouping Particle Swarm Optimization Programming (DSRegPSOP)

Our proposed algorithm, DSRegPSOP, incorporates stagnation avoidance from DSReg-PSO by utilizing dynamical sphere regrouping and conservation momentum mechanisms. These mechanisms work together to allow constant reinvigoration of the swarm. Dynamical sphere regrouping updates the positions of particles too near to the global best-known position or inside a sphere of diameter delta with the center as the global best-known position, and conservation momentum returns particles in the opposite direction when they travel away from the search space, as detailed in [33].

DSRegPSOP initializes with the same parameters required by DSRegPSO but is adapted to optimize the transformation of each particle with $TS : \boldsymbol{X} \in R^{n \times d} \mapsto \boldsymbol{M} \in S^{n \times ds}$ as detailed in Algorithm 1 to produce a desired equation. Thus, there are two parameters specifically defined for DSRegPSOP.

d or number of dimensions in DSRegPSO is the first parameter determined with Equation (1) according to TS in Section 2.1.

The second parameter is the cost function, which for DSRegPSOP is $f\left(\vec{M}_i\right)$ depending on $\vec{M}_i$ obtained with $TS\{X\}$. $f\left(\vec{M}_i\right)$ is the Mean Square Error (MSE) between y and $\hat{y}$, the desired and obtained responses of the generated equations, respectively. Moreover, we add two penalties to MSE as in Equation (24).

$$\mathrm{f}\left(\vec{M}_i\right) = \frac{1}{N}\sum_{n=1}^{N}(y - \hat{y})^2 + p_1 + p_2 \tag{24}$$

p_1 penalizes with gain γ_1 the absolute distance between the maximum and minimum values of $\hat{y}$, like in Equation (25).

$$p_1 = \frac{\gamma_1}{|min(\hat{y}) - max(\hat{y})|} \tag{25}$$

p_2 penalizes with gain γ_2 the number of times that a previous register is not in the current register. This forces the model to use previous registers in each register, like in Equation (26).

$$p_2 = \gamma_2 \cdot \left|(n_l - 1) - \sum_{lv=0}^{n_l-1} \Phi_{lv}\right| \tag{26}$$

Φ_{lv} verifies for each register if the previous ones are in it, like in Equation (27).

$$\Phi_{lv} = \begin{cases} 1, & m_1, \ldots, lv-1 \subset m_{lv} \\ 0, & else \end{cases} \tag{27}$$

Algorithm 4 shows the implementation of the proposed DSRegPSOP to obtain an optimized model for regression, classification, or prediction.

Algorithm 4 DSRegPSOP

Data: $TS, f\left(\vec{M}_i\right), L_l, L_u, c_1, c_2, M_{max}, \lambda, fd_{max}, fd_{min}, S_{max}, S_{min}, \zeta$
Result: $\vec{M}_i$
$LS_{ui} = \lambda(L_l - L_u)$;
$LSu = LS_{ui}$;
$LS_l = -LS_u$;
$\omega = \frac{M_{max}}{2}$;
$\vec{X}_i = \vec{\Psi}_i \in [L_i, L_u]$;
$\vec{V}_i = 0$;
$M = TS\{X\}$;
$C_i = f\left(\vec{M}_i\right)$;
$G_{P,i} = C_i$;
$\vec{P}_i = \vec{X}_i$;
$G_B = min\left(G_{P,i}\right)$;
$\vec{P}_G = argmin\left(f\left(\vec{X}_i\right)\right)$;
$S_s = S_{min}$;
$\delta_{max} = max\left(\vec{P}_G\right) \cdot fd_{max}$;
$\delta_{min} = min\left(\vec{P}_G\right) \cdot fd_{min}$;
$\delta = \delta_{min}$;
$it = 0$;

Algorithm 4 *Cont.*

while $k < k_{max}$ **or** $G_B \leq C_d$ **do**
$\vec{V}_i = \left[\left(\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}\right) \cdot \omega \cdot \vec{V}_i\right] + c_1 \cdot \vec{R}_1 \cdot \left(\vec{P}_G - \vec{X}_i\right) + c_2 \cdot \vec{R}_2 \cdot \left(\vec{P}_i - \vec{X}_i\right)$;
$\vec{V}_i = \vec{V}_i \in [LS_l, LS_u]$;
$\delta_i = \Sigma\left|\vec{X}_{i,d} - \vec{P}_G\right|$;
$U_i(\delta_i) = \begin{cases} 1, & \delta_i \leq \delta \\ 0, & \delta_i > \delta \end{cases}$;
$\vec{X}_i = (1 - U_i) \cdot \left(\vec{X}_i + \vec{V}_i\right) + U_i \cdot \vec{\Psi}_i$;
$\vec{X}_{i,d} = \begin{cases} max\left(L_l, L_u + \left(L_u - \vec{X}_{i,d}\right)\right), & \vec{X}_{i,d} > L_u \\ min\left(L_u, L_l + \left(L_l - \vec{X}_{i,d}\right)\right), & \vec{X}_{i,d} < L_l \end{cases}$;
$\vec{X}_i = \vec{X}_i \in [L_l, L_u]$;
$M = TS\{X\}$;
$C_i = f\left(\vec{M}_i\right)$;
for $i \leftarrow 1$ **to** n **do**
if $C_i < G_{P,i}$ **then**
$G_{P,i} = C_i$;
$\vec{P}_i = \vec{X}_i$;
end
if $G_{P,i} < G_B$ **then**
$G_B = G_{P,i}$;
$\vec{P}_G = \vec{P}_i$;
end
end
$it = it + 1$;
if $|G_B(k) - G_B(k-1)| \leq \zeta \cdot G_B(k)$ **then**
if $\delta < \delta_{max}$ **then**
$\delta = \delta + \delta_{max} \cdot S_s$;
else
$\delta = \delta_{min}$;
if $S_s \geq S_{max}$ **then**
$S_s = S_{min}$;
else
$S_s = S_s + S_{min}$;
end
end
else
$\delta = \delta_{min}$;
$S_s = S_{min}$;
end
$\delta_{max} = max\left(\vec{P}_G\right) \cdot fd_{max}$;
$\delta_{min} = min\left(\vec{P}_G\right) \cdot fd_{min}$;
$LS_u = \frac{\frac{\delta}{\delta_{max}} + \frac{S_s}{S_{max}}}{2}$;
$LS_l = -LS_u$;
end

2.5. Evaluation and Comparison Benchmarks

We evaluate the performance of DSRegPSOP using datasets. Many researchers have assessed their algorithms using datasets for symbolic regression tasks, but there is currently no standardized framework for benchmarking these problems [46]. Some examples of repositories with datasets for regression and classification tasks are UC machine learning [47], Kaggle [48], and PMLB [49]. Alternatively, SRbench is a framework designed for benchmarking symbolic regression algorithms. It's important to note that this framework

isn't a standardized benchmark but rather a tool for testing algorithms using datasets already published in other repositories [50].

In this work, we evaluate the DSRegPSOP with datasets of real-world problems. The datasets chosen for testing DSRegPSOP were selected based on their popularity of benchmarking symbolic regression and the existence of documented results in the literature for the purpose of comparison. The datasets used are

- Airfoil: A 1989 dataset of NASA for prediction of noise in aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel. The dataset contains the features of frequency, angle of attack, chord length, free stream velocity, suction side displacement thickness, and scaled sound pressure level; this last is the variable to predict [51,52].
- Concrete: A 1998 dataset that allows for the prediction of concrete strength based on amounts of cement, slag from blast furnaces, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, age of the concrete in days (1–365), and concrete compressive strength; this last one is the target variable [53].
- Energy: A dataset of 2012 for the prediction of cooling and heating requirements for buildings with input features including relative compactness, surface area, wall area, roof area, overall height, orientation, glazing area, and glazing area distribution [54,55]. In this work, we focus on the prediction of cooling load for the third dataset test.
- Yacht: Is a 2013 dataset to predict the hydrodynamic performance of sailing yachts based on the longitudinal position of the center of buoyancy, prismatic coefficient, length-displacement ratio, beam-draught ratio, length-beam ratio, Froude number, residuary resistance per unit weight of displacement, this last one is the target variable [56].
- Geographical Origin of Music: A 2014 dataset with audio features extracted from 1059. The goal is to predict the geographical origin of the track. The input features are 68 numerical values extracted from the music with the Music Analysis, Retrieval, and Synthesis for Audio Signals framework (MARSYAS) [57,58]. In this work, we focus only on the latitude related to music, as detailed in [59].
- Tecator: A 2006 dataset with data of the spectrum recorded on a food and feed analyzer working in the wavelength range 850–1050 nm by the near-infrared transmission principle. Each sample contains meat with different moisture, fat, and protein contents [60].

Additionally, we value DSRegPSOP with a proposed benchmark that includes 10 tests; each one uses known equations with two-dimensional functions that generate three-dimensional surfaces. Moreover, all of them have two dependent variables, $x \in X$ and $y \in Y$, that produce a surface in $Z \rightarrow X \times Y$. Thus, the DSRegPSO must optimize a model for the target surface generated in Z. The data is not split to optimize the equation and represent the target surface using the same information points as in numerical optimization benchmarks. In this work, we are dealing with three-dimensional surfaces, but the functions can exhibit increased dimensionality, like in numerical optimization benchmarks [61].

Analyzing three-dimensional surfaces enables the assessment of DSRegPSOP with visually represented data in a three-dimensional format. Nonetheless, we are still using the most common regression metrics for comparative purposes, including Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Square Error (MSE), and R-square (R^2) [33]. Moreover, as in other benchmarks, we evaluate the quality of results by adding noise to the cost function and the position of particles.

The 10 functions in the proposed benchmark for testing DSRegPSOP are

Function 1: This is the simplest test in the benchmark and includes a second-degree polynomial with a trigonometric function. The dependent variables in ranges $x = [-5, 5]$ and $y = [-2, 2]$ with 20 points per variable, i.e., 400 points for $X \times Y$. The different ranges for dependent variables simplify distinguishing between them. Equation (28) shows function 1, and Figure 4 shows its surface.

$$f_1(x,y) = x^2 + 3{\cdot}y - cos(x) \tag{28}$$

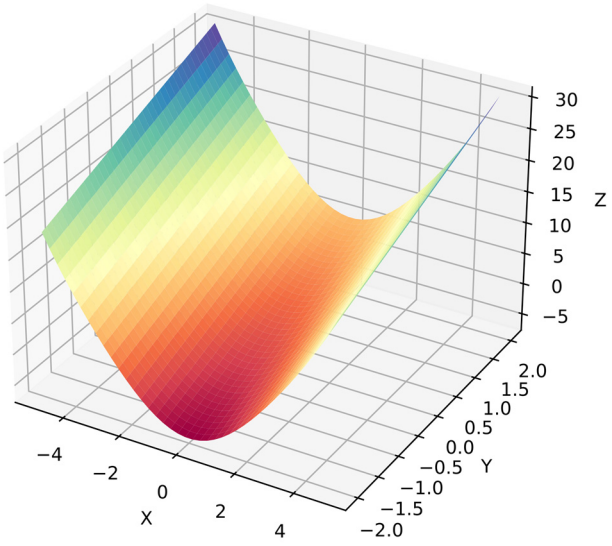


Figure 4. Surface of function 1.

Function 2: A sphere function is common in numerical optimization benchmarks [62] but it is a more complex problem in symbolic regression because the algorithm must optimize the equation to obtain the target surface. Now the algorithm cannot distinguish between dependent variables because they are in similar ranges $x = y = [-5, 5]$ with 40 points per variable, i.e., 1600 points for $X \times Y$. Equation (29) shows function 2, and Figure 5 shows its surface.

$$f_2(x,y) = \sqrt{(8 - x^2 - y^2)} + 1 \tag{29}$$

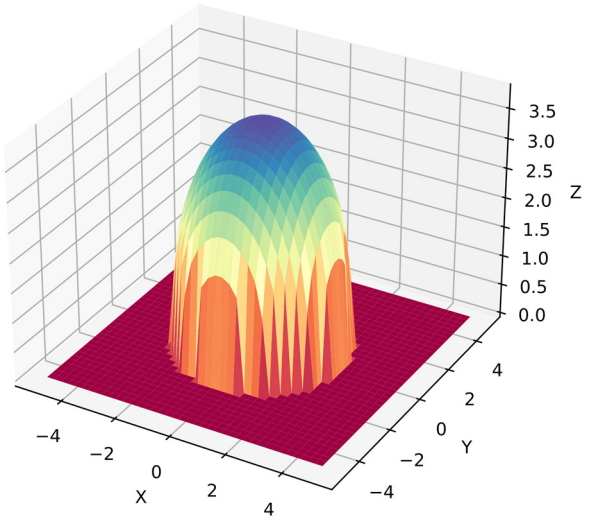


Figure 5. Surface of function 2.

Function 3: The Rastrigin function is also common in numerical optimization benchmarks [62], but it is a more complex problem in symbolic regression because the algorithm must optimize the equation for the target surface. Moreover, the algorithm cannot distinguish dependent variables since they are in similar ranges $x = y = [-5, 5]$ with 40 points

per variable, i.e., 1600 points for $X \times Y$. Equation (30) shows function 3, and Figure 6 shows its surface.

$$f_3(x, y) = 20 + x^2 - 10 \cdot cos(2 \cdot \pi \cdot x) + y^2 - 10 \cdot cos(2 \cdot \pi \cdot y) \tag{30}$$

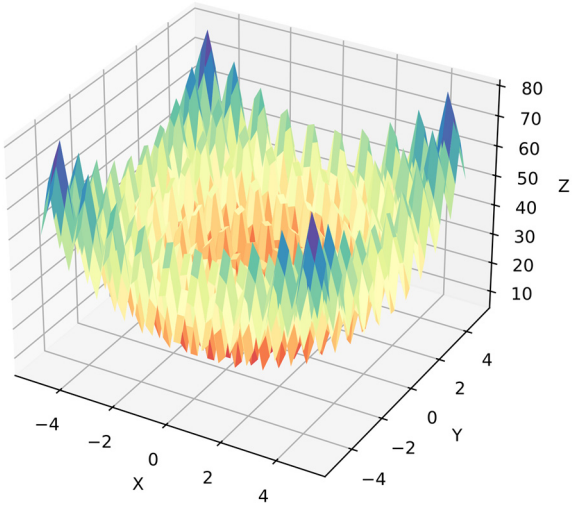


Figure 6. Surface of function 3.

Function 4: The Ackley function is also common in numerical optimization benchmarks [62], but it is a more complex problem in symbolic regression because the algorithm must optimize the equation for the target surface. Moreover, the algorithm cannot distinguish dependent variables since they are in similar ranges $x = y = [-5, 5]$ with 40 points per variable, i.e., 1600 points for $X \times Y$. Equation (31) shows function 4, and Figure 7 shows its surface.

$$f_4(x, y) = -20 \cdot e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5 \cdot (cos(2 \cdot \pi \cdot x) + cos(2 \cdot \pi \cdot y))} + e + 20 \tag{31}$$

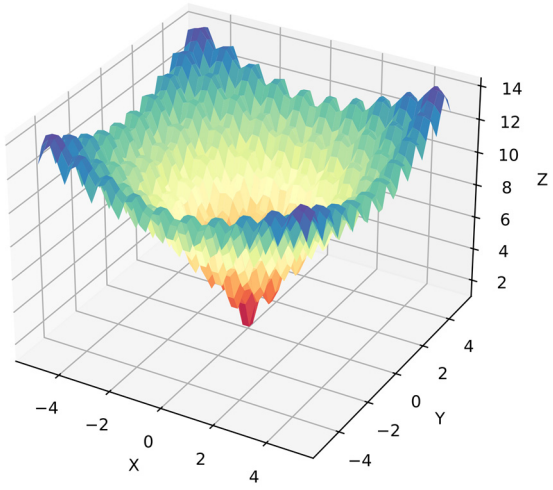


Figure 7. Surface of function 4.

Function 5: This is a trigonometric function with multiple dependent variables in its argument. Furthermore, the algorithm cannot distinguish dependent variables, and this time, the search space is bigger compared with previous functions. Moreover, there are fewer information points since the dependent variables are in ranges $x = y = [-10, 10]$ with 20 points per variable, i.e., 400 points for $X \times Y$. Equation (32) shows function 5, and Figure 8 shows its surface.

$$f_5(x, y) = 4 \cdot x \cdot sin(x + y) + 1 \tag{32}$$

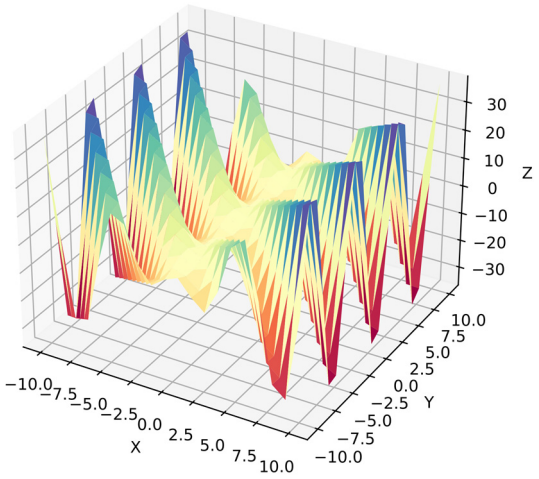


Figure 8. Surface of function 5.

Function 6: This is an exponential function with multiple dependent variables in its argument. Again, the algorithm cannot distinguish dependent variables. However, the search space is smaller in ranges $x = y = [-2, 2]$ because the exponential function produces bigger values, and there is less information than in the simpler functions since there are 20 points per variable, i.e., 400 points for $X \times Y$. Equation (33) shows function 6, and Figure 9 shows its surface.

$$f_6(x, y) = x \cdot e^{-x^2-y^2} + 1 \tag{33}$$

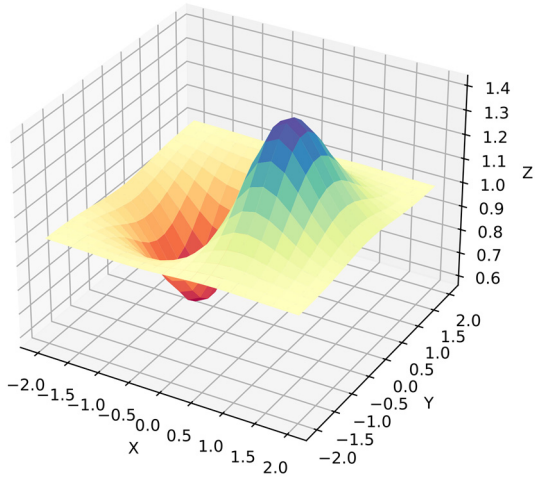


Figure 9. Surface of function 6.

Function 7: This is a hyperbolic paraboloid function with fractions and quadratic terms for each dependent variable. Again, the search space is bigger than in the simpler functions, and the algorithm cannot distinguish between dependent variables in ranges $x = y = [-10, 10]$. Similarly, there is less information than in the simpler functions since there are 20 points per variable, i.e., 400 points for $X \times Y$. Equation (34) shows function 7, and Figure 10 shows its surface.

$$f_7(x, y) = \frac{y^2}{35} - \frac{x^2}{20} + 6 \tag{34}$$

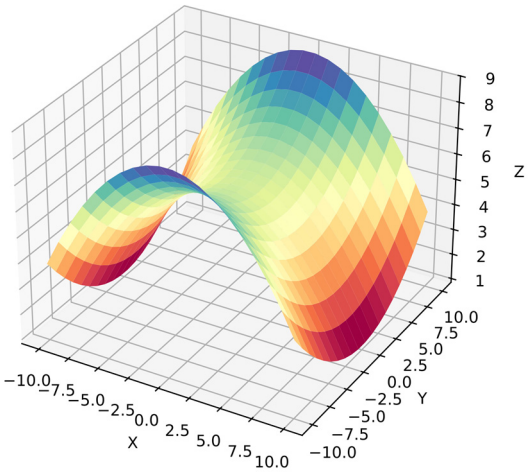


Figure 10. Surface of function 7.

Function 8: The Rosenbrock function has sharp peaks and regions with values near zero. It is also common in numerical optimization benchmarks [62], but it is a more complex problem in symbolic regression because the algorithm must optimize the equation for the target surface. Moreover, the algorithm cannot distinguish dependent variables since they are in similar ranges $x = y = [-10, 10]$. The search space has less information than in simpler functions with 20 points per variable, i.e., 400 points for $X \times Y$. Equation (35) shows function 8, and Figure 11 shows its surface.

$$f_8(x, y) = (1 - x)^2 + 100 \cdot \left(y - x^2\right)^2 \tag{35}$$

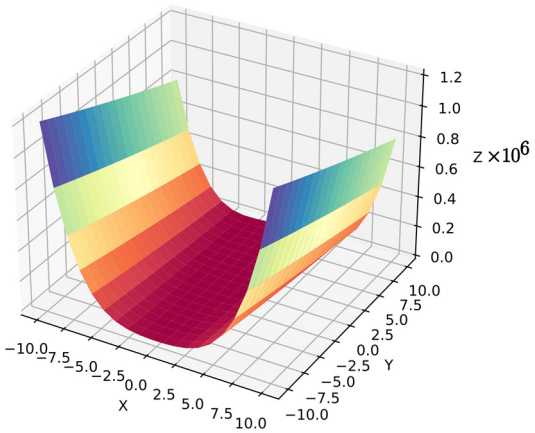


Figure 11. Surface of function 8.

