Neuro-PSO Algorithm for Large-scale Dynamic Optimization

Mohamed Radwan[a], Saber Elsayed[b,a], Ruhul Sarker[a], Daryl Essam[a], Carlos Coello Coello[c,d]

[a]*School of Engineering and Information Technology, University of New South Wales at Canberra, ACT, 2612, Australia*
[b]*Department of Computer Science, University of Sharjah, United Arab Emirates*
[c]*Departamento de Computación, CINVESTAV-IPN, Mexico City, 07360, Mexico*
[d]*Faculty of Excellence with School of Engineering and Sciences, Tecnologico de Monterrey, Monterrey, Mexico*

Abstract

Over the last few decades, dynamic optimization and large-scale optimization have been two challenging research topics. In this context, dynamic optimization with high dimensionality is undoubtedly another important research topic. For such a combined problem, this paper develops: (1) an algorithm that incorporates problem decomposition to deal with high dimensionality, (2) a search algorithm for optimization, and (3) a prediction strategy to deal with dynamic changes. Firstly, a decomposition method is introduced to divide the problem into multiple subproblems based on the level of interactions among the decision variables. For optimization, a multi-population search algorithm is proposed, where each subpopulation evolves individually. Finally, a machine learning-based prediction strategy is developed to learn information from historical solutions and predict some solutions that may be useful for the new environment. The proposed algorithm is tested using the generalised moving peaks benchmark problems. The results show that the proposed algorithm can find better solutions than existing approaches.

Keywords: Evolutionary dynamic optimization, Large-scale dynamic optimization problems, Tracking moving optimum, Generalized moving peaks benchmark

1. Introduction

In many real-world decision-making processes, such as transportation, production planning and scheduling, a decision is made by solving their representative optimization models repeatedly at regular intervals [1, 2]. The interval length varies depending on the nature of the problems and their processing needs. If the interval is too short, the problem may be considered as continuous. In these problems, some changes may occur in one or more of the objective function(s), constraint function(s), constraint right-hand side(s) and/or variable bounds as time passes in some or all the selected intervals [3, 4, 5]. These are known as dynamic optimization problems (DOPs). Unlike stationary optimization, which aims to find a unique optimum solution, the optimum solutions in dynamic problems change over time. Thus, in this case, the aim is to identify and track the changing optima over time.

As of the literature, evolutionary algorithms (EAs) are well-accepted approaches for solving complex static optimization problems. However, the population of solutions usually lose their diversity after they converge to an optimum (i.e., their exploration capability). In this case, specialized mechanisms are needed to let them adapt to any change that occurs in the environment [6, 7]. The most straightforward but ineffective method is to reset the optimizer when a change in the environment is detected. This method's limitation is that it does not transfer helpful information from one environment to the next.

Detecting the changes in a series of environments over time and locating the new optimum in each environment are two main tasks in solving DOPs. This has motivated researchers to propose different approaches for solving dynamic problems, such as diversity-based [8, 9, 10, 11], memory-based [12, 13], prediction-based [14, 15, 16] and multi-population approaches [4, 17]. It is worth mentioning that several dynamic optimization approaches are suitable for problems with a variety of changes [6, 18, 1]. Interestingly, most of these DOP approaches mainly deal with low-dimensional problems, whereas most real-world DOPs [19, 20, 21] are high-dimensional.

The computational cost of solving high-dimensional static problems is often high [22], and the performance of EAs deteriorates with an increasing number of decision variables. In some cases, the cost increase is exponential with respect to the dimensionality of the problem. Furthermore, EAs may get trapped in local optima [23]. To deal with the curse of dimensionality, researchers have applied cooperative co-evolution, which divides a large-scale optimization problem into a number of smaller subproblems that can be optimized cooperatively [24]. More details about problem decomposition for high-dimensional problems can be found in [25, 26, 27].

When solving high-dimensional problems with dynamic changes, the challenge is to design efficient algorithms capable of finding and tracking moving optima in a large search space within a limited time [28]. However, there is no well-accepted methodology that can efficiently solve dynamic large-scale optimization problems [25, 2]. Also, a problem's different environments may be correlated to each other, which means the solutions obtained from the current and past environments might be beneficial in optimizing future ones quickly. It is generally accepted that learning from previous solutions can play a pivotal role in speeding up the convergence of a new environment [29, 15].