Function 9: The Beale function has sharp peaks and regions with values near zero. The Beale function is also common in numerical optimization benchmarks [62], but it is a more complex problem in symbolic regression because the algorithm must optimize the equation for the target surface. Moreover, the algorithm cannot distinguish dependent variables since they are in similar ranges $x = y = [-10, 10]$. The search space has less information than simpler functions with 20 points per variable, i.e., 400 points in $X \times Y$. Equation (36) shows function 9, and Figure 12 shows its surface.

$$f_9(x, y) = (1.5 - x + x \cdot y)^2 + \left(2.25 - x + x \cdot y^2\right)^2 + \left(2.625 - x + x^3\right)^2 \tag{36}$$

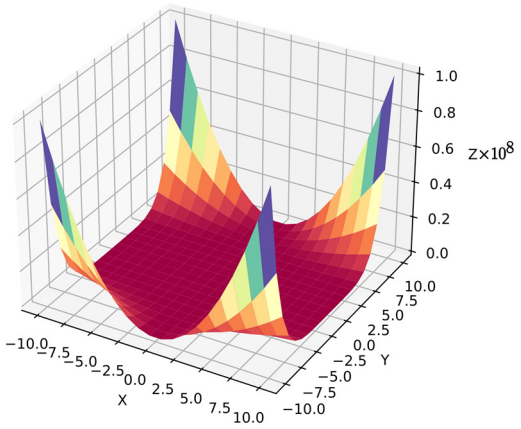


Figure 12. Surface of function 9.

Function 10: The Goldstein-Price function has sharp peaks and several local minimal regions. Goldstein-Price function is also common in numerical optimization benchmarks [62], but it is a more complex problem in symbolic regression because the algorithm must optimize the equation for the target surface. Moreover, the algorithm cannot distinguish dependent variables since they are in similar ranges $x = y = [-10, 10]$. The search space has less information than simpler functions with 20 points per variable, i.e., 400 points for $X \times Y$. Equation (37) shows function 10, and Figure 13 shows its surface.

$$f_{10}(x, y) = 1 + (x + y + 1)^2 \cdot \left(19 - 14 \cdot x + 3 \cdot x^2 - 14 \cdot y + 6 \cdot x \cdot y + 3 \cdot y^2\right) \tag{37}$$

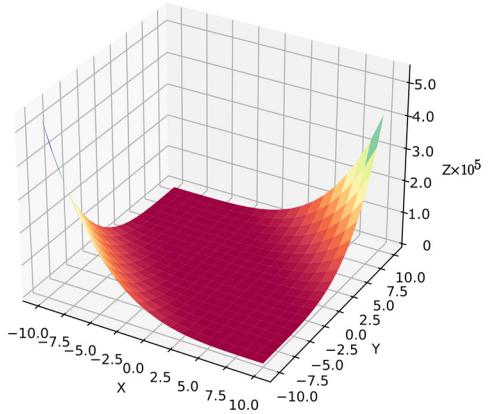


Figure 13. Surface of function 10.

3. Results and Discussion

The results and experiments were carried out on a computer running Microsoft Windows 11 Pro with a 10.0.22631-core 12th Gen Intel(R) Core (TM) i5-12400F processor, operating at 2500 Mhz. The system was equipped with 6 main cores, 12 logical cores, and 64.0 GB of RAM, and Python version 3.9.10 was used for all computations, without utilizing parallel computing.

3.1. Results in Benchmark with Three-Dimensional Functions

3.1.1. Experimental Setup of Benchmark in Three-Dimensional Functions

All the functions in this benchmark have the same considerations:

- The maximum number of function evaluations for each test is 2×10^7, but it could be less during hyperparameter tuning due to the next point.
- During hyperparameter tunning, the test is stopped, and the result is recorded if best cost value reaches 1×10^{-10}.

- The list of variables is
  $L_v = [b, x, y, e, \pi]$.
- The list of operators is
  $L_o = \left[\div, sin(\ ), cos(\ ), (,), \sqrt{\ }, \wedge, \Sigma, -, +, \times\right]$.
- The DSRegPSOP static parameters for all the functions are
  $n = 200, TS, f\left(\vec{M}_i\right), c_1 = c_2 = 2.00, M_{max} = 1.1, \lambda = 1.00$

$$S_{max} = 4.00 \times 10^{-2}, S_{min} = 1.00 \times 10^{-2}, \zeta = 1.00 \times 10^{-3}$$

- The parameters that are dependent on the optimized function are

$$L_u, L_l, fd_{max}, fd_{min}$$

- The fd_{max} and fd_{min} parameters are adapted so that $1 \times 10^{-2} < \delta_{max} < 1 \times 10^{-1}$ and $1 \times 10^{-12} < \delta_{min} < 1 \times 10^{-10}$ in the first iteration.
- The L_u and L_v are specified according to the search space limits in the test function.
- The n_l, n_g, and n_o parameters are obtained with exhaustive search exploring the 64 combinations produced with: $n_l = [1, 2, 3, 4]$, $n_g = [1, 2, 3, 4]$, and $n_o = [2, 3, 4, 5]$.
- There is no data split since the goal of this test is to optimize the surface as much as possible with the information points in the target surface.
  After finding the best configuration with an exhaustive search, the algorithm trains again but without stopping if the cost reaches 1×10^{-10} value.
- After finding the best configuration with an exhaustive search, the algorithm trains again but adds 0–25% noise in the cost function and position of particles to test the robustness of the algorithm.

3.1.2. Results Function 1

The best solution found in an exhaustive search for function 1 was with $n_g = 2$, $n_o = 4$ and $n_l = 2$ obtaining the best metrics $R^2 = 1.00 \times 10^0$, $f\left(\vec{M}_i\right) = 2.65 \times 10^{-12}$, MSE $= 2.05 \times 10^{-30}$, MAPE $= 1.37 \times 10^{-16}$, and MAE $= 7.59 \times 10^{-16}$, which support that DSRegPSOP successfully optimized the target surface. The top 10 configurations are in Table 1, and Figure 14 shows the parallel plot with the results of all the configurations used.

Table 1. Top 10 exhaustive search results testing function 1.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	2	4	2	7.59×10^{-16}	1.37×10^{-16}	2.05×10^{-30}	2.65×10^{-12}	1.00×10^{0}	00:01:37
2	3	5	2	1.41×10^{-12}	8.21×10^{-13}	2.93×10^{-24}	2.65×10^{-12}	1.00×10^{0}	01:15:01
3	1	5	2	3.41×10^{-13}	2.01×10^{-13}	1.16×10^{-25}	2.65×10^{-12}	1.00×10^{0}	00:06:37
4	1	4	2	9.16×10^{-16}	1.00×10^{-16}	3.22×10^{-30}	2.65×10^{-12}	1.00×10^{0}	00:03:29
5	2	3	2	4.53×10^{-13}	2.60×10^{-13}	2.49×10^{-25}	2.65×10^{-12}	1.00×10^{0}	00:08:14
6	2	2	2	7.82×10^{-16}	1.28×10^{-16}	2.15×10^{-30}	2.65×10^{-12}	1.00×10^{0}	00:03:00
7	3	2	2	6.82×10^{-13}	4.02×10^{-13}	4.65×10^{-25}	2.65×10^{-12}	1.00×10^{0}	00:15:47
8	1	5	1	4.45×10^{-16}	5.89×10^{-17}	1.26×10^{-30}	2.65×10^{-12}	1.00×10^{0}	00:01:57
9	3	3	1	1.01×10^{-15}	1.40×10^{-16}	3.56×10^{-30}	2.65×10^{-12}	1.00×10^{0}	00:05:25
10	2	4	3	7.20×10^{-16}	1.44×10^{-16}	1.86×10^{-30}	2.65×10^{-12}	1.00×10^{0}	00:00:59

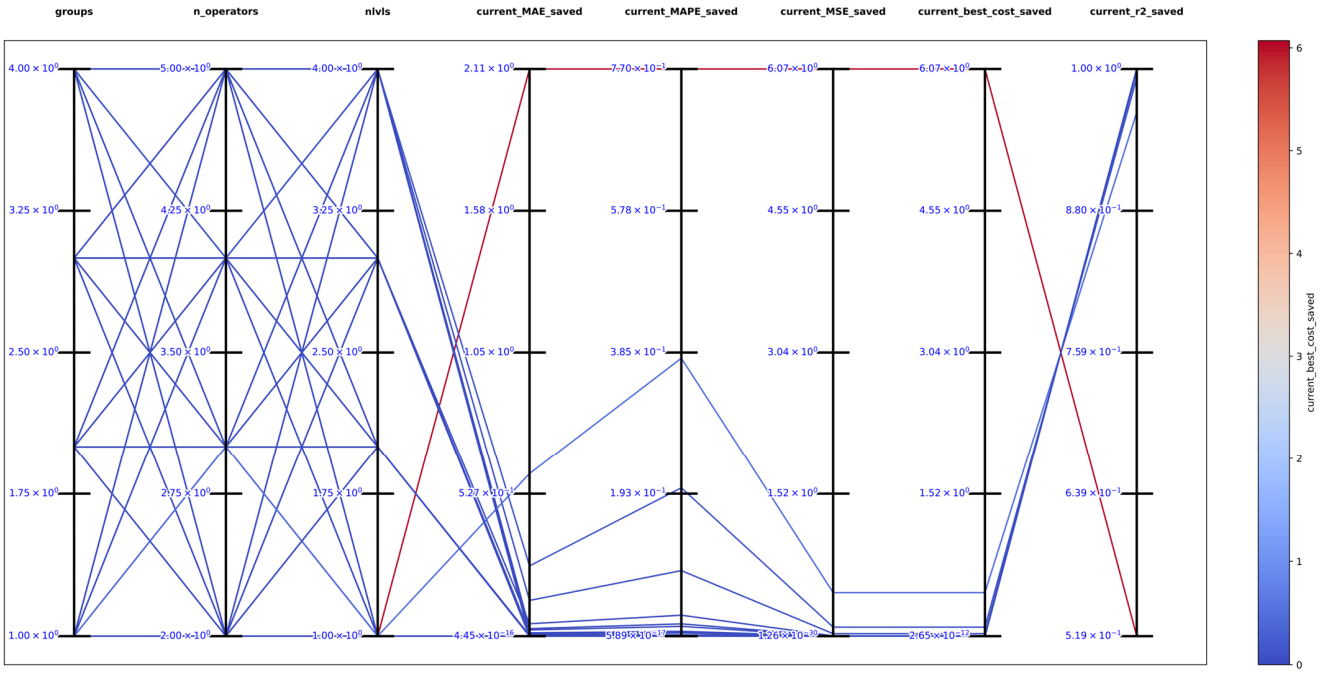


Figure 14. Parallel plot of hyperparameters for exhaustive search in function 1.

Figure 15 shows the comparison of the best obtained and target surface, including the heatmap and principal component analysis, reduced to three dimensions to show the positions of particles in the last iteration. Furthermore, the particles could still improve since the DSRegPSOP algorithm never converges and still explores the search space as DSRegPSO does.

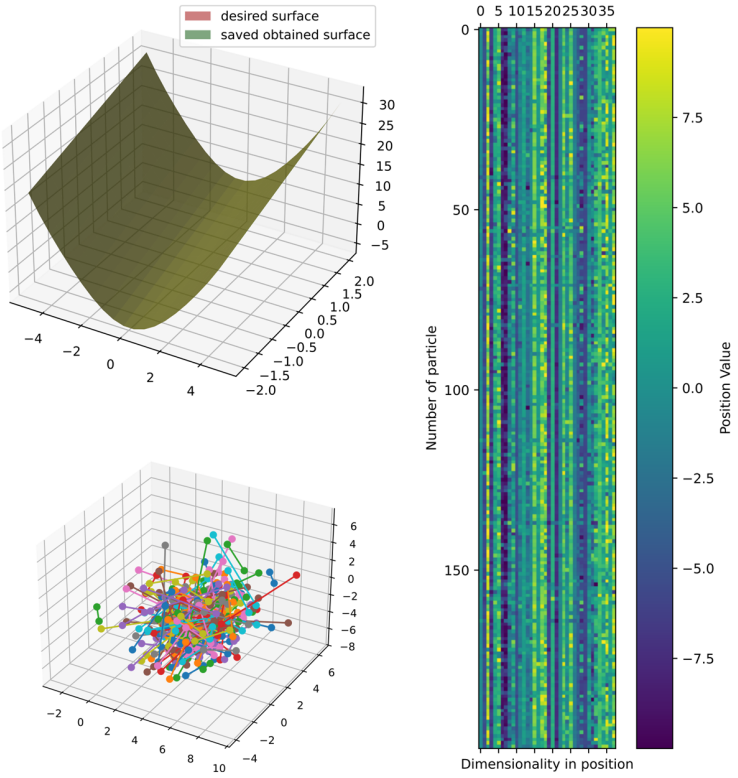


Figure 15. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 1.

Additionally, we assess DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% in function 1. The results in Table 2 show a minimal variation in best cost and R^2. Moreover, the minimal variation with noise is also evident in the convergence diagram in Figure 16.

Table 2. Effects of noise in position and cost evaluation for function 1.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	2.50×10^{-1}	0.00×10^{0}	6.24×10^{-11}	1.00×10^{0}
2	0.00×10^{0}	1.50×10^{-1}	2.65×10^{-12}	1.00×10^{0}
3	0.00×10^{0}	2.50×10^{-1}	2.65×10^{-12}	1.00×10^{0}
4	5.00×10^{-2}	0.00×10^{0}	2.73×10^{-12}	1.00×10^{0}
5	0.00×10^{0}	0.00×10^{0}	2.65×10^{-12}	1.00×10^{0}
6	1.50×10^{-1}	0.00×10^{0}	3.36×10^{-12}	1.00×10^{0}
7	0.00×10^{0}	0.00×10^{0}	2.65×10^{-12}	1.00×10^{0}
8	0.00×10^{0}	1.00×10^{-1}	2.65×10^{-12}	1.00×10^{0}
9	0.00×10^{0}	5.00×10^{-2}	2.65×10^{-12}	1.00×10^{0}
10	1.00×10^{-1}	0.00×10^{0}	7.97×10^{-11}	1.00×10^{0}

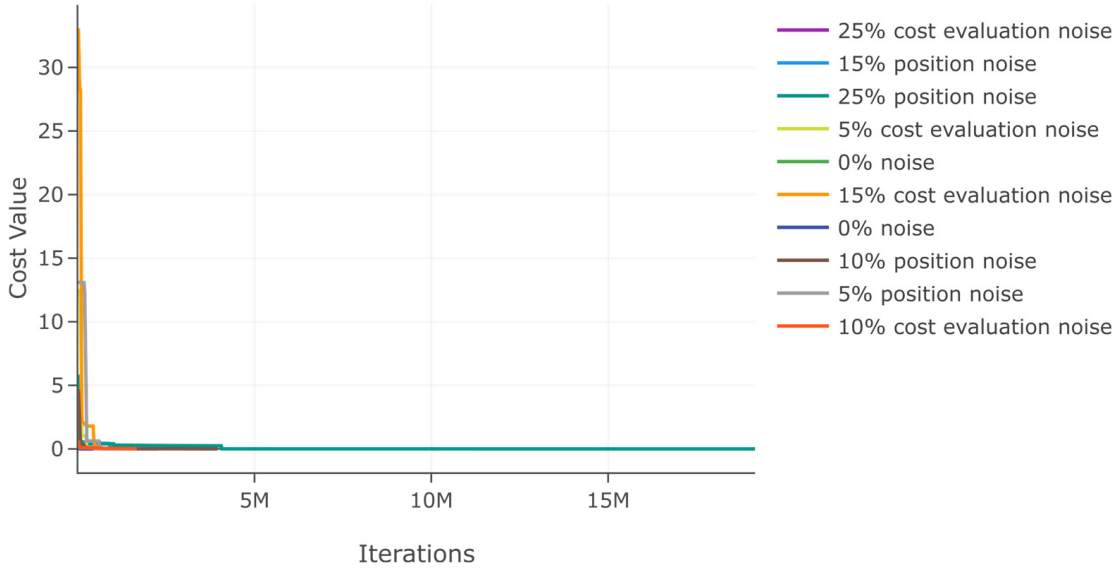


Figure 16. Effects of noise in position and cost evaluation for function 1.

3.1.3. Results Function 2

The best solution in an exhaustive search for function 2 was obtained with $n_g = 1$, $n_o = 4$ and $n_l = 3$ obtaining the best metrics $R^2 = 1.00 \times 10^{0}$, $f\left(\vec{M}_i\right) = 3.55 \times 10^{-4}$, MSE $= 3.55 \times 10^{-4}$, MAPE $= 3.16 \times 10^{-3}$, and MAE $= 3.40 \times 10^{-1}$, which support that DSRegPSOP successfully optimized the target surface. The top 10 configurations are in Table 3, and Figure 17 shows the parallel plot with the results of all the configurations.

Table 3. Top 10 exhaustive search results testing function 2.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	1	4	3	3.40×10^{-1}	3.16×10^{-3}	3.55×10^{-4}	3.55×10^{-4}	1.00×10^{0}	06:45:00
2	2	4	4	6.16×10^{-1}	1.78×10^{13}	1.03×10^{-2}	1.03×10^{-2}	9.81×10^{-1}	07:46:53
3	4	2	3	1.79×10^{-1}	1.85×10^{13}	1.15×10^{-2}	1.15×10^{-2}	9.80×10^{-1}	15:30:56
4	2	5	1	5.59×10^{-1}	3.17×10^{13}	1.25×10^{-2}	1.25×10^{-2}	9.87×10^{-1}	19:22:58
5	1	3	2	4.21×10^{-2}	3.12×10^{13}	1.25×10^{-2}	1.25×10^{-2}	9.88×10^{-1}	08:34:41
6	2	3	2	5.74×10^{-1}	3.87×10^{13}	1.27×10^{-2}	1.27×10^{-2}	9.36×10^{-1}	09:09:51
7	3	5	1	7.59×10^{-1}	3.11×10^{13}	1.48×10^{-2}	1.48×10^{-2}	9.77×10^{-1}	14:58:36
8	3	5	4	3.57×10^{-1}	2.20×10^{-2}	2.02×10^{-2}	2.02×10^{-2}	9.83×10^{-1}	06:33:45
9	4	3	3	1.81×10^{-1}	1.22×10^{14}	2.12×10^{-2}	2.12×10^{-2}	5.31×10^{-1}	04:44:04
10	3	2	4	7.27×10^{-1}	6.18×10^{13}	2.91×10^{-2}	2.91×10^{-2}	9.78×10^{-1}	04:58:08

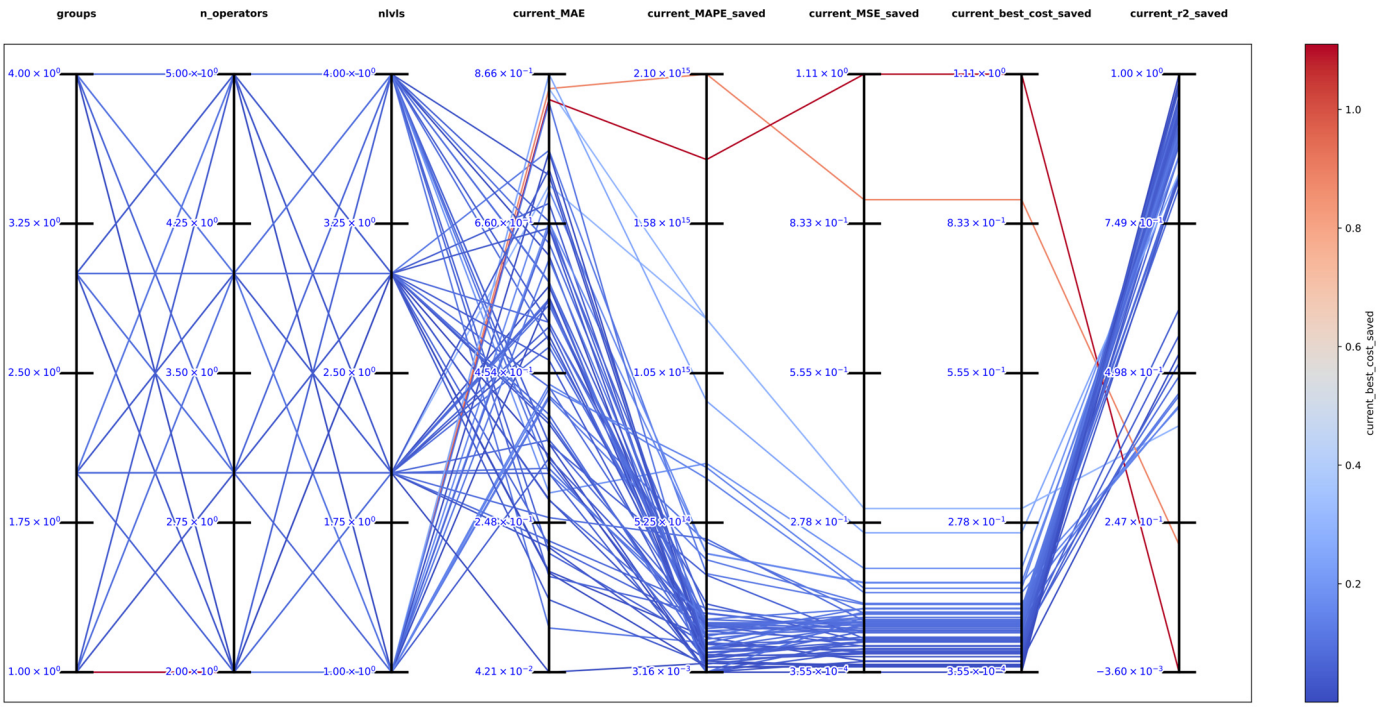


Figure 17. Parallel plot of hyperparameters for exhaustive search in function 2.

Figure 18 shows the best model obtained compared with the target surface; the heatmap and principal component analysis were reduced to three dimensions to show the positions of particles. However, the best cost could still improve since the position of particles in the DSRegPSOP algorithm never converges due to the dynamical sphere regrouping behavior inherited from the DSRegPSO, which reinvigorates the swarm by assigning a new position $\vec{X}_i$ to the particles positioned inside the sphere limited by the diameter δ_i according to Equation (14). Thus, DSRegPSOP remains exploring the search space, avoiding premature convergence.

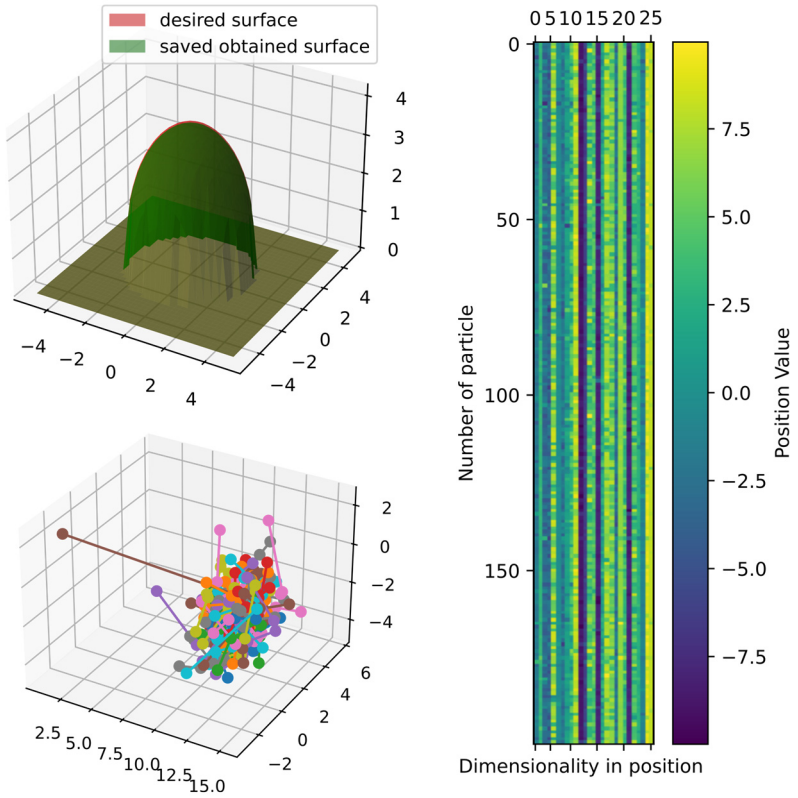


Figure 18. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 2.

Similarly, we evaluate DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% for function 2. The results in Table 4 show that the variation with noise is bigger than in function 1, but the variation is minimal if the noise is applied to the cost value. From the convergence diagram for noise evaluations in Figure 19, it is also evident that there is a bigger affectation with noise in position.

Table 4. Effects of noise in position and cost evaluation for function 2.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	0.00×10^{0}	0.00×10^{0}	3.55×10^{-4}	1.00×10^{0}
2	0.00×10^{0}	0.00×10^{0}	3.55×10^{-4}	1.00×10^{0}
3	2.50×10^{-1}	0.00×10^{0}	1.84×10^{-3}	9.99×10^{-1}
4	5.00×10^{-2}	0.00×10^{0}	3.01×10^{-3}	9.97×10^{-1}
5	1.00×10^{-1}	0.00×10^{0}	1.25×10^{-2}	9.91×10^{-1}
6	1.50×10^{-1}	0.00×10^{0}	1.67×10^{-2}	9.86×10^{-1}
7	0.00×10^{0}	5.00×10^{-2}	2.03×10^{-1}	4.20×10^{-1}
8	0.00×10^{0}	1.50×10^{-1}	4.11×10^{-1}	2.85×10^{-1}
9	0.00×10^{0}	1.00×10^{-1}	5.71×10^{-1}	6.76×10^{-1}
10	0.00×10^{0}	2.50×10^{-1}	6.75×10^{-1}	2.43×10^{-1}

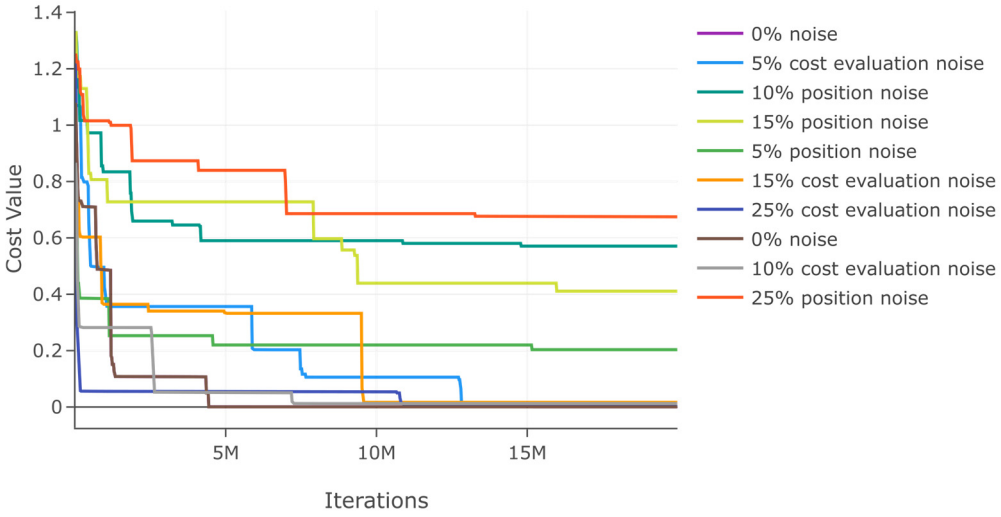


Figure 19. Effects of noise in position and cost evaluation for function 2.

3.1.4. Results Function 3

The best solution in an exhaustive search for function 3 was obtained with $n_g = 2$, $n_o = 5$ and $n_l = 3$ obtaining the best metrics $R^2 = 6.21 \times 10^{-1}$, $f\left(\vec{M}_i\right) = 3.91 \times 10^1$, MSE $= 3.91 \times 10^1$, MAPE $= 2.04 \times 10^{-1}$, and MAE $= 5.68 \times 10^0$, which show that DSRegPSOP optimized the target surface, but the error is considerable based on R^2.

The top 10 configurations are in Table 5, and Figure 20 shows the parallel plot with the results of all the configurations. Figure 21 also shows that the heatmap and principal component analysis are reduced to three dimensions to show the positions of particles in the last iteration.

Table 5. Top 10 exhaustive search results testing function 3.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	2	5	3	5.68×10^0	2.04×10^{-1}	3.91×10^1	3.91×10^1	6.21×10^{-1}	16:18:45
2	3	5	1	5.81×10^0	2.06×10^{-1}	4.12×10^1	4.12×10^1	6.01×10^{-1}	09:25:19
3	3	5	2	6.02×10^0	2.11×10^{-1}	4.56×10^1	4.56×10^1	5.57×10^{-1}	06:28:07
4	4	2	3	5.96×10^0	2.05×10^{-1}	4.64×10^1	4.64×10^1	5.50×10^{-1}	23:26:15
5	3	4	2	6.26×10^0	2.23×10^{-1}	4.75×10^1	4.75×10^1	5.39×10^{-1}	18:00:00
6	4	3	2	6.11×10^0	2.02×10^{-1}	4.89×10^1	4.89×10^1	5.26×10^{-1}	15:28:07
7	2	5	1	6.45×10^0	2.29×10^{-1}	5.08×10^1	5.08×10^1	5.07×10^{-1}	19:38:26
8	4	3	4	6.34×10^0	2.20×10^{-1}	5.09×10^1	5.09×10^1	5.06×10^{-1}	02:03:45
9	1	4	3	6.46×10^0	2.26×10^{-1}	5.12×10^1	5.12×10^1	5.04×10^{-1}	17:54:23
10	3	2	1	6.46×10^0	2.26×10^{-1}	5.12×10^1	5.12×10^1	5.04×10^{-1}	18:05:38

Despite the error in R^2 for function 3, based on the comparison with the target and obtained surfaces in Figure 21, we found that the two surfaces are similar in the prediction, but the obtained surface cannot mimic the natural noise in function 3. However, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

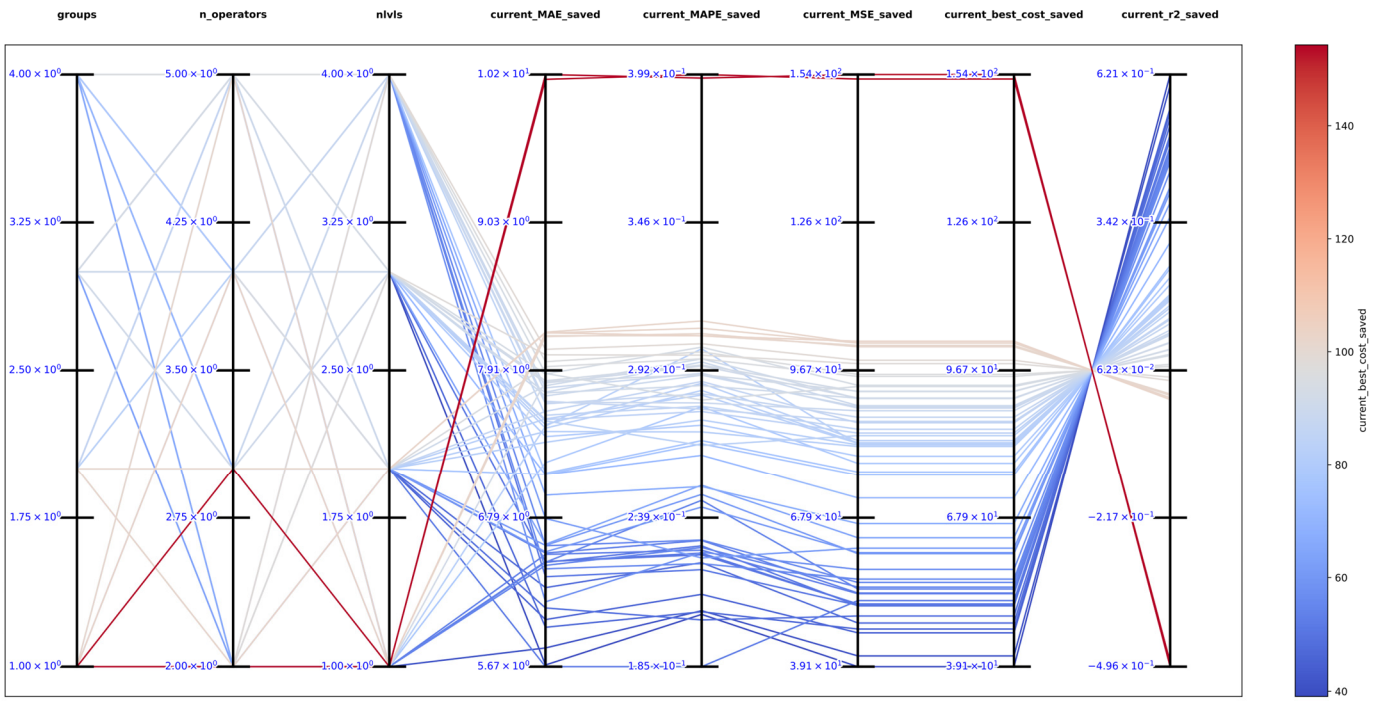


Figure 20. Parallel plot of hyperparameters for exhaustive search in function 3.

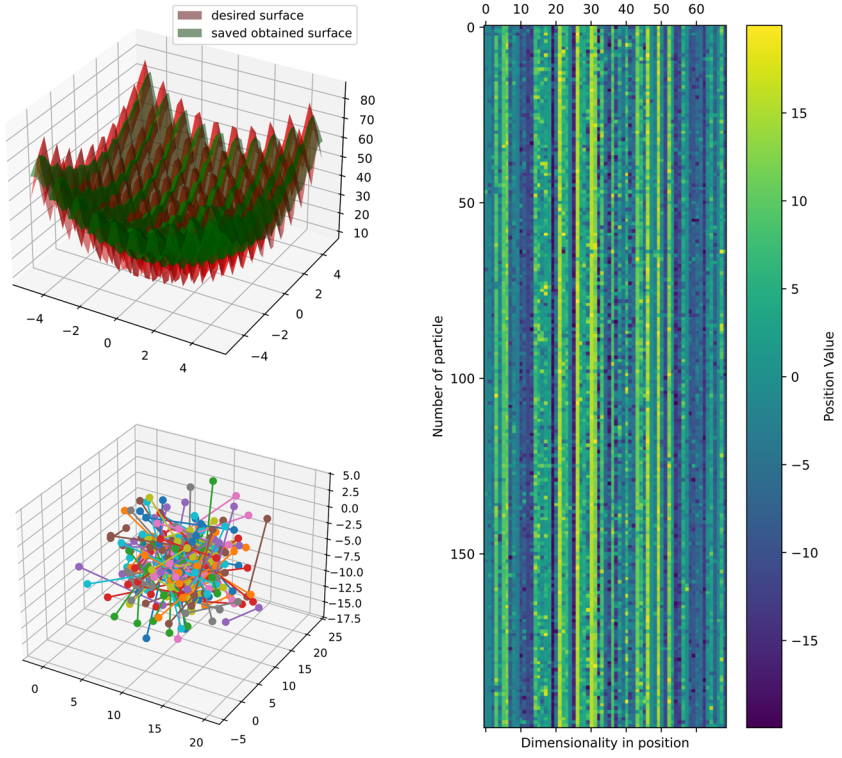


Figure 21. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 3.

Similarly, we evaluate DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% for function 3. The results in Table 6 show that the variation with noise is bigger than in previous functions, and this time, the noise also affects the cost value.

The convergence diagram for noise evaluations in Figure 22 shows that the error increases with both noises, but the noise in position affects more than in cost value.

Table 6. Effects of noise in position and cost evaluation for function 3.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	2.50×10^{-1}	0.00×10^{0}	6.43×10^{1}	1.69×10^{-1}
2	0.00×10^{0}	1.50×10^{-1}	1.02×10^{2}	7.03×10^{-3}
3	0.00×10^{0}	2.50×10^{-1}	1.03×10^{2}	4.92×10^{-4}
4	5.00×10^{-2}	0.00×10^{0}	7.55×10^{1}	2.30×10^{-1}
5	0.00×10^{0}	0.00×10^{0}	3.91×10^{1}	6.21×10^{-1}
6	0.00×10^{0}	1.00×10^{-1}	1.01×10^{2}	1.83×10^{-2}
7	1.50×10^{-1}	0.00×10^{0}	6.23×10^{1}	2.90×10^{-1}
8	1.00×10^{-1}	0.00×10^{0}	6.70×10^{1}	2.84×10^{-1}
9	0.00×10^{0}	0.00×10^{0}	3.91×10^{1}	6.21×10^{-1}
10	0.00×10^{0}	5.00×10^{-2}	9.31×10^{1}	9.76×10^{-2}

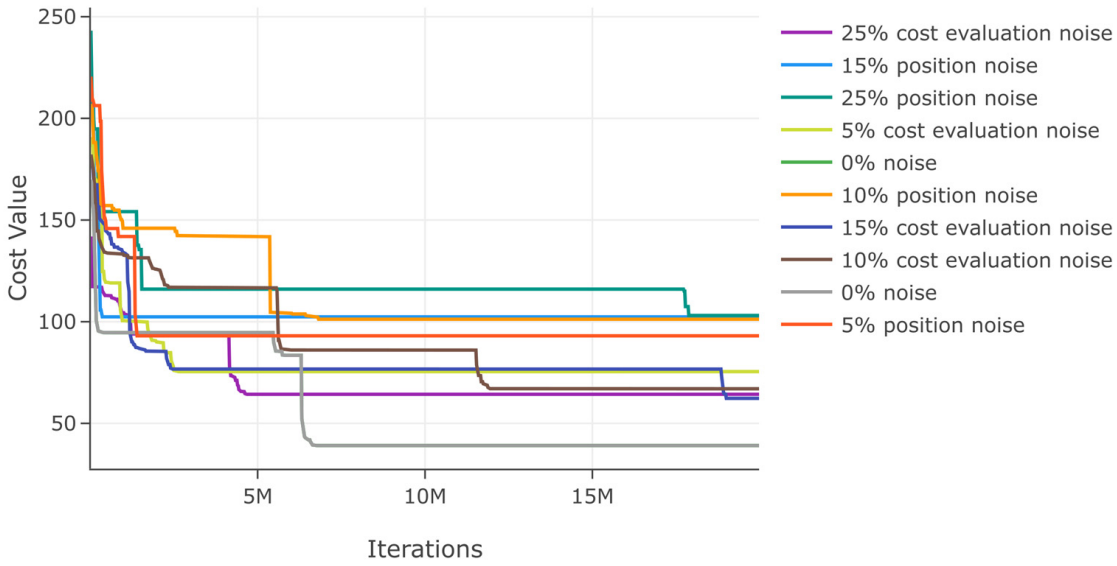


Figure 22. Effects of noise in position and cost evaluation for function 3.

3.1.5. Results Function 4

The best solution in an exhaustive search for function 4 was obtained with $n_g = 4$, $n_o = 2$ and $n_l = 2$ obtaining the best metrics $R^2 = 8.04 \times 10^{-1}$, $f\left(\vec{M}_i\right) = 3.76 \times 10^{-1}$, MSE $= 3.76 \times 10^{-1}$, MAPE $= 5.90 \times 10^{-2}$, and MAE $= 4.86 \times 10^{-1}$, which show that DSRegPSOP optimized the target surface, but the error is considerable based on R^2.

The top 10 configurations are in Table 7 and Figure 23 shows the parallel plot with the results of all the configurations tested. Figure 23 also shows the heatmap and principal component analysis reduced to three dimensions to show the positions of particles in the last iteration.

Table 7. Top 10 exhaustive search results testing function 4.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	4	2	2	4.86×10^{-1}	5.90×10^{-2}	3.76×10^{-1}	3.76×10^{-1}	8.04×10^{-1}	06:42:11
2	4	5	3	4.86×10^{-1}	6.20×10^{-2}	3.82×10^{-1}	3.82×10^{-1}	8.09×10^{-1}	07:46:52
3	3	2	3	4.97×10^{-1}	5.86×10^{-2}	3.85×10^{-1}	3.85×10^{-1}	8.06×10^{-1}	01:52:30
4	3	2	4	5.03×10^{-1}	5.84×10^{-2}	3.93×10^{-1}	3.93×10^{-1}	7.93×10^{-1}	05:26:15
5	3	3	3	4.90×10^{-1}	6.45×10^{-2}	4.16×10^{-1}	4.16×10^{-1}	8.19×10^{-1}	18:33:45
6	3	2	2	5.25×10^{-1}	6.75×10^{-2}	4.27×10^{-1}	4.27×10^{-1}	8.01×10^{-1}	11:37:30
7	1	5	2	5.26×10^{-1}	6.36×10^{-2}	4.35×10^{-1}	4.35×10^{-1}	7.76×10^{-1}	06:15:28
8	3	5	3	5.26×10^{-1}	6.07×10^{-2}	4.54×10^{-1}	4.54×10^{-1}	7.73×10^{-1}	20:15:00
9	4	3	1	5.40×10^{-1}	6.79×10^{-2}	4.58×10^{-1}	4.58×10^{-1}	7.70×10^{-1}	08:23:26
10	2	3	3	5.59×10^{-1}	7.07×10^{-2}	4.73×10^{-1}	4.73×10^{-1}	7.86×10^{-1}	12:56:15

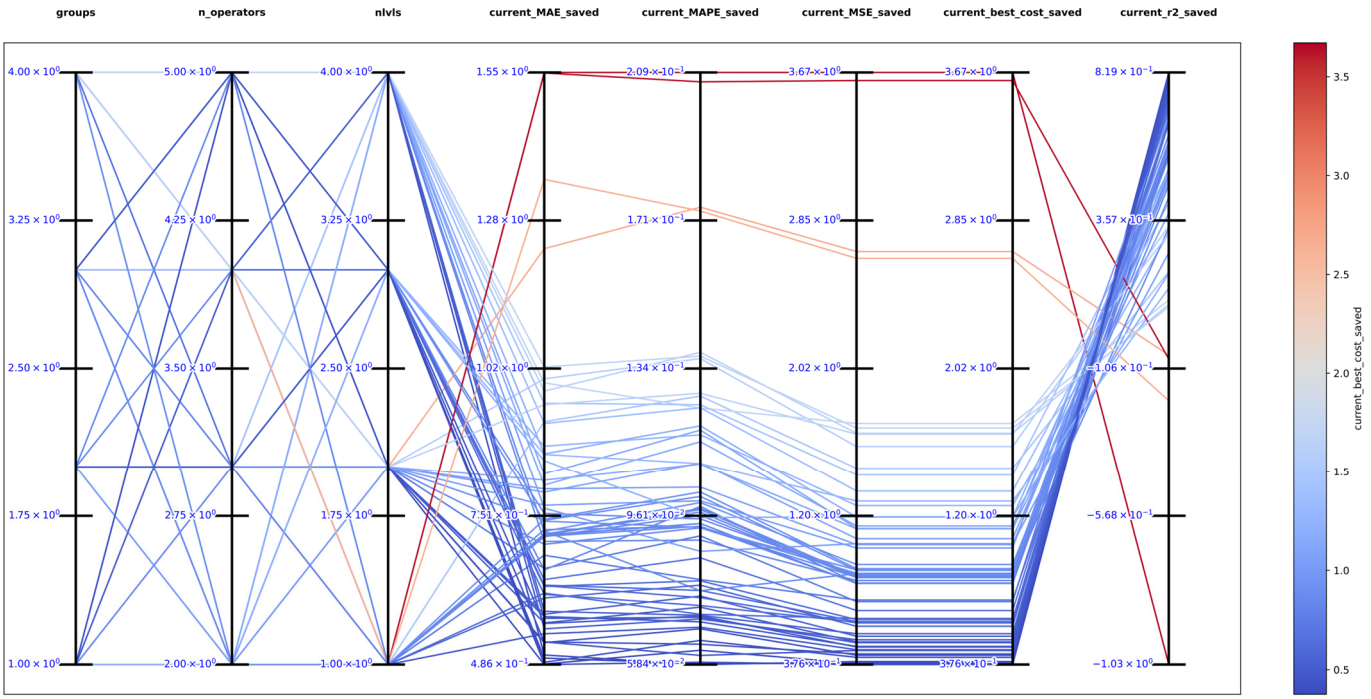


Figure 23. Parallel plot of hyperparameters for exhaustive search in function 4.