Some attempts have been made to reuse previous observations for solving DOPs by either adopting previously found solutions in the new environment [12] or predicting the solution in new environments based on learning correlations among consecutive environments using machine learning and other statistical techniques [15]. For example, in [30], a Kalman filter has been integrated to determine the locations of new optima. Simões and Costa [31] used two prediction models: a linear regression one to estimate when a change would occur and Markov chains to predict changes. Hamza et al. [5] proposed a sensitive constraint detection mechanism to determine the appropriate movement of solutions (direction and amount of movement) for constrained problems under changing environments. However, learning a change from previous solutions is challenging, especially for large-scale problems [29].

Considering the above-mentioned needs, this paper introduces a novel approach for solving single-objective large-scale dynamic optimization problems (LSDOPs) using a three-phase algorithm consisting of problem decomposition, optimization, and machine learning-based prediction. The first phase focuses on dividing the LSDOP into smaller subproblems that are assigned to separate subpopulations for independent evolution in the second phase. The solutions from these subpopulations are then combined to find the overall solution to the problem. In the second phase, a search technique is proposed based on the concept of multi-population optimizer. In the final phase, a reaction-based method addresses the loss of diversity during environmental changes. Additionally, a neural network approach is developed to learn the movement of optima through environmental changes and predict solutions for future environments. This predicted information is then used as input to the search process in the second stage, thereby supporting the optimizer in adjusting to the new environment. Our key contribution lies on an effective integration and adaptation of these techniques to address the challenges in solving large-scale dynamic optimization problems. To the best of our knowledge, no existing methodologies for large-scale dynamic optimization integrated these techniques in the manner we proposed in this paper. Evaluation of the proposed algorithm through the solution of LSDOP test problems demonstrates its superior performance compared to existing approaches, making it a novel and promising solution approach for tackling LSDOPs.

The rest of this paper is organized as follows. In Section 2, the background and related literature is reviewed. In Section 3, the proposed approach is explained in detail. In Section 4, the experimental results are discussed and the performance of the proposed method is analyzed. Finally, our conclusions and some future research directions are provided in Section 5.

2. Previous Related Work

This paper deals with LSDOPs. In this section, the most representative studies on DOPs will be reviewed, followed by an overview of the CC strategy that has been used for large-scale problems. Then, a brief introduction to Neural Networks will be given.

2.1. Dynamic Optimization Algorithms

Under certain assumptions, a DOP can be defined as a series (or collection) of stationary problem instances that need to be optimized. The dynamism is expressed by the change frequency and magnitude of the environmental changes that may occur in the objective function(s), constraints, variable bounds and variables.

The definition of optima in DOPs and their search approaches are different from those used in static optimization problems. In the literature, EAs and swarm intelligence techniques that are applied to DOPs mainly deal with environmental changes through diversity-driven mechanisms and information obtained from past environments. Some of the DOP approaches are briefly discussed in the following subsections.

2.1.1. Diversity-based approaches

If a change in an environment is detected, in population-based methods, this approach introduces more diversity into the population through either adding random individuals [32], increasing the mutation rate [33], changing the mutation step-size [34] or transferring individuals between subpopulations [32]. However, by increasing diversity using the above methods, knowledge gained from previous environments may be lost and that may slow down the convergence of the current environment. Also, it is hard to determine the necessary level of diversity [7].

Some approaches introduce random immigrants into the population [35], when the diversity drops below a predefined threshold [36], or to avoid individuals from getting too close to each other [11]. Although these algorithms have been found effective in slow and large changes, they may be slow in convergence and can be less effective for small changes [37].

2.1.2. Memory-based approaches

In dynamic optimization, as there are some similarities with the past environment(s), it may be useful to share knowledge from previous environments, specifically for the problems that follow periodical or recurrent changes. By incorporating a memory component into the optimizer, computational time could be saved, and diversity can be maintained. The memory can be either implicitly integrated (as a redundant representation) or explicitly maintained (by archiving good solutions and information from previous environments). However, they may only be helpful when optima reappear in their earlier locations; otherwise, the previously stored information may become redundant [1].