Despite the error in R^2 for function 4, based on the comparison with the target and obtained surfaces in Figure 24, we found that the two surfaces are similar in general prediction, but the obtained surface cannot mimic the natural noise in function 4. However, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

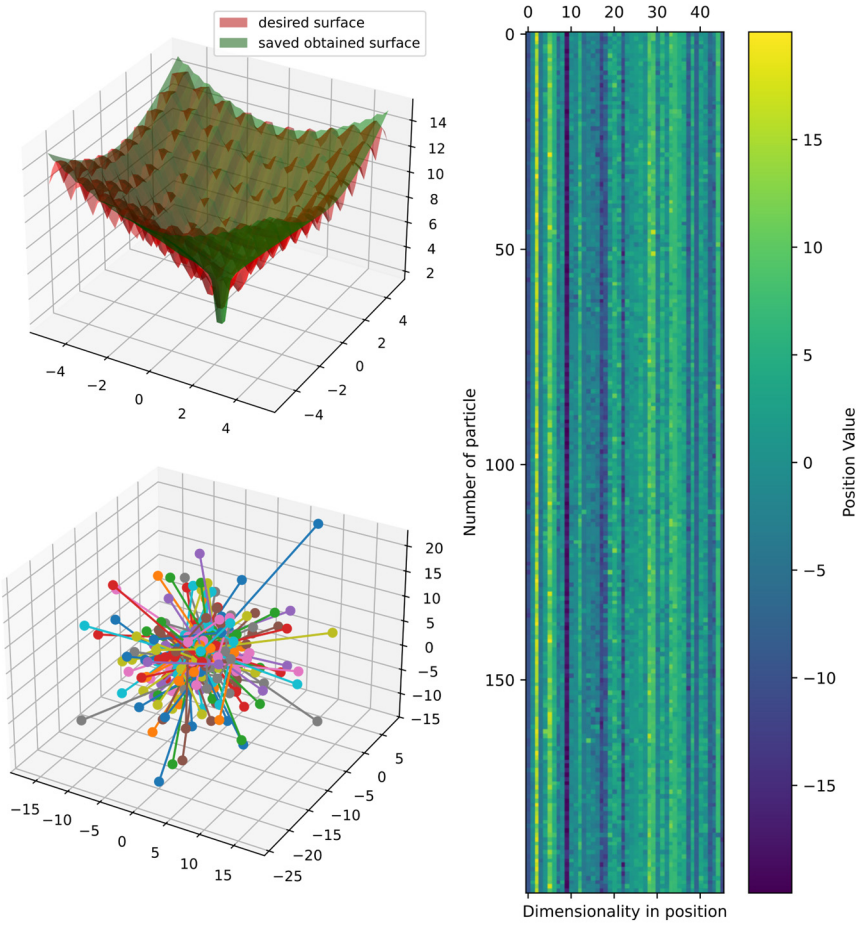


Figure 24. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 4.

Similarly, we assess DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% for function 4. The results in Table 8 show that the variation with noise is bigger than in functions 1 and 2 but lower than in 3. The convergence diagram for noise evaluations is in Figure 25 shows that there is affectation with both noises, but the noise in position affects more than in cost value.

Table 8. Effects of noise in position and cost evaluation for function 4.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	0.00×10^{0}	5.00×10^{-2}	6.44×10^{-1}	6.80×10^{-1}
2	1.00×10^{-1}	0.00×10^{0}	5.13×10^{-1}	6.56×10^{-1}
3	0.00×10^{0}	1.50×10^{-1}	2.49×10^{0}	-5.10×10^{-1}
4	0.00×10^{0}	0.00×10^{0}	3.76×10^{-1}	8.04×10^{-1}
5	0.00×10^{0}	1.00×10^{-1}	1.95×10^{0}	3.47×10^{-1}
6	1.50×10^{-1}	0.00×10^{0}	7.08×10^{-1}	4.55×10^{-1}
7	5.00×10^{-2}	0.00×10^{0}	1.28×10^{0}	5.93×10^{-1}
8	2.50×10^{-1}	0.00×10^{0}	7.26×10^{-1}	6.45×10^{-1}
9	0.00×10^{0}	0.00×10^{0}	3.76×10^{-1}	8.04×10^{-1}
10	0.00×10^{0}	2.50×10^{-1}	2.82×10^{0}	-4.40×10^{-1}

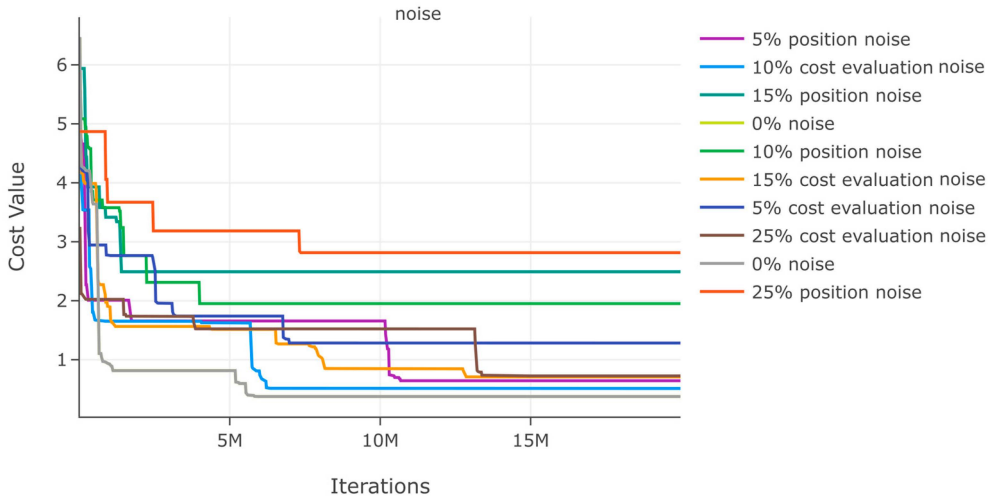


Figure 25. Effects of noise in position and cost evaluation for function 4.

3.1.6. Results Function 5

The best solution in an exhaustive search for function 5 was obtained with $n_g = 1$, $n_o = 2$ and $n_l = 3$ obtaining the best metrics $R^2 = 1.00 \times 10^0$, $f\left(\vec{M}_i\right) = 1.37 \times 10^{-12}$, MSE $= 6.13 \times 10^{-32}$, MAPE $= 1.23 \times 10^{-16}$, and MAE $= 1.32 \times 10^{-16}$, which show that DSRegPSOP successfully optimized the target surface based on R^2. The top 10 configurations are in Table 9, and Figure 26 shows the parallel plot with the results of all the configurations tested.

Table 9. Top 10 exhaustive search results testing function 5.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	1	2	3	1.32×10^{-16}	1.23×10^{-16}	6.13×10^{-32}	1.37×10^{-12}	1.00×10^{0}	19:24:37
2	3	3	4	2.14×10^{-7}	5.79×10^{-8}	8.49×10^{-14}	1.46×10^{-12}	1.00×10^{0}	18:09:51
3	4	4	1	3.47×10^{-7}	1.86×10^{-7}	1.35×10^{-13}	1.51×10^{-12}	1.00×10^{0}	15:57:27
4	1	3	4	5.84×10^{-7}	3.96×10^{-7}	5.14×10^{-13}	1.89×10^{-12}	1.00×10^{0}	12:36:18
5	1	5	1	1.37×10^{-6}	1.14×10^{-7}	3.60×10^{-12}	4.97×10^{-12}	1.00×10^{0}	02:09:14
6	2	5	2	2.47×10^{-6}	9.52×10^{-7}	8.85×10^{-12}	1.02×10^{-11}	1.00×10^{0}	00:47:17
7	2	5	1	2.61×10^{-6}	1.33×10^{-6}	9.33×10^{-12}	1.07×10^{-11}	1.00×10^{0}	21:55:01
8	2	3	3	2.42×10^{-6}	2.46×10^{-7}	1.10×10^{-11}	1.24×10^{-11}	1.00×10^{0}	12:40:31
9	1	3	2	2.43×10^{-6}	1.89×10^{-7}	1.11×10^{-11}	1.25×10^{-11}	1.00×10^{0}	12:57:09
10	1	2	2	2.75×10^{-6}	4.20×10^{-7}	1.30×10^{-11}	1.44×10^{-11}	1.00×10^{0}	02:31:10

Figure 27 shows the comparison of the best obtained and target surface, including the heatmap and principal component analysis, which are reduced to three dimensions to show the positions of particles in the last iteration. Furthermore, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

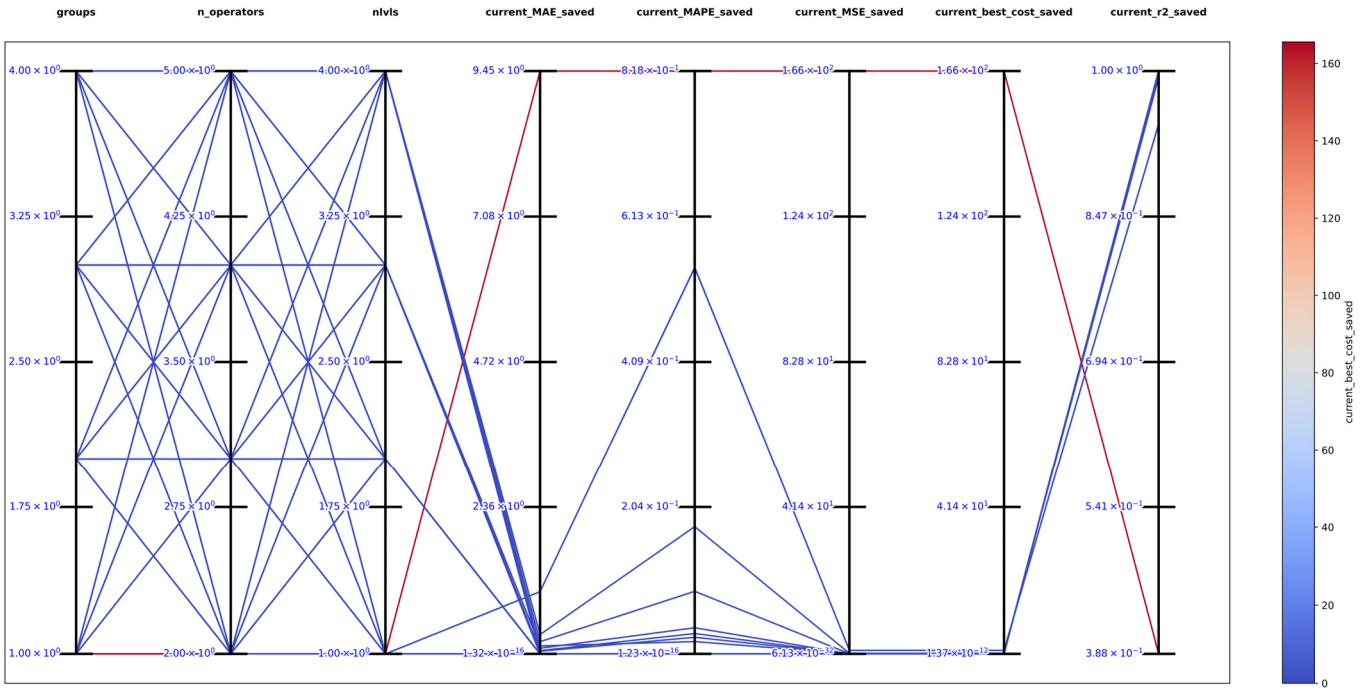


Figure 26. Parallel plot of hyperparameters for exhaustive search in function 5.

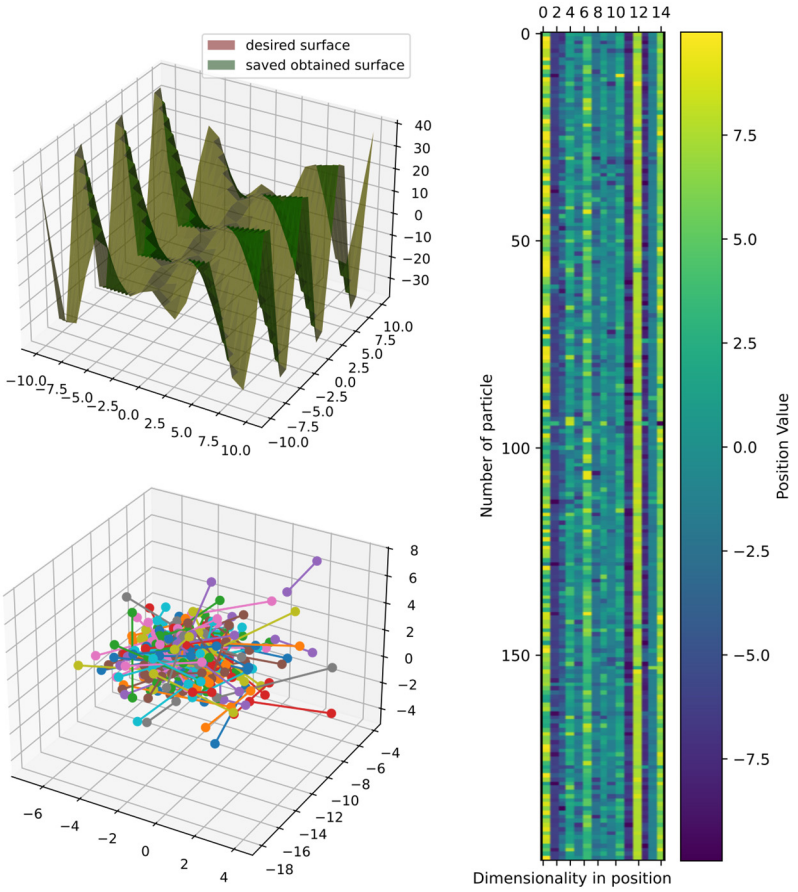


Figure 27. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 5.

Again, we evaluate DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% in function 5. The results in Table 10 show a minimal variation in best cost and R^2. Moreover, the minimal variation with noise is also evident in the convergence diagram in Figure 28.

Table 10. Effects of noise in position and cost evaluation for function 5.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	0.00×10^{0}	2.50×10^{-1}	1.37×10^{-12}	1.00×10^{0}
2	1.00×10^{-1}	0.00×10^{0}	7.49×10^{-11}	1.00×10^{0}
3	0.00×10^{0}	0.00×10^{0}	1.37×10^{-12}	1.00×10^{0}
4	0.00×10^{0}	5.00×10^{-2}	1.37×10^{-12}	1.00×10^{0}
5	0.00×10^{0}	1.00×10^{-1}	1.37×10^{-12}	1.00×10^{0}
6	0.00×10^{0}	1.50×10^{-1}	9.37×10^{-05}	1.00×10^{0}
7	1.50×10^{-1}	0.00×10^{0}	7.96×10^{-11}	1.00×10^{0}
8	0.00×10^{0}	0.00×10^{0}	1.37×10^{-12}	1.00×10^{0}
9	5.00×10^{-2}	0.00×10^{0}	3.19×10^{-11}	1.00×10^{0}
10	2.50×10^{-1}	0.00×10^{0}	9.44×10^{-12}	1.00×10^{0}

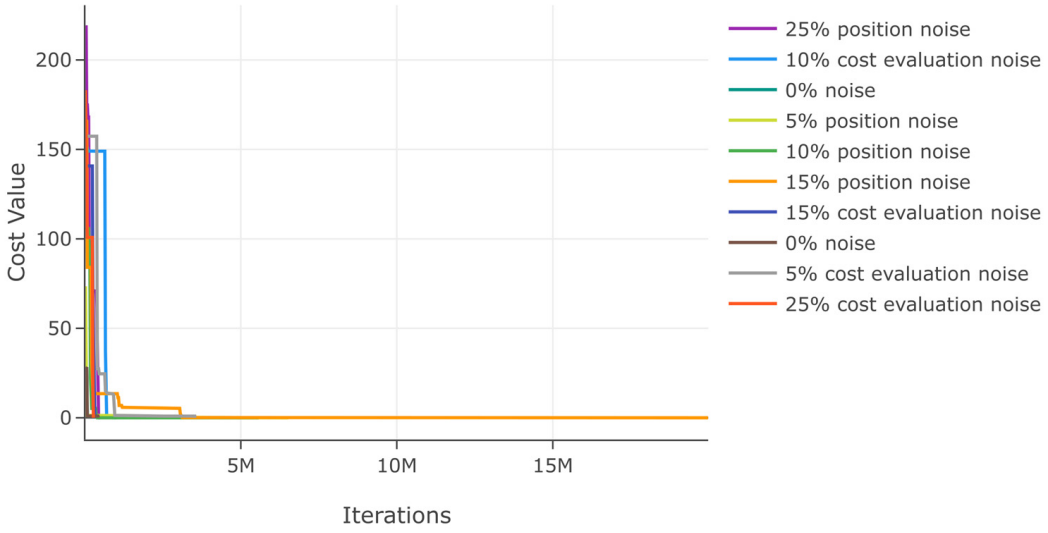


Figure 28. Effects of noise in position and cost evaluation for function 5.

3.1.7. Results Function 6

The best solution in an exhaustive search for function 6 was obtained with $n_g = 1$, $n_o = 4$ and $n_l = 2$ obtaining the best metrics $R^2 = 1.00 \times 10^{0}$, $f\left(\vec{M}_i\right) = 1.18 \times 10^{-10}$, MSE $= 5.04 \times 10^{-20}$, MAPE $= 1.54 \times 10^{-10}$, and MAE $= 1.46 \times 10^{-10}$, which show that DSRegPSOP successfully optimized the target surface based on R^2. The top 10 configurations are in Table 11, and Figure 29 shows the parallel plot with the results of all the configurations tested.

Table 11. Top 10 exhaustive search results testing function 6.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	1	4	2	1.46×10^{-10}	1.54×10^{-10}	5.04×10^{-20}	1.18×10^{-10}	1.00×10^{0}	17:29:46
2	1	5	3	7.85×10^{-3}	8.18×10^{-3}	1.26×10^{-4}	1.26×10^{-4}	9.23×10^{-1}	03:11:15
3	2	3	3	9.01×10^{-3}	9.50×10^{-3}	1.65×10^{-4}	1.65×10^{-4}	8.94×10^{-1}	08:24:51
4	3	2	4	1.16×10^{-2}	1.18×10^{-2}	2.20×10^{-4}	2.20×10^{-4}	8.17×10^{-1}	18:37:58
5	2	3	4	1.14×10^{-2}	1.18×10^{-2}	2.27×10^{-4}	2.27×10^{-4}	9.37×10^{-1}	11:47:21
6	4	4	1	1.08×10^{-2}	1.12×10^{-2}	2.45×10^{-4}	2.45×10^{-4}	5.53×10^{-1}	18:50:38
7	2	4	3	1.16×10^{-2}	1.23×10^{-2}	2.49×10^{-4}	2.49×10^{-4}	6.81×10^{-1}	12:43:36
8	3	3	3	1.20×10^{-2}	1.27×10^{-2}	2.72×10^{-4}	2.72×10^{-4}	9.15×10^{-1}	01:03:17
9	1	4	4	1.34×10^{-2}	1.38×10^{-2}	2.81×10^{-4}	2.81×10^{-4}	7.11×10^{-1}	06:29:32
10	1	4	3	1.06×10^{-2}	1.07×10^{-2}	2.89×10^{-4}	2.89×10^{-4}	9.68×10^{-1}	18:33:03

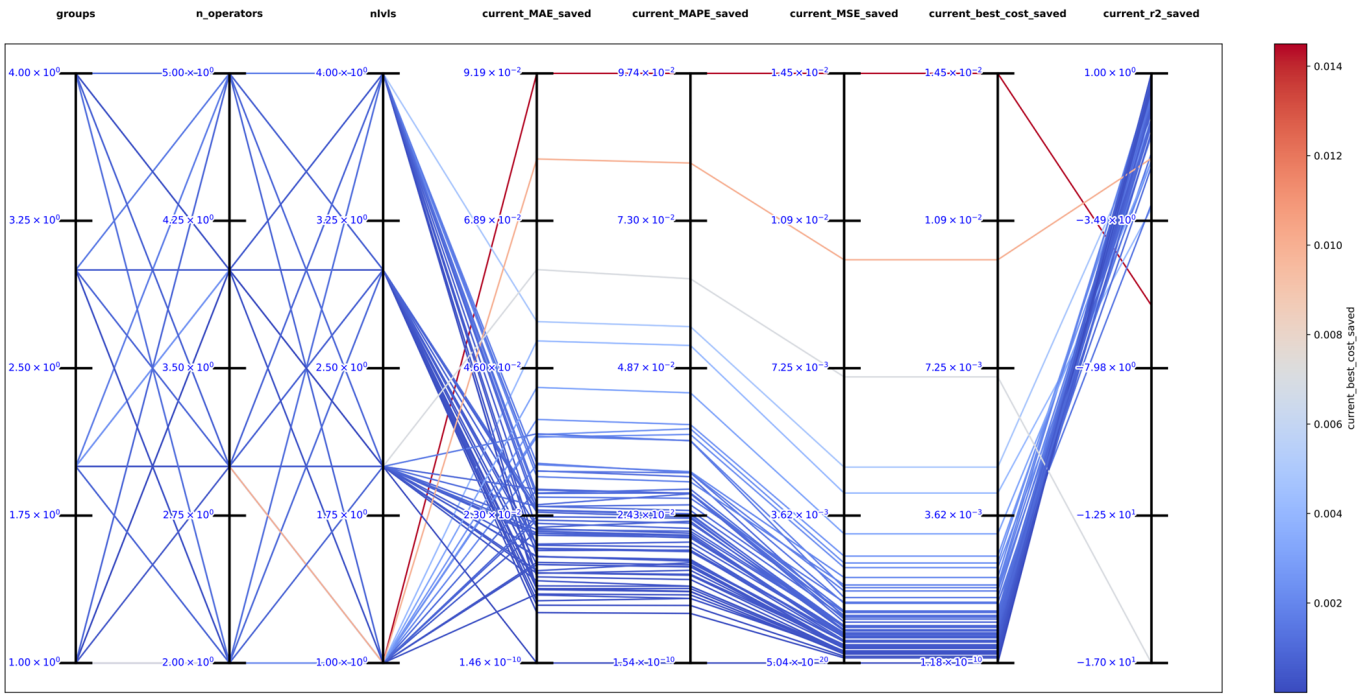


Figure 29. Parallel plot of hyperparameters for exhaustive search in function 6.

Figure 30 shows the comparison of the obtained and target surface, including the heatmap and principal component analysis, which are reduced to three dimensions for visualizing the positions of particles in the last iteration. Furthermore, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

Similarly, we evaluate DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% for function 6. The results in Table 12 show that the variation with noise is bigger than in functions 1 and 2 but lower than in 3. The convergence diagram for noise evaluations is in Figure 31, which shows that there is affectation with both noises, but the noise in position affects more than in cost value.

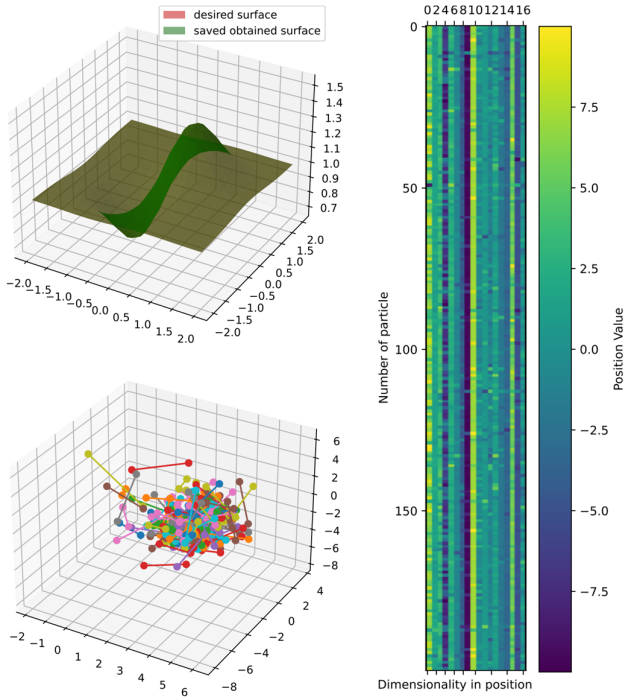


Figure 30. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 6.

Table 12. Effects of noise in position and cost evaluation for function 6.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	0.00×10^{0}	1.00×10^{-1}	1.59×10^{-3}	2.18×10^{-1}
2	1.50×10^{-1}	0.00×10^{0}	1.54×10^{-3}	7.99×10^{-1}
3	0.00×10^{0}	0.00×10^{0}	1.18×10^{-10}	1.00×10^{0}
4	1.00×10^{-1}	0.00×10^{0}	1.78×10^{-3}	-2.2×10^{-2}
5	2.50×10^{-1}	0.00×10^{0}	2.13×10^{-4}	8.18×10^{-1}
6	0.00×10^{0}	0.00×10^{0}	1.18×10^{-10}	1.00×10^{0}
7	0.00×10^{0}	1.50×10^{-1}	2.46×10^{-3}	-2.9×10^{-1}
8	5.00×10^{-2}	0.00×10^{0}	4.86×10^{-4}	8.57×10^{-1}
9	0.00×10^{0}	2.50×10^{-1}	2.62×10^{-3}	3.34×10^{-2}
10	0.00×10^{0}	5.00×10^{-2}	1.94×10^{-3}	-1.3×10^{0}

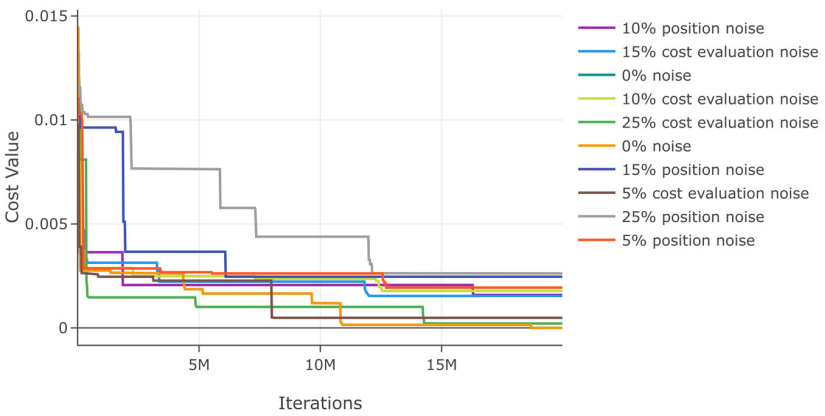


Figure 31. Effects of noise in position and cost evaluation for function 6.

3.1.8. Results Function 7

The best solution in an exhaustive search for function 7 was obtained with $n_g = 3$, $n_o = 3$ and $n_l = 2$ obtaining the best metrics $R^2 = 9.94 \times 10^{-1}$, $f\left(\vec{M}_i\right) = 4.95 \times 10^{-3}$, MSE $= 4.95 \times 10^{-3}$, MAPE $= 1.42 \times 10^{-2}$, and MAE $= 6.08 \times 10^{-2}$, which show that DSRegPSOP successfully optimized the target surface based on R^2. The top 10 configurations are in Table 13, and Figure 32 shows the parallel plot with the results of all the configurations tested.

Table 13. Top 10 exhaustive search results testing function 7.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	3	3	2	6.08×10^{-2}	1.42×10^{-2}	4.95×10^{-3}	4.95×10^{-3}	9.94×10^{-1}	11:26:15
2	4	4	4	1.06×10^{-1}	2.29×10^{-2}	1.98×10^{-2}	1.98×10^{-2}	9.78×10^{-1}	21:05:37
3	3	3	1	1.36×10^{-1}	3.20×10^{-2}	2.23×10^{-2}	2.23×10^{-2}	9.75×10^{-1}	4:08:54
4	4	5	1	1.46×10^{-1}	3.94×10^{-2}	3.05×10^{-2}	3.05×10^{-2}	9.65×10^{-1}	11:47:21
5	4	4	2	1.61×10^{-1}	4.76×10^{-2}	3.69×10^{-2}	3.69×10^{-2}	9.58×10^{-1}	2:15:00
6	2	3	1	1.69×10^{-1}	5.06×10^{-2}	4.19×10^{-2}	4.19×10^{-2}	9.52×10^{-1}	1:10:19
7	1	4	2	1.67×10^{-1}	5.03×10^{-2}	4.36×10^{-2}	4.36×10^{-2}	9.50×10^{-1}	19:37:44
8	4	2	1	1.61×10^{-1}	3.70×10^{-2}	4.70×10^{-2}	4.70×10^{-2}	9.47×10^{-1}	13:40:33
9	4	3	1	1.60×10^{-1}	3.48×10^{-2}	6.10×10^{-2}	6.10×10^{-2}	9.31×10^{-1}	14:51:34
10	1	5	3	2.02×10^{-1}	6.52×10^{-2}	6.55×10^{-2}	6.55×10^{-2}	9.26×10^{-1}	12:47:49

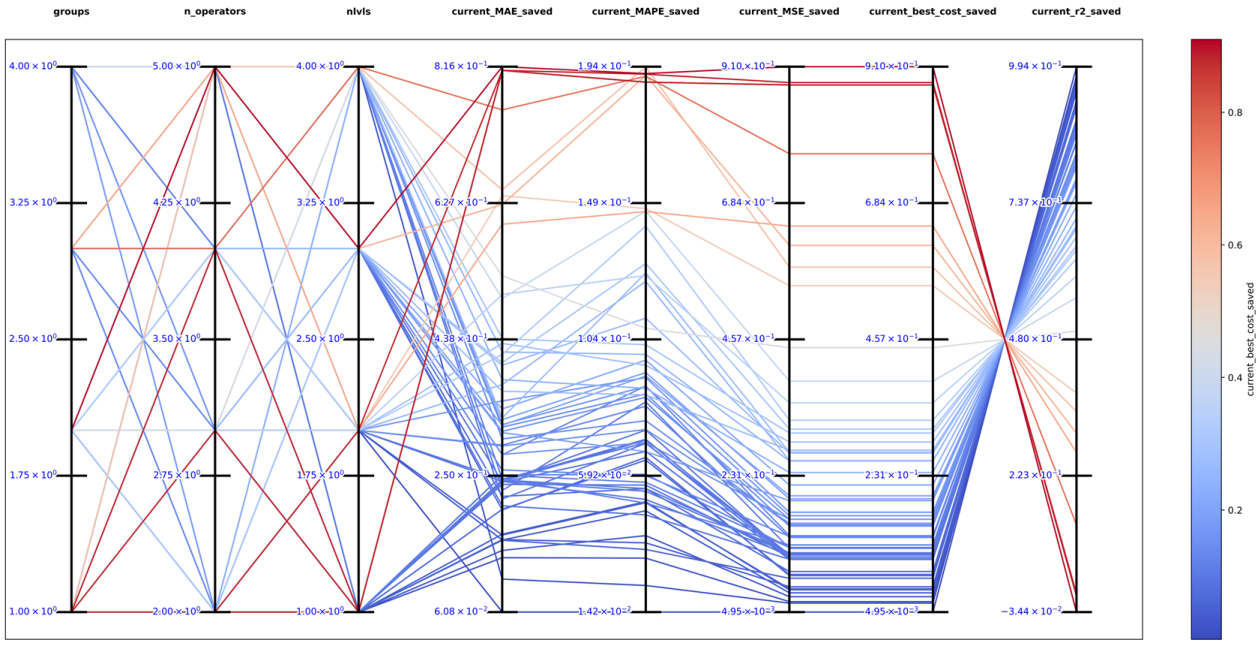


Figure 32. Parallel plot of hyperparameters for exhaustive search in function 7.

Figure 33 shows the comparison of the best obtained and target surface, including the heatmap and principal component analysis, which are reduced to three dimensions to show the positions of particles in the last iteration. Furthermore, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

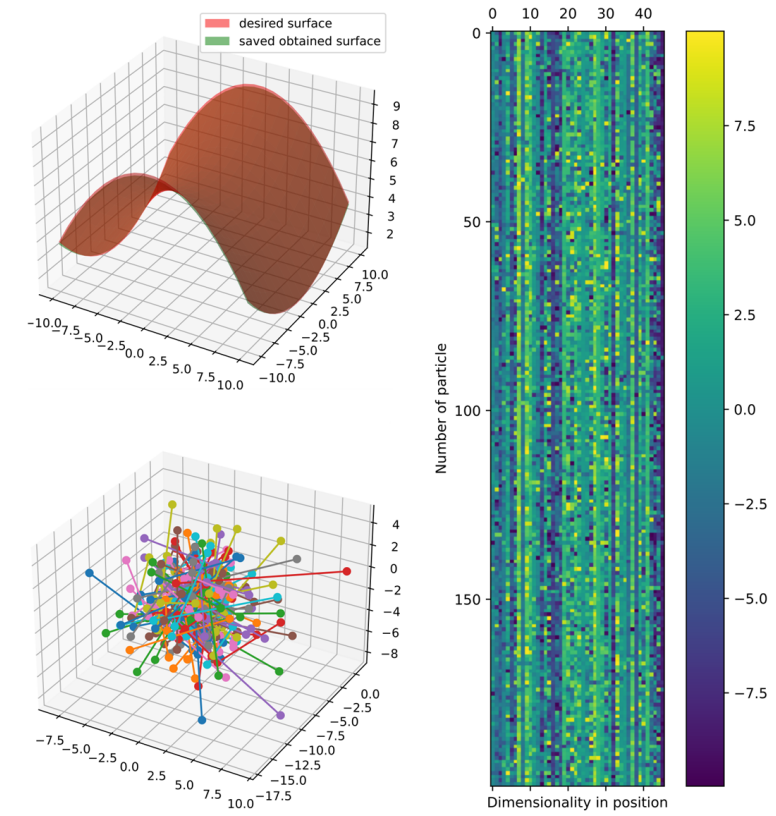


Figure 33. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 7.

Similarly, we evaluate DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% for function 7. The results in Table 14 show that the variation with noise is bigger than in functions 1 and 2 but lower than in 3. The convergence diagram for noise evaluations is in Figure 34; it shows that there is affectation with both noises, but the noise in position affects more than in cost value.

Table 14. Effects of noise in position and cost evaluation for function 7.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	0.00×10^{0}	1.50×10^{-1}	1.13×10^{0}	-2.8×10^{-1}
2	0.00×10^{0}	2.50×10^{-1}	1.23×10^{0}	-3.9×10^{-1}
3	1.50×10^{-1}	0.00×10^{0}	1.15×10^{-1}	8.46×10^{-1}
4	0.00×10^{0}	0.00×10^{0}	4.95×10^{-3}	9.94×10^{-1}
5	0.00×10^{0}	5.00×10^{-2}	8.54×10^{-1}	2.88×10^{-2}
6	1.00×10^{-1}	0.00×10^{0}	1.21×10^{-1}	8.48×10^{-1}
7	0.00×10^{0}	1.00×10^{-1}	1.08×10^{0}	-2.2×10^{-1}
8	2.50×10^{-1}	0.00×10^{0}	2.83×10^{-1}	5.72×10^{-1}
9	5.00×10^{-2}	0.00×10^{0}	9.58×10^{-2}	8.85×10^{-1}
10	0.00×10^{0}	0.00×10^{0}	4.95×10^{-3}	9.94×10^{-1}

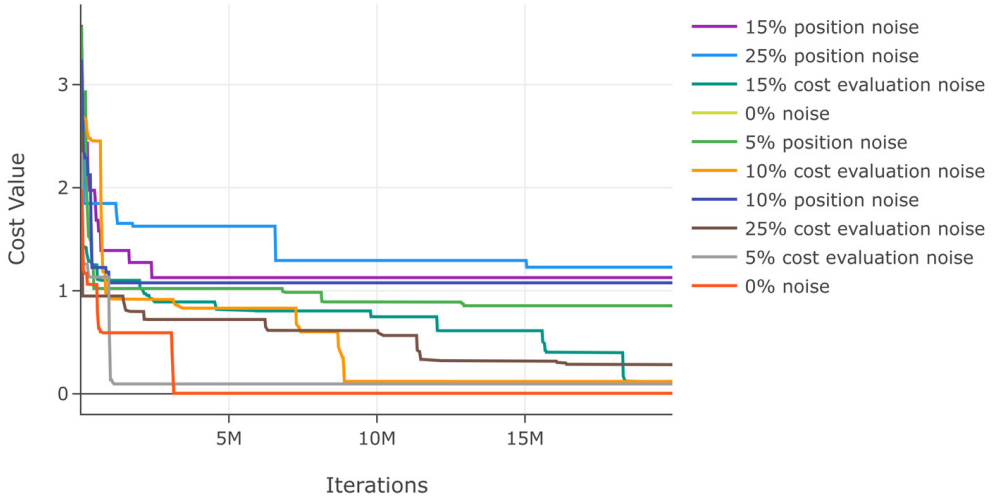


Figure 34. Effects of noise in position and cost evaluation for function 7.

3.1.9. Results Function 8

The best solution in an exhaustive search for function 8 was obtained with $n_g = 3$, $n_o = 4$ and $n_l = 2$ obtaining the best metrics $R^2 = 1.00 \times 10^0$, $f\left(\vec{M}_i\right) = 7.73 \times 10^{-1}$, MSE $= 7.73 \times 10^{-1}$, MAPE $= 9.12 \times 10^{-4}$, and MAE $= 7.79 \times 10^{-1}$, which show that DSRegPSOP successfully optimized the target surface based on R^2. The top 10 configurations are in Table 15, and Figure 35 shows the parallel plot with the results of all the configurations tested.

Table 15. Top 10 exhaustive search results testing function 8.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	3	4	2	7.79×10^{-1}	9.12×10^{-4}	7.73×10^{-1}	7.73×10^{-1}	1.00×10^{0}	8:17:49
2	2	4	2	3.42×10^{0}	1.17×10^{-2}	1.82×10^{1}	1.82×10^{1}	1.00×10^{0}	3:01:24
3	2	2	3	3.95×10^{0}	4.14×10^{-3}	2.07×10^{1}	2.07×10^{1}	1.00×10^{0}	8:27:39
4	3	3	2	5.25×10^{0}	5.10×10^{-3}	3.71×10^{1}	3.71×10^{1}	1.00×10^{0}	23:45:56
5	3	2	3	5.29×10^{0}	5.90×10^{-3}	3.72×10^{1}	3.72×10^{1}	1.00×10^{0}	6:08:26
6	2	3	3	6.00×10^{0}	1.03×10^{-2}	5.41×10^{1}	5.41×10^{1}	1.00×10^{0}	18:33:45
7	2	4	3	6.85×10^{0}	7.10×10^{-3}	7.19×10^{1}	7.19×10^{1}	1.00×10^{0}	19:10:19
8	1	4	2	7.05×10^{0}	1.25×10^{-2}	7.47×10^{1}	7.47×10^{1}	1.00×10^{0}	12:00:00
9	1	5	2	8.44×10^{0}	3.72×10^{-2}	8.52×10^{1}	8.52×10^{1}	1.00×10^{0}	23:30:28
10	1	4	3	8.31×10^{0}	2.43×10^{-2}	1.08×10^{2}	1.08×10^{2}	1.00×10^{0}	12:40:47

Figure 36 shows the comparison of the best obtained and target surface, including the heatmap and principal component analysis, which are reduced to three dimensions to show the positions of particles in the last iteration. Furthermore, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

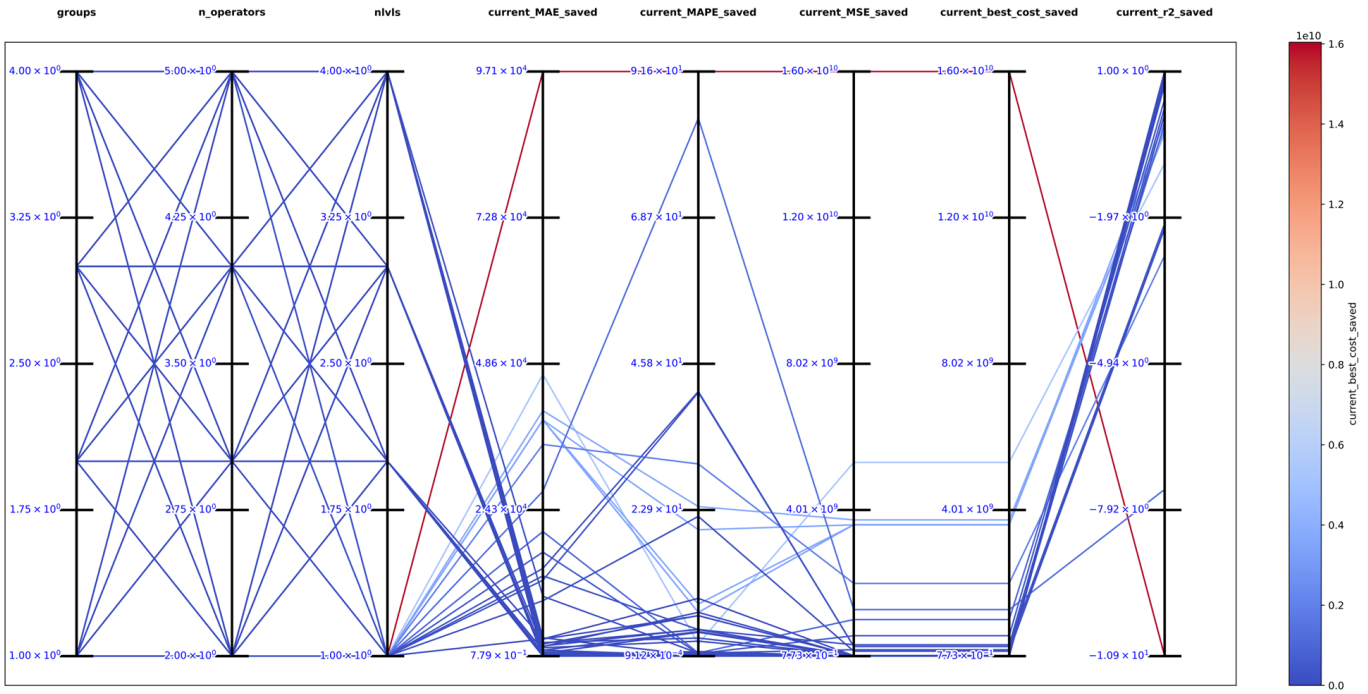


Figure 35. Parallel plot of hyperparameters for exhaustive search in function 8.