Xu et al. [38] introduced a memory-enhanced dynamic multi-objective evolutionary algorithm that leverages decomposition. This approach divides a dynamic multi-objective optimization problem (DMOP) into multiple dynamic scalar optimization subproblems, enabling their co-evolution. To address environmental changes effectively, the algorithm incorporates a subproblem-based memory scheme that selectively stores high-quality solutions from previous environments and reuses them when needed.

2.1.3. Prediction-based approaches

If environmental changes follow a specific pattern, it might be useful to learn these patterns to predict future changes. As previously mentioned, Kalman filters have been incorporated to predict the location of optima in new environments [30]. In another case, in a dynamic combinatorial optimization problem, two prediction models, namely linear regression and Markov chains, have been adopted [31]. The former was used to estimate the time when a change in the environment will happen, while the latter estimates which environment will arise in the following change. Also, interaction-based prediction methods have been developed to forecast task scheduling in cooperative multi-robot systems, enabling improved collaboration and adaptability [39]. These prediction methods were beneficial as the algorithm could find the next optima quickly. However, it is unlikely to obtain good solutions for irregular changes.

2.1.4. Multi-population approaches

This approach is considered the most efficient and popular category for solving DOPs. In this approach, certain subpopulations may be responsible for exploring for the global optimum, whereas others may track any potential changes [40]. In some other approaches, a parent subpopulation is used to explore the new peaks, and some child subpopulations are used to follow the found peaks [41]. However, there is little guidance on selecting the number and sizes of subpopulations, which would be useful, as an excessive number of subpopulations may impede the efficiency of the search process [11].

2.2. Cooperative Coevolution (CC)

As shown in Algorithm 1, CC was initially developed by [24]. In the process, firstly, a high-dimensional optimization problem is decomposed into several subproblems, each with fewer decision variables. Each subproblem is then allowed to evolve with its own subpopulation for a certain number of generations in a round-robin fashion, where computational resources are shared evenly across all subproblems [24]. Finally, cooperative action is required, which involves exchanging information among all subproblems and then merging their solutions to update a context vector (because a solution consists of the best solutions from all subproblems) [42]. The CC framework helps to speed up the convergence of the search process, to enhance the search capabilities and to maintain the population's diversity.

In the CC domain, the decomposition of a problem is based on a variable grouping approach, which is generally classified as either static [43], random [44] or variable interaction grouping [45, 46, 47]. In static and random grouping methods, the arrangement of an n-dimensional problem is decomposed into s subproblems of size k manually and randomly, respectively. Thus, a

Algorithm 1 Cooperative Co-evolution

```
1: Randomly generate initial population ($P$);
2: $[V_1, ..., V_{N_P}] \leftarrow evaluate(f(\overrightarrow{X}_i), \forall i = 1, ..., NP$;
3: $(\overrightarrow{X}_{best}, f_{best}) \leftarrow P(min([V_1, ..., V_{N_P}]))$;
4: initialize context vector;
5: $S = \{s_1, ..., s_k\} \leftarrow decompose(f(x), dim)$;
6: $cycle \leftarrow 0$;
7: **while** $FEs < FEs_{max}$ **do**
8:     $cycle \leftarrow cycle + 1$;
9:     **for** $i = 1 : k$ **do**
10:        evaluate subproblem $s_i$, within the context vector;
11:        update context vector;
12:        $FEs = FEs + used\_FEs$;
13:    **end for**
14: **end while**
```

number of subproblems and their sizes need to be specified. In variable interaction grouping, the interactions among the decision variables are the key factor for grouping. This grouping is conducted as a pre-processing phase that consumes a portion of the allocated resources.

The round-robin technique of the CC method treats all subproblems equally, regardless of their level of contribution [24]. However, this could result in a significant waste of computational resources since the lower contributing subproblems would use more resources than required. To deal with this issue, the resource allocation approach based on each subproblem's contribution is suggested [48, 49]. Here, more emphasis is given to the subproblems that contribute more to the overall fitness value [48].