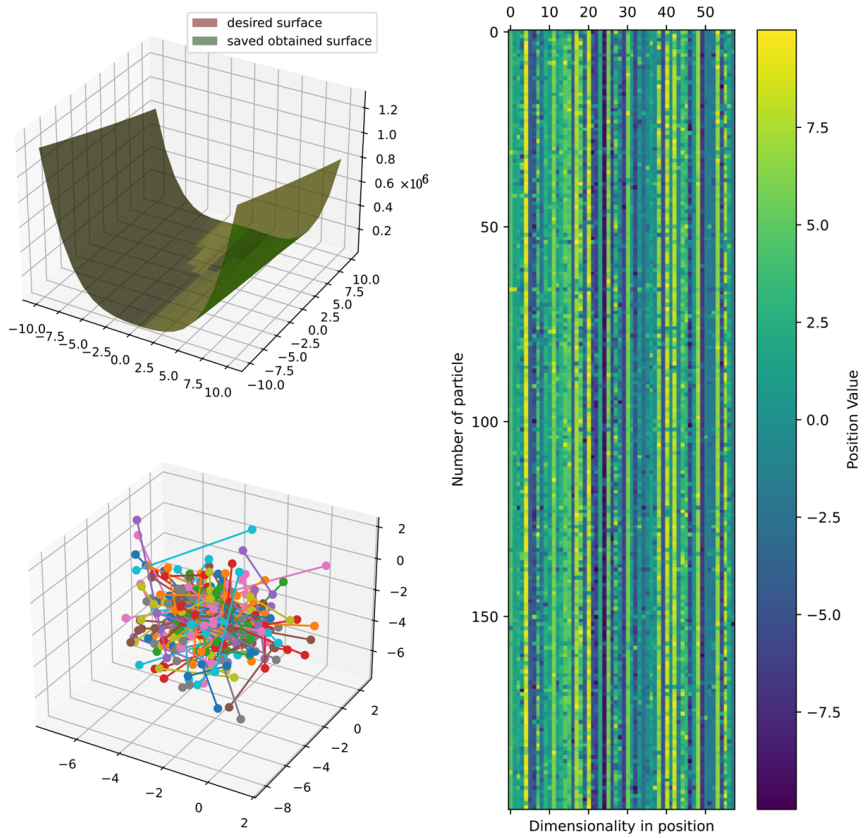


Figure 36. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 8.

Similarly, we evaluate DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% for function 8. The results in Table 16 show that the variation with noise

is minimal and reaches its maximum error at 25% of noise in position. The convergence diagram for noise evaluations is in Figure 37; it shows that there is minimal affectation with both noises, but the noise in position affects when it reaches 25% of noise.

Table 16. Effects of noise in position and cost evaluation for function 8.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	1.00×10^{-1}	0.00×10^{0}	3.16×10^{6}	9.65×10^{-1}
2	0.00×10^{0}	0.00×10^{0}	7.73×10^{-1}	1.00×10^{0}
3	0.00×10^{0}	0.00×10^{0}	7.73×10^{-1}	1.00×10^{0}
4	0.00×10^{0}	1.50×10^{-1}	9.09×10^{6}	9.94×10^{-1}
5	1.50×10^{-1}	0.00×10^{0}	1.20×10^{3}	1.00×10^{0}
6	0.00×10^{0}	2.50×10^{-1}	4.29×10^{9}	-3.0×10^{-01}
7	0.00×10^{0}	1.00×10^{-1}	1.77×10^{8}	5.80×10^{-1}
8	5.00×10^{-2}	0.00×10^{0}	1.39×10^{6}	8.86×10^{-1}
9	0.00×10^{0}	5.00×10^{-2}	1.60×10^{7}	9.75×10^{-1}
10	2.50×10^{-1}	0.00×10^{0}	1.96×10^{5}	9.99×10^{-1}

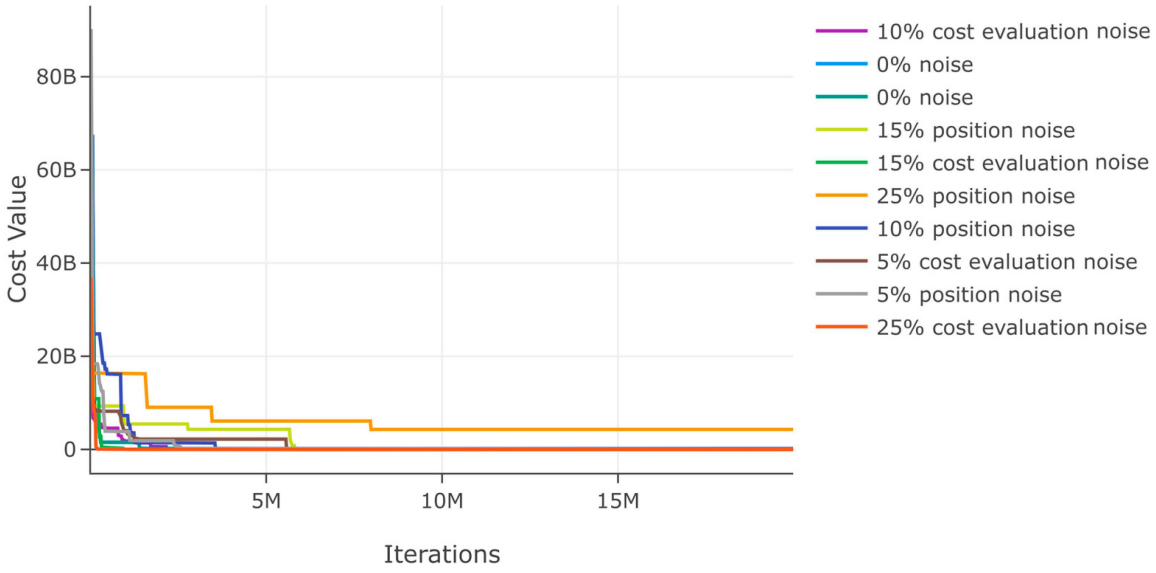


Figure 37. Effects of noise in position and cost evaluation for function 8.

3.1.10. Results Function 9

The best solution in an exhaustive search for function 9 was obtained with $n_g = 2$, $n_o = 4$ and $n_l = 4$ obtaining the best metrics $R^2 = 1.00 \times 10^{0}$, $f\left(\vec{M}_i\right) = 9.46 \times 10^{4}$, MSE $= 9.46 \times 10^{4}$, MAPE $= 3.07 \times 10^{-1}$, and MAE $= 2.20 \times 10^{2}$, which show that DSRegPSOP successfully optimized the target surface based on R^2. The top 10 configurations are in Table 17, and Figure 38 shows the parallel plot with the results of all the configurations tested.

Table 17. Top 10 exhaustive search results testing function 9.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	2	4	4	2.20×10^2	3.07×10^{-1}	9.46×10^4	9.46×10^4	1.00×10^0	5:51:34
2	4	2	2	5.45×10^2	8.57×10^0	4.86×10^5	4.86×10^5	1.00×10^0	15:11:15
3	4	4	2	5.36×10^2	8.39×10^0	5.32×10^5	5.32×10^5	1.00×10^0	21:37:58
4	2	2	4	5.57×10^2	7.79×10^0	6.14×10^5	6.14×10^5	1.00×10^0	0:26:43
5	3	3	4	5.71×10^2	6.60×10^0	7.09×10^5	7.19×10^5	1.00×10^0	23:36:06
6	3	2	3	5.06×10^2	2.24×10^{-1}	7.22×10^5	7.22×10^5	1.00×10^0	20:10:47
7	2	3	3	5.12×10^2	6.29×10^{-1}	7.97×10^5	8.07×10^5	1.00×10^0	8:24:51
8	4	2	3	6.48×10^2	7.65×10^0	8.69×10^5	8.69×10^5	1.00×10^0	4:24:22
9	4	3	3	8.12×10^2	7.05×10^0	1.22×10^6	1.23×10^6	1.00×10^0	21:46:24
10	3	3	2	8.70×10^2	3.72×10^0	1.39×10^6	1.39×10^6	1.00×10^0	23:41:01

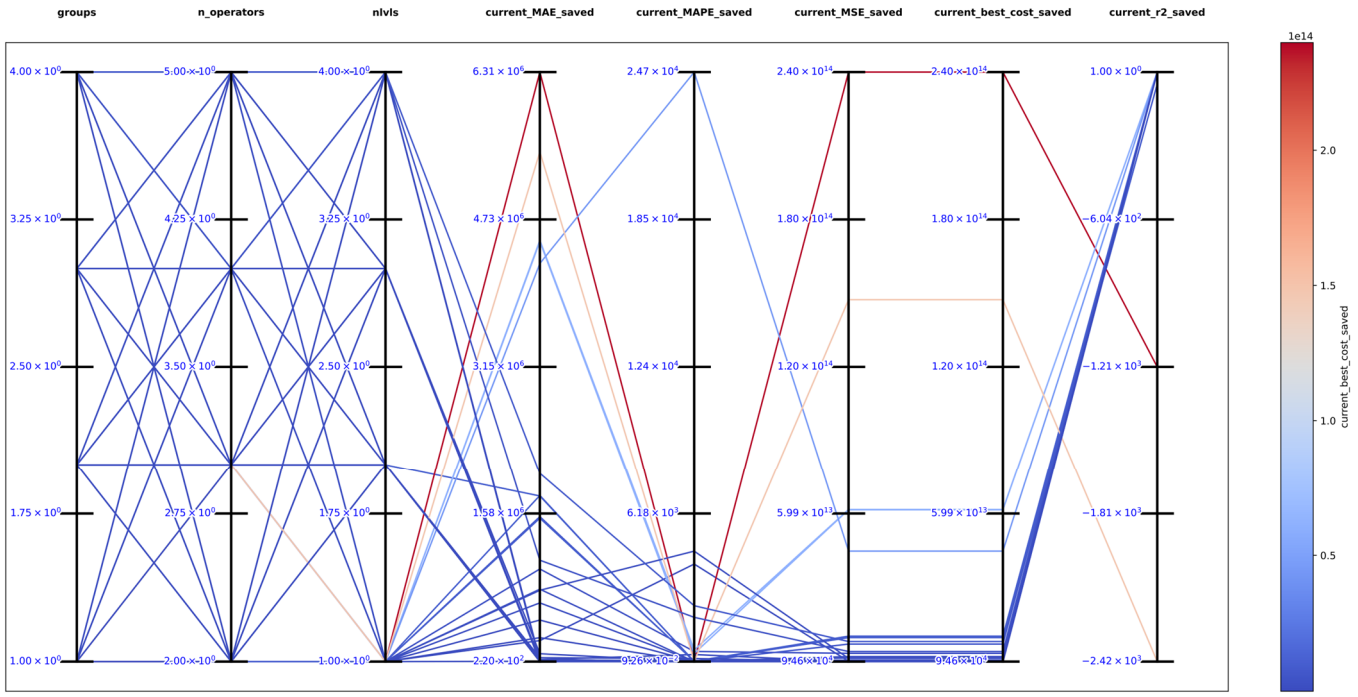


Figure 38. Parallel plot of hyperparameters for exhaustive search in function 9.

Figure 39 shows the comparison of the obtained and target surface, including the heatmap and principal component analysis, which are reduced to three dimensions to show the positions of particles in the last iteration. Furthermore, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

Again, we evaluate DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% in function 9. The results in Table 18 show a minimal variation in best cost and R^2. Moreover, the minimal variation with noise is also evident in the convergence diagram in Figure 40.

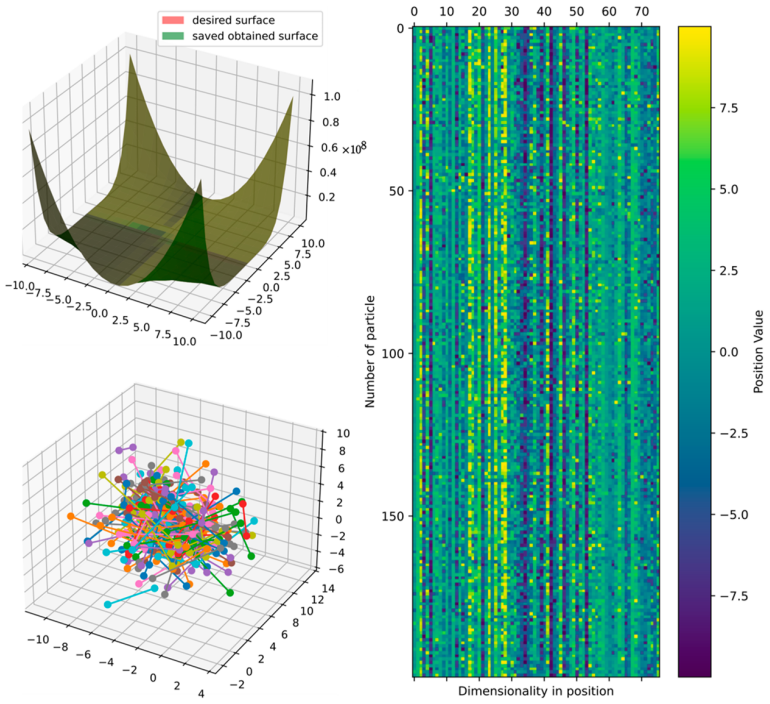


Figure 39. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 9.

Table 18. Effects of noise in position and cost evaluation for function 9.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	5.00×10^{-2}	0.00×10^{0}	7.96×10^{5}	1.00×10^{0}
2	0.00×10^{0}	0.00×10^{0}	9.46×10^{4}	1.00×10^{0}
3	0.00×10^{0}	2.50×10^{-1}	3.88×10^{10}	1.00×10^{0}
4	1.00×10^{-1}	0.00×10^{0}	1.49×10^{6}	1.00×10^{0}
5	0.00×10^{0}	1.50×10^{-1}	4.92×10^{9}	1.00×10^{0}
6	0.00×10^{0}	5.00×10^{-2}	5.26×10^{8}	1.00×10^{0}
7	0.00×10^{0}	1.00×10^{-1}	3.28×10^{9}	1.00×10^{0}
8	1.50×10^{-1}	0.00×10^{0}	1.63×10^{6}	1.00×10^{0}
9	0.00×10^{0}	0.00×10^{0}	9.46×10^{4}	1.00×10^{0}
10	2.50×10^{-1}	0.00×10^{0}	1.86×10^{6}	1.00×10^{0}

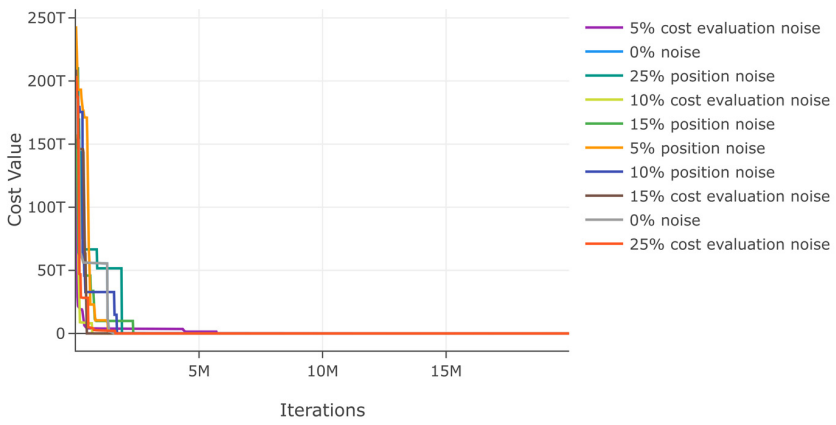


Figure 40. Effects of noise in position and cost evaluation for function 9.

3.1.11. Results Function 10

The best solution in an exhaustive search for function 10 was obtained with $n_g = 3$, $n_o = 4$ and $n_l = 2$ obtaining the best metrics $R^2 = 1.00 \times 10^0$, $f\left(\vec{M}_i\right) = 6.25 \times 10^2$, MSE $= 6.25 \times 10^2$, MAPE $= 7.13 \times 10^{-1}$, and MAE $= 2.10 \times 10^1$, which show that DSRegPSOP successfully optimized the target surface based on R^2. The top 10 configurations are in Table 19, and Figure 41 shows the parallel plot with the results of all the configurations tested.

Table 19. Top 10 exhaustive search results testing function 10.

Rank	n_g	n_o	n_l	Best MAE	Best MAPE	Best MSE	Best $f\left(\vec{M}_i\right)$	Best R^2	Time
1	3	4	2	2.10×10^1	7.13×10^{-1}	6.25×10^2	6.25×10^2	1.00×10^0	21:09:08
2	3	3	4	3.06×10^1	3.23×10^{-1}	1.21×10^3	1.21×10^3	1.00×10^0	13:31:24
3	2	2	2	1.01×10^2	1.68×10^{-1}	1.78×10^4	1.78×10^4	1.00×10^0	08:05:52
4	1	2	4	1.17×10^2	1.31×10^0	2.13×10^4	2.13×10^4	1.00×10^0	10:01:10
5	3	4	4	8.78×10^1	2.80×10^0	1.41×10^4	2.41×10^4	1.00×10^0	03:53:26
6	2	3	3	1.26×10^2	3.34×10^{-1}	3.01×10^4	3.01×10^4	1.00×10^0	15:59:04
7	1	3	4	1.79×10^2	2.86×10^{-1}	5.92×10^4	5.92×10^4	1.00×10^0	16:18:03
8	2	2	4	1.77×10^2	1.49×10^0	6.25×10^4	6.25×10^4	1.00×10^0	15:33:45
9	4	4	3	2.15×10^2	3.81×10^{-1}	8.31×10^4	8.31×10^4	1.00×10^0	17:05:09
10	4	3	4	2.87×10^2	9.89×10^{-1}	1.70×10^5	1.80×10^5	1.00×10^0	16:38:26

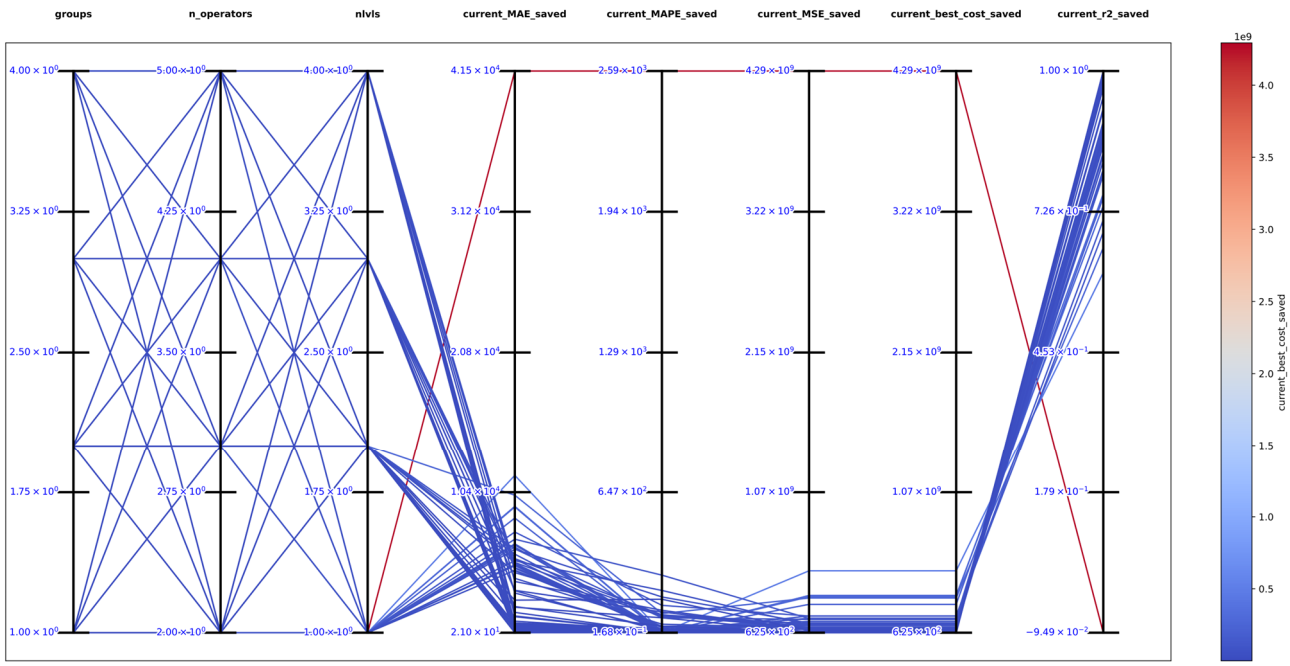


Figure 41. Parallel plot of hyperparameters for exhaustive search in function 10.

Figure 42 shows the comparison of the best obtained and target surface, including the heatmap and principal component analysis, which are reduced to three dimensions to show the positions of particles in the last iteration. Furthermore, the particles could still improve since the DSRegPSOP algorithm never converges and is still exploring the search space as DSRegPSO does.

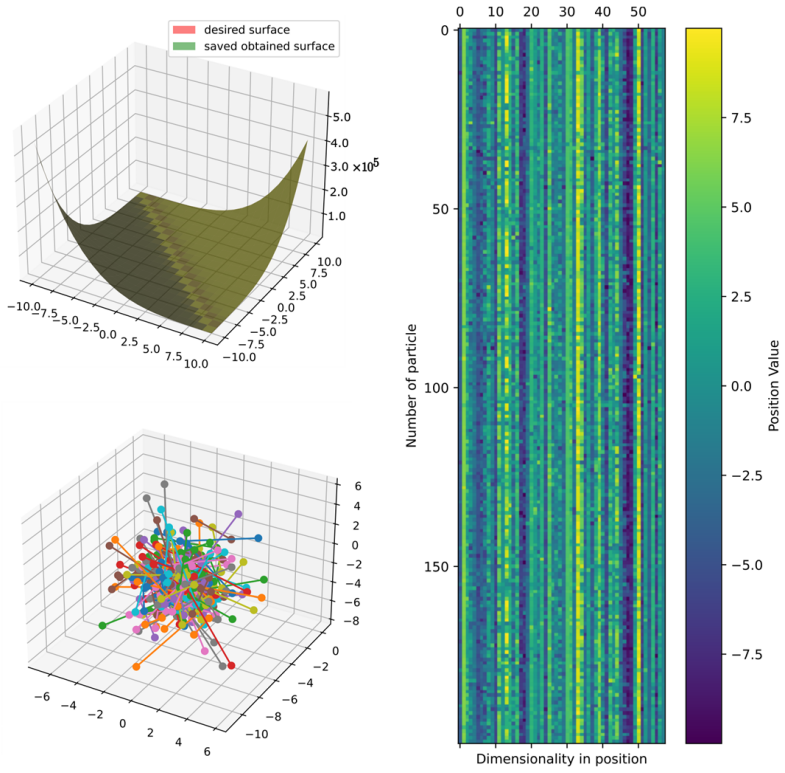


Figure 42. Comparison of obtained and target surface showing PCA with random color for each of the 200 particles and heatmap for the position of particles from 1.98×10^7 to 2.00×10^7 function evaluations for function 10.

Similarly, we test DSRegPSOP for noise effect in position and cost evaluation with ranges 0–25% for function 10. The results in Table 20 show that the variation with noise is minimal and reaches its maximum error at 25% of noise in position. The convergence diagram for noise evaluations is in Figure 43; it shows that there is minimal affectation with both noises, but the noise in position affects more than in cost value.

Table 20. Effects of noise in position and cost evaluation for function 10.

Rank	Noise in $f(M)$	Noise in X	Best $f(M)$	Best R^2
1	0.00×10^{0}	1.00×10^{-1}	2.91×10^{6}	9.97×10^{-1}
2	0.00×10^{0}	2.50×10^{-1}	1.19×10^{8}	8.79×10^{-1}
3	0.00×10^{0}	1.50×10^{-1}	4.16×10^{7}	9.70×10^{-1}
4	2.50×10^{-1}	0.00×10^{0}	1.06×10^{6}	9.97×10^{-1}
5	0.00×10^{0}	0.00×10^{0}	6.25×10^{2}	1.00×10^{0}
6	1.50×10^{-1}	0.00×10^{0}	2.84×10^{3}	1.00×10^{0}
7	1.00×10^{-1}	0.00×10^{0}	1.51×10^{6}	9.96×10^{-1}
8	5.00×10^{-2}	0.00×10^{0}	4.06×10^{5}	9.99×10^{-1}
9	0.00×10^{0}	0.00×10^{0}	6.25×10^{2}	1.00×10^{0}
10	0.00×10^{0}	5.00×10^{-2}	3.65×10^{7}	9.91×10^{-1}

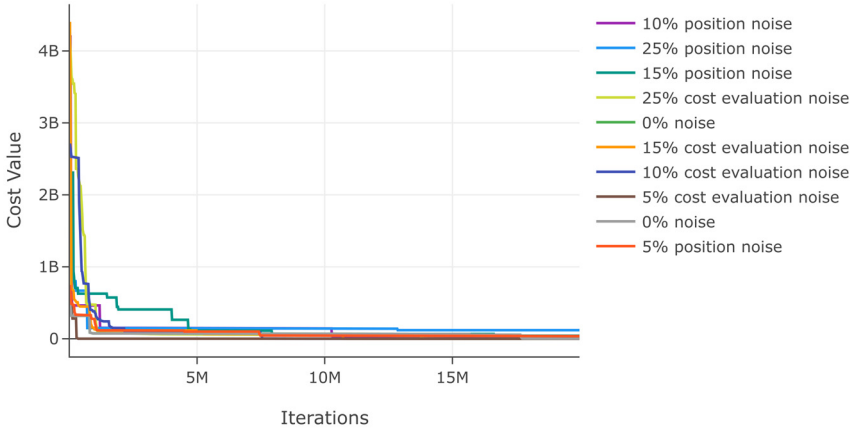


Figure 43. Effects of noise in position and cost evaluation for function 10.

3.2. Results in Benchmark with Datasets

3.2.1. Experimental Setup of Benchmark with Datasets

All the datasets in this benchmark have the same considerations:

- We split datasets into 80% for training and 20% for testing and performing six-fold cross-validation.
- The maximum number of function evaluations for each dataset is 2×10^3 to compare with several algorithms in the first stage of optimization described in [63].
- During hyperparameter tunning, the test stops, and the result is recorded if the cost value reaches 1×10^{-10}.
- The list of variables is:
  $L_v = [b, x, y, e, \pi]$.
- The list of operators is:
  $L_o = \left[\div, sin(\ ), cos(\ ), (,), \sqrt{}, \wedge, \Sigma, -, +, \times\right]$.
- The DSRegPSOP static parameters for all the functions are:
  $n = 5, TS, f\left(\vec{M}_i\right), L_l = -1.00, L_u = 1.00, c_1 = c_2 = 2.00, M_{max} = 1.10, \lambda = 1.00,$

$$S_{max} = 4.00 \times 10^{-2}, S_{min} = 1.00 \times 10^{-2}, \zeta = 1.00 \times 10^{-3}$$

We chose $n = 5$ to demonstrate the capabilities of DSRegPSOP by optimizing with only five particles. This number was sufficient for the datasets we used. However, for the more complex initial benchmark involving surfaces, we needed to increase $n = 200$ for effective optimization.

- The parameters that are dependent on the optimized dataset are:
  fd_{max} and fd_{min}
- The fd_{max} and fd_{min} parameters are adapted for each function so that:
  $1 \times 10^{-2} < \delta_{max} < 1 \times 10^{-1}$ and $1 \times 10^{-12} < \delta_{min} < 1 \times 10^{-10}$ in the first iteration.
- The n_l, n_g, and n_o parameters are obtained with an exhaustive search exploring the 64 combinations produced with: $n_l = [1, 2, 3, 4]$, $n_g = [1, 2, 3, 4]$, and $n_o = [2, 3, 4, 5]$.
- Again, we perform a hyperparameter tuning with 64 configurations and six-fold cross-validation, as before, but we also perform running/function evaluations to compare the DSRegPSOP results with those in the literature for each dataset.
- All the generated models for dataset prediction apply the RELU activation function at the final register or the maximum across the evaluation and zero value. This has shown improved results in regression tasks with ANNs, and we found DSRegPSOP benefits of it.

3.2.2. Results in Datasets Benchmark

Table 21 shows the mean RMSE and standard deviation obtained with the 2×10^3 function evaluations for the training stage in the six-fold cross-validation for regression in the real-world datasets. These results allow the comparison with other algorithms described in [63]. DSRegPSOP is not the best solution, which is understandable with 2×10^3 function evaluations when the equation is generated with metaheuristic algorithms with AP approaches. However, it is better than at least one of the variants of GP (dcGP, GP, or GSGP) in five of the six datasets. Moreover, with more function evaluations, DSRegPSOP obtains competitive results that are the best reported in the literature for each dataset.

Table 21. Comparison of algorithms with DSRegPSOP training RMSE and standard deviation with benchmark datasets.

Algorithm	Airfoil	Concrete	Cooling	Yacht	Geo	Tecator
ITEA	2.27 ± 0.06	5.81 ± 0.11	1.44 ± 0.05	0.49 ± 0.06	32.94 ± 2.21	0.7 ± 0.09
SymTree	1.50 ± 0.08	3.67 ± 0.14	1.12 ± 0.08	1.21 ± 0.35	31.95 ± 0.90	0.42 ± 0.04
FEAT-full	3.06 ± 0.33	6.00 ± 0.29	1.60 ± 0.04	0.54 ± 0.13	41.96 ± 4.35	1.02 ± 0.54
FEAT	3.48 ± 0.46	6.63 ± 0.44	1.72 ± 0.13	1.73 ± 0.63	41.87 ± 1.44	0.83 ± 0.30
dCGP	7.56 ± 1.33	19.45 ± 2.42	12.89 ± 2.35	17.09 ± 1.76	55.60 ± 3.41	4.10 ± 0.35
GP	10.28 ± 7.26	13.64 ± 1.89	4.44 ± 0.91	6.38 ± 2.45	49.66 ± 2.58	3.00 ± 0.63
GSGP	10.60 ± 4.62	7.24 ± 0.74	2.07 ± 0.43	8.81 ± 1.11	42.03 ± 0.96	1.04 ± 0.46
Forest	1.80 ± 0.03	2.97 ± 0.07	1.22 ± 0.03	0.04 ± 0.02	11.26 ± 0.76	0.63 ± 0.02
Tree	2.04 ± 0.08	2.90 ± 0.16	1.68 ± 0.07	0.33 ± 0.41	19.73 ± 3.50	0.66 ± 0.21
kNN	5.90 ± 0.08	8.27 ± 0.09	1.97 ± 0.04	8.75 ± 0.68	31.51 ± 0.52	1.39 ± 0.04
LR	4.82 ± 0.04	10.47 ± 0.12	3.22 ± 0.02	8.94 ± 0.15	43.70 ± 0.18	0.80 ± 0.03
DSRegPSOP	6.41 ± 1.01	13.10 ± 1.88	5.24 ± 0.59	7.55± 2.63	26.30 ± 8.48	8.19 ± 2.07

Table 22 shows the mean RMSE and standard deviation obtained with the 2×10^3 function evaluations for the testing stage in the six-fold cross-validation for regression in the real-world datasets. Again, DSRegPSOP is not the best solution. However, it is better than GP in eight of the nine datasets. Moreover, with more function evaluations, DSRegPSOP obtains competitive results that are the best reported in the literature for each dataset.

Table 22. Comparison of algorithms with DSRegPSOP testing RMSE and standard deviation with benchmark datasets.

Algorithm	Airfoil	Concrete	Cooling	Yacht	Geo	Tecator
ITEA	2.45 ± 0.21	6.33 ± 0.47	1.53 ± 0.13	0.75 ± 0.58	35.65 ± 2.69	1.2 ± 0.37
SymTree	2.12 ± 0.20	5.56 ± 0.38	1.61 ± 0.17	1.61 ± 1.22	43.53 ± 1.49	0.71 ± 0.16
FEAT-full	3.18 ± 0.31	6.35 ± 0.28	1.63 ± 0.12	0.83 ± 0.22	45.11 ± 9.02	2.02 ± 4.93
FEAT	3.55 ± 0.44	7.06 ± 1.09	1.75 ± 0.16	1.83 ± 0.65	45.67 ± 9.12	1.03 ± 0.32
dCGP	7.55 ± 1.38	19.50 ± 2.27	12.84 ± 2.25	17.01 ± 2.20	63.75 ± 46.90	4.10 ± 0.43
GP	10.31 ± 7.21	13.65 ± 1.90	4.46 ± 0.91	6.68 ± 2.94	49.72 ± 2.94	3.05 ± 0.70
GSGP	10.60 ± 4.35	7.86 ± 1.10	2.13 ± 0.42	9.03 ± 1.03	44.35 ± 1.51	1.41 ± 0.56
Forest	2.37 ± 0.07	5.25 ± 0.32	1.79 ± 0.12	1.07 ± 0.31	25.78 ± 3.65	1.25 ± 0.12
Tree	2.93 ± 0.10	5.01 ± 0.35	1.85 ± 0.14	1.44 ± 0.26	31.92 ± 7.42	1.68 ± 0.15
kNN	5.86 ± 0.26	7.85 ± 0.73	1.92 ± 0.13	7.56 ± 1.31	31.17 ± 0.87	1.29 ± 0.09
LR	4.82 ± 0.17	10.44 ± 0.49	3.21 ± 0.11	8.93 ± 0.71	42.95 ± 1.26	0.74 ± 0.06
DSRegPSOP	6.46 ± 0.90	12.55 ± 1.61	5.26 ± 0.54	6.12± 2.02	26.75 ± 9.89	8.14 ± 2.54

After 2×10^6 function evaluations in a time span of 1 h, 58 min, and 12 s, the DSRegPSOP obtains an RMSE and R^2 around the average of the best results obtained with algorithms that have been used for regression for predicting pressure with the Airfoil dataset (Table 23) [64,65]. However, it is the only one that produces a model entirely generated with AP approaches instead of tuning parameters in a human-proposed structure. The parameters for obtaining the best model in Equations (38) to (41) are $n_g = 3$, $n_o = 2$ and $n_l = 4$.

$$m_0 = \left(X_3 + \frac{1}{\pi}\right)\sin(0.121X_3 + 0.266) + \cos(X_3 - \sin(X_3)) \tag{38}$$

$$m_1 = \sin(m_0\pi)\left(\sin\left(X_2 \sum_{i_s=0}^{n_s} X_{2i_s}X_{4i_s}\right)\right)X_2 + \cos(X_0\cos^{m_0}(1)) \tag{39}$$

$$m_2 = \left(X_1 + X_2^{0.5} - X_4\right)\sin\left(X_0\left(e^{X_4}\right)^{X_4}\right) + \cos(m_1X_4 + \pi) \tag{40}$$

$$m_3 = X_0\sin(m_2\pi + X_2)\sin^{X_0}(X_4) + \cos(e + m_2 - \sin(m_0)) \tag{41}$$

Table 23. Best results in the literature for regression with the Airfoil dataset.

Algorithm/Dataset	Airfoil R^2	Airfoil RMSE
Linear Regression	0.46	5.15
Ridge Regression	0.46	5.15
Lasso Regression	0.46	5.16
SVM	0.80	3.14
ANN	0.89	2.33
DNN	0.95	1.54
DSRegPSOP	0.66	4.03

Similarly, after 2×10^6 function evaluations in a time span of 3 h, 10 min, and 47 s, the DSRegPSOP obtains the best RMSE and R^2 results compared with other algorithms that have been used for regression with the concrete strength dataset as detailed in Table 24 [66]. Moreover, DSRegPSOP is the only one that produces a model entirely generated with AP approaches instead of tuning parameters in a proposed structure. The parameters for obtaining the best model in Equations (42) to (44) are $n_g = 3$, $n_o = 3$, and $n_l = 3$.

$$m_0 = 0.841X_3X_6^{0.5}\sin\left(\frac{X_0^2X_5}{X_7}\right) + 0.463X_3 + 0.54\pi \tag{42}$$

$$m_1 = X_1 + X_4^{X_3}\cos(X_4) + (-X_3X_5 + X_7 + \sin(X_4))\sin(m_0\pi\cos(X_3) + 0.408) \tag{43}$$

$$m_2 = 0.523X_0 + 0.477X_7 + m_1e^{X_1}\sin^{X_3}(X_7)\sin\left(X_3e^{X_2} + X_5\sin(X_6)\right) \tag{44}$$

Table 24. Best results in the literature for regression with the Concrete dataset.

Algorithm/Dataset	Concrete R^2	Concrete RMSE
Linear Regression	0.57	10.28
Ridge Regression	0.57	10.29
Lasso Regression	0.54	10.68
SVM	0.66	9.13
ANN	0.75	6.02
DSRegPSOP	0.78	8.03

DSRegPSOP, after the 2×10^6 function evaluations in a time span of 1 h, 33 min, and 31 s, obtains an RMSE and R^2 around the average of the best results obtained with algorithms that have been used for regression with the cooling load dataset in Table 25 [67]. However, it is the only one that produces a model entirely generated with AP approaches instead of tuning parameters in a human-proposed structure. The parameters for obtaining the best model in Equations (45) to (47) are $n_g = 4$, $n_o = 2$ and $n_l = 3$.

$$m_0 = \pi - \sin\left(X_1^{X_6}\sqrt{\pi}\right)\cos\left(X_1\sum_{i_s=0}^{n_s} 0.5 + X_2\right) - \sin(X_4 + \sin(X_1) - 1) - \cos(X_6) \quad (45)$$

$$m_1 = 0.841X_3 + m_0 + \sin\left(X_0 e^{X_4+X_6}\right) - \sin\left(X_1 - X_4 e^{X_1}\right)\cos\left(m_0 + X_5 - \sum_{i_s=0}^{n_s} X_{2i_s}\right) \quad (46)$$

$$m_2 = -0.0625X_3\cos(m_1) - \sin\left(\frac{X_0}{m_0}\right)\cos\left(e^{X_3} - \sum_{i_s=0}^{n_s} X_{5i_s}\right) + \sin(X_4 + 0.179X_6) \quad (47)$$

Table 25. Best results in the literature for regression with Cooling Load dataset.

Algorithm/Dataset	Cooling Load R^2	Cooling Load RMSE
Linear Regression	0.95	2.96
Gaussian Regression	0.95	2.92
Multilayer Perceptron	0.97	2.10
SVM	0.94	3.10
ANN	0.95	2.78
ANFIS	0.99	1.60
DSRegPSOP	0.94	2.38

DSRegPSOP, after 2×10^6 function evaluations in a time span of 1 h, 32 min, and 18 s, obtains an RMSE and R^2 around the average of the best results obtained with algorithms that have been used for regression in the prediction of sailing resistance in yachts as detailed in Table 26 [68,69]. However, it is the only one that produces a model entirely generated with AP approaches instead of tuning parameters in a human-proposed structure. The parameters for obtaining the best model in Equations (48) to (51) are $n_g = 2$, $n_o = 3$, and $n_l = 4$.

$$m_0 = \frac{X_1 + e + \sin\left(\frac{e}{\pi}\right)}{X_3 + \frac{\sum_{i_s=0}^{n_s} X_{0i_s}}{\pi X_0}} \quad (48)$$

$$m_1 = \frac{X_1\cos(X_0) + m_0 + \pi}{X_5\cos(X_3X_5m_0)} \quad (49)$$

$$m_2 = \frac{m_1 + e^{m_0}}{X_1 + m_0^{0.5}X_0 + \pi} \quad (50)$$

$$m_3 = \frac{X_5^2\left(e^{X_5}\right)^{X_5}}{m_1 + 0.488\cos(m_2\pi)} \quad (51)$$

DSRegPSOP obtains after 2×10^6 function evaluations in a time span of 1 h, 51 min, and 32 s, an RMSE and R^2 around the average of the best results obtained with algorithms that have been used for regression in the prediction of geographical latitude based on music as detailed in Table 27 [57,59]. Moreover, DSRegPSOP is the only one that produces a model entirely generated with AP approaches instead of tuning parameters in a human-proposed structure. The parameters for obtaining the best model for latitude prediction in Equations (52) to (54) are $n_g = 2$, $n_o = 5$, and $n_l = 3$.

$$m_0 = X_{32}^{X_{73}} + X_{50} - X_{55} + \left(e^{X_{61}}\right)^{X_{93}} - \sin\left(-X_{30}X_{90} + X_{91} + \cos(X_{57}) + \left(\sum_{i_s=0}^{n_s} X_{95i_s}\right)^{X_{59}}\right) \quad (52)$$

$$m_1 = X_{112} - X_{29}^{X_2^{X_{64}}} - X_{33}^{0.5}m_0 + \sin\left(X_{16}\sin(X_{73}) - X_{82} + 1.55X_{99} + \sum_{i_s=0}^{n_s} X_{113i_s}\right) \quad (53)$$

$$m_2 = X_2X_{48} + X_{79}e^{X_1} - X_{83}e^{X_{31}} + \sin\left(X_{26}m_1\cos(X_{19} + X_{35}) + \sin^{X_{31}}(X_{31})\right) \quad (54)$$

Table 26. Best results in the literature for regression with yachts sailing resistance dataset.

Algorithm/Dataset	Yachts Sailing Resistance R^2	Yachts Sailing Resistance RMSE
Position-gradient type fuzzy model	1.00	0.17
Neuro-fuzzy model based on extended RVM	1.00	0.21
NN model based on ELM	1.00	0.16
NN model based on IELM	1.00	0.14
NN model based on CIELM	1.00	0.13
ANN with L2 regularization	0.99	1.51
DSRegPSOP	1.00	1.08

Table 27. Best results in the literature for regression with geographical location dataset.

Algorithm/Dataset	Geographical Location of Music R^2	Geographical Location of Music RMSE
XGBoost	0.32	15.19
Random Forest	0.28	15.67
Ridge Regression	0.14	17.11
Regression Tree	0.04	18.09
Generalized Additive Model	0.12	17.33
DSRegPSOP	0.21	15.94

DSRegPSOP obtains after 2×10^6 function evaluations in a time span of 2 h, 56 min, and 27 s, the best RMSE and R^2 obtained compared with algorithms that have been used for regression in the prediction of fat content of in a meat sample as detailed in Table 28 [70]. Moreover, DSRegPSOP is the only one that produces a model entirely generated with AP approaches instead of tuning parameters in a human-proposed structure. The parameters for obtaining the best model in Equations (55) to (57) are $n_g = 2$, $n_o = 4$, and $n_l = 3$.

$$m_0 = \left(X_5 + X_{58}^{0.5X_{29}}\left(X_{120}^{X_{16}}\right)^{0.5}\right)\cos\left(X_{99} + \cos(X_{110}) - \frac{e^{X_{77}+X_{93}}}{e}\right) \quad (55)$$

$$m_1 = (-X_{105}X_5X_9 + X_{121} + \cos(X_{54}))\cos(-X_{12} + X_{61}X_{89}\sin(X_2) + \cos(m_0)) \quad (56)$$

$$m_2 = \left(-X_{122} + \frac{X_{13}}{X_{19}} - X_{25} + X_{40}\right)\cos\left(X_{40}^{X_{108}}m_1^{X_{63}} - \cos(X_{101})\right) \quad (57)$$

Table 28. Best results in the literature for regression with a prediction of fat in the Tecator dataset.

Algorithm/Dataset	Tecator Fat R^2	Tecator Fat RMSE
Partial Least Square Regression	0.95	2.96
Tree Regression	0.95	2.92
Random Forest	0.97	2.10
DSRegPSOP	0.99	1.29

3.3. Statistical Proofing Comparing DSRegPSO against Other Algorithms

To assess the effectiveness of DSRegPSOP in comparison to various algorithms on different datasets, we conducted the Kruskal-Wallis test. This non-parametric test enables the comparison of multiple independent groups to ascertain whether they stem from the same distribution [71]. RMSE values allow the Kruskal-Wallis test to be obtained to pinpoint any noteworthy disparities in performance, as shown in Table 29.

Table 29. Kruskal-Wallis Statistic Test.

Dataset	Kruskal-Wallis Test Statistic Value	*p*-Value	Significant Differences
Airfoil	6	0.423190081	No
Concrete	5	0.415880187	No
Cooling Load	6	0.423190081	No
Yachts Sailing Resistance	6	0.423190081	No
Geographical Location of Music	5	0.415880187	No
Tecator Fat	3	0.391625176	No

The results clearly indicate that, in all datasets analyzed, the *p*-values from the Kruskal-Wallis test exceeded 0.05, demonstrating no significant differences in performance (measured by RMSE) among the compared algorithms. In essence, all algorithms demonstrated statistically similar performance across the datasets. This finding strongly supports the notion that our algorithm, DSRegPSOP, is competitive and performs comparably to other state-of-the-art algorithms in the evaluated datasets.

4. Conclusions

In this work, we introduce DSRegPSOP, an algorithm developed for automatic programming to address the limitations of PSO in PSP, which was constrained to theoretical applications as it was prone to stagnation in high-dimensional spaces with numerous local optimal regions like those in automatic programming search spaces.

DSRegPSOP focuses on avoiding stagnation in local optimal regions in high-dimensional spaces by thoroughly exploring the search space and reinvigorating the swarm using dynamical sphere regrouping and momentum conservation effects inherited from DSRegPSO.

The models created using DSRegPSOP effectively transform the particle positions into intricate model equations. This is achieved by generating sets of equations organized into multiple levels with various parentheses, operators, variables, and numeric coefficients, which are dependent on the particle positions.

We evaluated DSRegPSOP using two different methods. First, we created a benchmark with 10 complex surfaces in a three-dimensional space to visually assess the performance of DSRegPSOP while registering R^2, MSE, MAE, and MAPE. These surfaces were similar to those used in numerical optimization tests such as the CEC'13 benchmark. We chose this approach because there is no universally accepted test for evaluating algorithms in symbolic regression. Our intention is for other researchers to compare their algorithms with DSRegPSOP using this first evaluation as a reference point.

We found in this test that DSRegPSOP successfully optimized 8 of the 10 functions with remarkable results with an R^2 near 1.0, and functions 3 and 4 being more complex registered $R^2 = 6.21 \times 10^{-1}$ and $R^2 = 8.04 \times 10^{-1}$, respectively. However, the complexity of this test can be upscaled by increasing the search space dimensionality in each surface as it occurs in numerical optimization benchmarks.

The second evaluation method considers that algorithms in symbolic regression are assessed using diverse datasets. Therefore, our second approach tests and compares DSRegPSOP with other algorithms using six datasets of real-world problems found in the literature. These datasets include the Airfoil, Concrete Strength, Cooling Load, Yachts Sailing Resistance, Geographical Location of Music, and Tecator Fat of Meals.

After obtaining the RMSE achieved with the DSRegPSOP regression models, we conducted the Kruskal-Wallis test. The results of this statistical test show that DSRegPSOP has comparable performance to other machine learning algorithms such as artificial neural networks, support vector machines, and random forests. However, unlike these algorithms, which optimize numerical parameters defined by humans (e.g., layer weights and biases in artificial neural networks), DSRegPSOP creates a specific model equation entirely on its own. Furthermore, DSRegPSOP effectively addresses the PSO limitations on PSP, obtaining comparable average performance to other algorithms. This represents a significant advancement, as previous PSP implementations were unable to achieve this level of performance. In addition, DSRegPSOP combines the advantages of automatic programming for symbolic regression with the lower-order computations of PSO and its parallelism, which, as a consequence, offers a compelling solution for regression modeling tasks.

In addition, the DSRegPSOP-generated mathematical models yield concise and comprehensible equations, facilitating easier sharing and analysis compared to other machine learning algorithms that optimize several numerical parameters in complex structures.

Future Work

DSRegPSOP leverages the parallelization capabilities of PSO, allowing it to take advantage of matrix parallel computing functionalities described in Matrix-Based Evolutionary Computation. As a result, future improvements in DSRegPSOP will focus on enhancing its performance and parallel computing capabilities.

Author Contributions: M.M.R., conceptualization and methodology; M.M.R. and C.G.-M., validation; formal analysis and investigation; all authors, experimentation and results; M.M.R. and C.G.-M., writing—original draft; all authors, writing—review and editing, visualization, and supervision; M.M.R. project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this research is publicly available and can be accessed at the following URL: https://github.com/mmrMontes/DSRegPSOP. accessed on 20 September 2024.

Acknowledgments: We acknowledge the support, time, and space for experimentation in Unidad Académica de Ciencia y Tecnología de la Luz y la Materia, Universidad Autónoma de Zacatecas, Campus Siglo XXI, Zacatecas 98160. We also thank CONAHCyT for its support in the scholarship Estancias Posdoctorales México 2022(1) (CVU: 471898).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Montes Rivera, M.; Padilla Díaz, A.; Ponce Gallegos, J.C.; Canul-Reich, J.; Ochoa Zezzatti, A.; Meza de Luna, M.A. Performance of Human Proposed Equations, Genetic Programming Equations, and Artificial Neural Networks in a Real-Time Color Labeling Assistant for the Colorblind. In *Advances in Soft Computing*; Martínez-Villaseñor, L., Batyrshin, I., Marín-Hernández, A., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2019; Volume 11835, pp. 557–578. ISBN 978-3-030-33748-3.
2. Middela, M.S.; Ramadurai, G. Modelling Urban Freight Generation Using Linear Regression and Proportional Odds Logit Models. *Transp. Policy* **2024**, *148*, 145–153. [CrossRef]

3. Li, L.; Luo, Z.; Miao, F.; Du, L.; Wang, K. Prediction of Product Yields from Lignocellulosic Biomass Pyrolysis Based on Gaussian Process Regression. *J. Anal. Appl. Pyrolysis* **2024**, *177*, 106295. [CrossRef]
4. Alahmer, A.; Alahmer, H.; Handam, A.; Rezk, H. Environmental Assessment of a Diesel Engine Fueled with Various Biodiesel Blends: Polynomial Regression and Grey Wolf Optimization. *Sustainability* **2022**, *14*, 1367. [CrossRef]
5. Muftah, M.N.; Faudzi, A.A.M.; Sahlan, S.; Shouran, M. Modeling and Fuzzy FOPID Controller Tuned by PSO for Pneumatic Positioning System. *Energies* **2022**, *15*, 3757. [CrossRef]
6. Lokanan, M.E. Predicting Money Laundering Using Machine Learning and Artificial Neural Networks Algorithms in Banks. *J. Appl. Secur. Res.* **2024**, *19*, 20–44. [CrossRef]
7. Olvera-Gonzalez, E.; Rivera, M.M.; Escalante-Garcia, N.; Flores-Gallegos, E. Modeling Energy LED Light Consumption Based on an Artificial Intelligent Method Applied to Closed Plant Production System. *Appl. Sci.* **2021**, *11*, 2735. [CrossRef]
8. Fleck, P.; Werth, B.; Affenzeller, M. Population Dynamics in Genetic Programming for Dynamic Symbolic Regression. *Appl. Sci.* **2024**, *14*, 596. [CrossRef]
9. Sarkar, B.; Dutta, S.; Lahiri, S.K. Multigene Genetic Programming Approach for Modelling and Optimisation of Removal of Heavy Metals from Ash Pond Water Using Cyanobacterial-Microalgal Consortium. *Indian Chem. Eng.* **2024**, 1–19. [CrossRef]
10. Zeiträg, Y.; Rui Figueira, J.; Figueira, G. A Cooperative Coevolutionary Hyper-Heuristic Approach to Solve Lot-Sizing and Job Shop Scheduling Problems Using Genetic Programming. *Int. J. Prod. Res.* **2024**, *62*, 5850–5877. [CrossRef]
11. Niazkar, H.R.; Moshari, J.; Khajavi, A.; Ghorbani, M.; Niazkar, M.; Negari, A. Application of Multi-Gene Genetic Programming to the Prognosis Prediction of COVID-19 Using Routine Hematological Variables. *Sci. Rep.* **2024**, *14*, 2043. [CrossRef]
12. Bahadori, M.K.; Shokouhi, M.; Golhosseini, R. Measurements of Density and Viscosity of Carbon Dioxide-Loaded and -Unloaded Nano-Fluids: Experimental, Genetic Programming and Physical Interpretation Approaches. *Chem. Eng. J. Adv.* **2024**, *18*, 100600. [CrossRef]
13. Huang, Z.; Zhang, F.; Mei, Y.; Zhang, M. An Investigation of Multitask Linear Genetic Programming for Dynamic Job Shop Scheduling. In *Genetic Programming*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2022; Volume 13223, pp. 162–178. [CrossRef]
14. Zhang, F.; Mei, Y.; Nguyen, S.; Zhang, M. Survey on Genetic Programming and Machine Learning Techniques for Heuristic Design in Job Shop Scheduling. *IEEE Trans. Evol. Comput.* **2024**, *28*, 147–167. [CrossRef]
15. Huang, Z.; Mei, Y.; Zhong, J. Semantic Linear Genetic Programming for Symbolic Regression. *IEEE Trans. Cybern.* **2024**, *54*, 1321–1334. [CrossRef] [PubMed]
16. Humberto Velasco Arellano, M.M.R.; Mendoza, J.E.G. Sensor Linearization Using Linear Genetic Programming. In *Robótica y Computaciión, Retos y Perspectivas*; Iliana Castro Liera, M.C.L., Ed.; Instituto Tecnológico de la Paz: La Paz, Mexico, 2018; pp. 117–122. ISBN 978-607-97128-7-7.
17. Arellano, H.V.; Rivera, M.M. Forward Kinematics for 2 DOF Planar Robot Using Linear Genetic Programming. *Res. Comput. Sci.* **2019**, *148*, 123–133. [CrossRef]
18. Hu, T.; Ochoa, G.; Banzhaf, W. Phenotype Search Trajectory Networks for Linear Genetic Programming. *arXiv* **2022**, arXiv:2211.08516. [CrossRef]
19. Miralavy, I.; Banzhaf, W. Spatial Genetic Programming. In *Genetic Programming*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2023; Volume 13986, pp. 260–275. [CrossRef]
20. Huang, Z.; Mei, Y.; Zhang, F.; Zhang, M. Graph-Based Linear Genetic Programming: A Case Study of Dynamic Scheduling. In Proceedings of the GECCO '22: Genetic and Evolutionary Computation Conference, GECCO '22: Genetic and Evolutionary Computation Conference, Boston, MA, USA, 9–13 July 2022; pp. 955–963. [CrossRef]
21. Lensen, A.; Xue, B.; Zhang, M. GPGC: Genetic Programming for Automatic Clustering Using a Flexible Non-Hyper-Spherical Graph-Based Approach. In Proceedings of the GECCO '17: Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 15–19 July 2017; pp. 449–456. [CrossRef]
22. Alrawi, A.W.; Awad, K.R.; Alakidi, A.M.J. The Impact of Oil Price Volatility on the Economic Development: The Linear Programming Method Study. *J. Gov. Regul.* **2023**, *12*, 361–368. [CrossRef]
23. Olmo, J.L.; Romero, J.R.; Ventura, S. Swarm-Based Metaheuristics in Automatic Programming: A Survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2014**, *4*, 445–469. [CrossRef]
24. Sheta, A.; Abdel-Raouf, A.; Fraihat, K.M.; Baareh, A. Evolutionary Design of a PSO-Tuned Multigene Symbolic Regression Genetic Programming Model for River Flow Forecasting. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 806–814. [CrossRef]
25. Sotto, L.F.D.P.; Kaufmann, P.; Atkinson, T.; Kalkreuth, R.; Basgalupp, M.P. A Study on Graph Representations for Genetic Programming. In Proceedings of the GECCO '20: Genetic and Evolutionary Computation Conference, Cancún, Mexico, 8–12 July 2020; pp. 931–939. [CrossRef]
26. Françoso Dal Piccol Sotto, L.; Kaufmann, P.; Atkinson, T.; Kalkreuth, R.; Porto Basgalupp, M. Graph Representations in Genetic Programming. *Genet. Program. Evolvable Mach.* **2021**, *22*, 607–636. [CrossRef]
27. Salehi-Abari, A.; White, T. The Uphill Battle of Ant Programming vs. Genetic Programming. In *Proceedings of the International Joint Conference on Computational Intelligence (IJCCI 2009)—ICEC*; SciTePress: Setúbal, Portugal, 2016; pp. 171–176. [CrossRef]
28. Piotrowski, A.P.; Napiorkowski, M.J.; Napiorkowski, J.J.; Rowinski, P.M. Swarm Intelligence and Evolutionary Algorithms: Performance versus Speed. *Inf. Sci.* **2017**, *384*, 34–85. [CrossRef]

29. Bansal, R.; Khanesar, M.A.; Branson, D. Ant Colony Optimization Algorithm for Industrial Robot Programming in a Digital Twin. In Proceedings of the 2019 25th International Conference on Automation and Computing (ICAC), Lancaster, UK, 5–7 September 2019. [CrossRef]
30. Moeini, R.; Afshar, M.H. Hybridizing Ant Colony Optimization Algorithm with Nonlinear Programming Method for Effective Optimal Design of Sewer Networks. *Water Environ. Res.* **2019**, *91*, 300–321. [CrossRef]
31. Togelius, J.; De Nardi, R.; Moraglio, A. Geometric PSO + GP = Particle Swarm Programming. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008; pp. 3594–3600.
32. O'Neill, M.; Brabazon, A. Grammatical Swarm: The Generation of Programs by Social Programming. *Nat. Comput.* **2006**, *5*, 443–462. [CrossRef]
33. Montes Rivera, M.; Guerrero-Mendez, C.; Lopez-Betancur, D.; Saucedo-Anaya, T. Dynamical Sphere Regrouping Particle Swarm Optimization: A Proposed Algorithm for Dealing with PSO Premature Convergence in Large-Scale Global Optimization. *Mathematics* **2023**, *11*, 4339. [CrossRef]
34. Skiena, S.S.; Revilla, M.A. *Programming Challenges*; Springer: New York, NY, USA, 2003; ISBN 978-0-387-00163.
35. Diáz, H.; Sala, A.; Armesto, L. A Linear Programming Methodology for Approximate Dynamic Programming. *Int. J. Appl. Math. Comput. Sci.* **2020**, *30*, 363–375. [CrossRef]
36. Bilal; Rani, D.; Pant, M.; Jain, S.K. Dynamic Programming Integrated Particle Swarm Optimization Algorithm for Reservoir Operation. *Int. J. Syst. Assur. Eng. Manag.* **2020**, *11*, 515–529. [CrossRef]
37. Liu, S.; Wang, P.; Xu, Z.; Feng, Z.; Zhang, C.; Wang, J.; Chen, C. Hydropower Unit Commitment Using a Genetic Algorithm with Dynamic Programming. *Energies* **2023**, *16*, 5842. [CrossRef]
38. Liu, D.; Xue, S.; Zhao, B.; Luo, B.; Wei, Q. Adaptive Dynamic Programming for Control: A Survey and Recent Advances. *IEEE Trans. Syst. Man. Cybern. Syst.* **2021**, *51*, 142–160. [CrossRef]
39. Shi, Y.; Eberhart, R. Modified Particle Swarm Optimizer. In Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73. [CrossRef]
40. Sedighizadeh, D.; Masehian, E.; Sedighizadeh, M.; Akbaripour, H. GEPSO: A New Generalized Particle Swarm Optimization Algorithm. *Math. Comput. Simul.* **2021**, *179*, 194–212. [CrossRef]
41. Pluhacek, M.; Senkerik, R.; Viktorin, A.; Kadavy, T. PSO with Attractive Search Space Border Points. In *Artificial Intelligence and Soft Computing*; Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2017; Volume 10246, pp. 665–675. [CrossRef]
42. Evers, G.I.; Ghalia, M. Ben Regrouping Particle Swarm Optimization: A New Global Optimization Algorithm with Improved Performance Consistency across Benchmarks. In Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11–14 October 2009; pp. 3901–3908. [CrossRef]
43. Li, F.; Yue, Q.; Liu, Y.; Ouyang, H.; Gu, F. A Fast Density Peak Clustering Based Particle Swarm Optimizer for Dynamic Optimization. *Expert. Syst. Appl.* **2024**, *236*, 121254. [CrossRef]
44. Akan, Y.Y.; Herrmann, J.M. Stability, Entropy and Performance in PSO. In Proceedings of the GECCO '23 Companion: Companion Conference on Genetic and Evolutionary Computation, Lisbon, Portugal, 15–19 July 2023; pp. 811–814. [CrossRef]
45. Tsujimoto, T.; Shindo, T.; Jin'no, K. The Neighborhood of Canonical Deterministic PSO. In Proceedings of the Evolutionary Computation (CEC), 2011 IEEE Congress, New Orleans, LA, USA, 5–8 June 2011; pp. 1811–1817.
46. La Cava, W.; Orzechowski, P.; Burlacu, B.; de França, F.O.; Virgolin, M.; Jin, Y.; Kommenda, M.; Moore, J.H. Contemporary Symbolic Regression Methods and Their Relative Performance. *arXiv* **2021**, arXiv:2107.14351. [CrossRef]
47. Home—UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/ (accessed on 7 August 2024).
48. Kaggle: Your Home for Data Science. Available online: https://www.kaggle.com/ (accessed on 7 August 2024).
49. Romano, J.D.; Le, T.T.; La Cava, W.; Gregg, J.T.; Goldberg, D.J.; Chakraborty, P.; Ray, N.L.; Himmelstein, D.; Fu, W.; Moore, J.H. PMLB v1.0: An Open-Source Dataset Collection for Benchmarking Machine Learning Methods. *Bioinformatics* **2022**, *38*, 878–880. [CrossRef]
50. GitHub—Cavalab/Srbench: A Living Benchmark Framework for Symbolic Regression. Available online: https://github.com/cavalab/srbench (accessed on 7 August 2024).
51. Airfoil Self-Noise—UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/dataset/291/airfoil+self+noise (accessed on 7 August 2024).
52. Brooks, T.F.; Stuart, D.; Marcolini, M.A. NASA Reference Publication 1218 Airfoil Self-Noise and Prediction. 1989. Available online: https://ntrs.nasa.gov/citations/19890016302 (accessed on 20 September 2024).
53. Yeh, I.C. Modeling of Strength of High-Performance Concrete Using Artificial Neural Networks. *Cem. Concr. Res.* **1998**, *28*, 1797–1808. [CrossRef]
54. Tsanas, A.; Xifara, A. Accurate Quantitative Estimation of Energy Performance of Residential Buildings Using Statistical Machine Learning Tools. *Energy Build.* **2012**, *49*, 560–567. [CrossRef]
55. Energy Efficiency—UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/dataset/242/energy+efficiency (accessed on 7 August 2024).
56. Yacht Hydrodynamics—UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/dataset/243/yacht+hydrodynamics (accessed on 7 August 2024).

57. Zhou, F.; Claire, Q.; King, R.D. Predicting the Geographical Origin of Music. In Proceedings of the 2014 IEEE International Conference on Data Mining, Shenzhen, China, 14–17 December 2014; pp. 1115–1120. [CrossRef]
58. Geographical Origin of Music—UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/dataset/315/geographical+original+of+music (accessed on 7 August 2024).
59. Fischer, S.; Harutyunyan, L.; Feurer, M.; Bischl, B. OpenML-CTR23—A Curated Tabular Regression Benchmarking Suite 2023. Available online: https://openreview.net/pdf?id=HebAOoMm94 (accessed on 20 September 2024).
60. Borggaard, C.; Thodberg, H.H. Optimal Minimal Neural Interpretation of Spectra. *Anal. Chem.* **1992**, *64*, 545–551. [CrossRef]
61. Liao, T.; Stutzle, T. Benchmark Results for a Simple Hybrid Algorithm on the CEC 2013 Benchmark Set for Real-Parameter Optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, 20–23 June 2013; pp. 1938–1944. [CrossRef]
62. Li, X.; Tang, K.; Omidvar, M.N.; Yang, Z.; Qin, K.; China, H. Benchmark Functions for the CEC 2013 Special Session and Competition on Large-Scale Global Optimization. *Gene* **2013**, *7*, 8.
63. de Franca, F.O.; Aldeia, G.S.I. Interaction-Transformation Evolutionary Algorithm for Symbolic Regression. *Evol. Comput.* **2019**, *29*, 367–390. [CrossRef] [PubMed]
64. Redonnet, S.; Bose, T.; Seth, A.; Li, L.K.B. Airfoil Self-Noise Prediction Using Deep Neural Networks. *Eng. Anal. Bound. Elem.* **2024**, *159*, 180–191. [CrossRef]
65. Claudinei-Daitx/Airfoil-Self-Noise-Prediction: Machine Learning Course SCS 3253. Available online: https://github.com/claudinei-daitx/airfoil-self-noise-prediction (accessed on 23 August 2024).
66. Elhishi, S.; Elashry, A.M.; El-Metwally, S. Unboxing Machine Learning Models for Concrete Strength Prediction Using XAI. *Sci. Rep.* **2023**, *13*, 19892. [CrossRef]
67. Bilimleri, F.; Dergisi, E.; Makalesi, A.; Akgundogdu, A. Comparative Analysis of Regression Learning Methods for Estimation of Energy Performance of Residential Structures. *Erzincan Univ. J. Sci. Technol.* **2020**, *13*, 600–608. [CrossRef]
68. Baressi Šegota, S.; Anđelić, N.; Kudláček, J.; Čep, R. Artificial Neural Network for Predicting Values of Residuary Resistance per Unit Weight of Displacement. *Pomor. Zb.* **2019**, *57*, 9–22. [CrossRef]
69. Lazarevska, E. A Neuro-Fuzzy Model of the Residuary Resistance of Sailing Yachts. In Proceedings of the 2016 IEEE 8th International Conference on Intelligent Systems (IS), Sofia, Bulgaria, 4–6 September 2016; pp. 173–179. [CrossRef]
70. Kucheryavskiy, S. Analysis of NIR Spectroscopic Data Using Decision Trees and Their Ensembles. *J. Anal. Test.* **2018**, *2*, 274–289. [CrossRef]
71. Kruskal-Wallis Test. In *The Concise Encyclopedia of Statistics*; Springer: New York, NY, USA, 2008; pp. 288–290. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.