Only a few studies use the CC strategy to effectively handle the curse of dimensionality in LSDOPs. The algorithm in [50] uses the concept of divide and conquer to solve high-dimensional moving peaks problems. It incorporates the differential grouping method [51] to divide the problem into independent subproblems before using the species-based PSO [52] to evolve each subproblem's swarm. When an environmental change occurs, a memory strategy is conducted to exploit the previous information. However, it is assumed that prior knowledge about the number of peaks in each subproblem and the number of generations between consecutive changes are

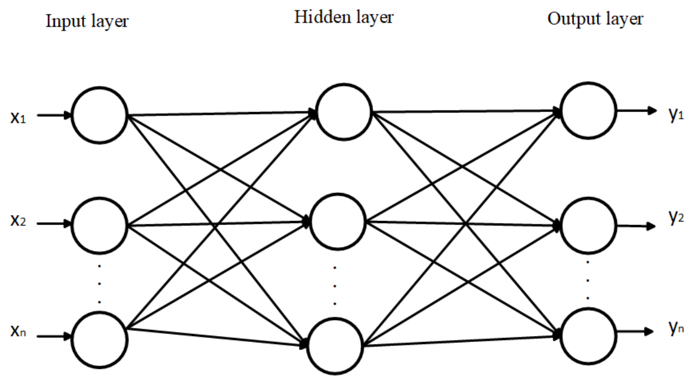

Fig. 1: The architecture of a neural network

available. Yazdani et al. [28] proposed a CC multi-population framework for solving LSDOPs by using a composite MPB suite generator with up to 200 variables. The framework divides the problem into lower-dimensional subproblems and controls the resource allocation to these subproblems for the following multiple moving optima. However, they solved fully nonseparable problems without obviously exploitable modularity.

2.3. Neural Networks

Researchers have increasingly utilized statistical approaches, including neural networks, as effective tools for solving complex optimization problems [53, 54]. In the context of dynamic optimization, these methods are frequently employed for prediction. A neural network is a simplified representation of a biological neural network that retains functions in neurons and their connections, where it learns to execute useful functions through data-driven training [55]. Fig. 1 demonstrates the basic framework of a neural network. It comprises several layers, each containing several neurons and interconnections between them. The leftmost layer is denoted as the input layer, which accepts the input vector, the rightmost layer is called the output layer, which produces the output of the neural network, and the intermediate layers are referred to as hidden layers. Each neuron in the network has an activation function and a bias, and the weight between each neuron-to-neuron connection is represented by w. When detecting environmental changes, the input layer and the output layer contain the old and new environment's solutions, respectively.

3. Our Proposed approach

For solving LSDOPs, a new algorithm is proposed in this paper, as shown in Fig. 2 and Algorithm 2 , which has three major phases, namely: 1) decomposition, 2) optimization, and 3) learning and prediction. The proposed algorithm decomposes the large-scale problem, and the generated subproblems are evolved using a multi-population optimizer. A reaction-based method is employed to handle the diversity loss when an environmental change is detected. Details of the proposed algorithm are provided in the following sub-sections.

3.1. Decomposition Phase

The problem decomposition (or variable grouping) is conducted using a variable interaction method [45, 47] to ensure that each subcomponent contains a group of highly dependent variables and the interdependencies among subcomponents are minimal. This method identifies the interactions among the decision variables based on fitness variations after perturbation, which employs the DG2 [56] to decompose problems. It is an improved version of DG [51], with higher decomposition accuracy and a lower computational time. DG2 is a parameter free decomposition method which automatically calculates a different threshold for each pair by approximating the magnitude of the roundoff errors, as identifying variable interactions is not effective using only one threshold value, when dealing with imbalanced contributing components. DG2 calculates the lower bound $e_{\inf}$ and upper bound $e_{\sup}$ for computational errors. When comparing each pair of decision variables, if the value $\lambda = |\Delta^{(1)} - \Delta^{(2)}| < e_{\inf}$, it is considered zero, leading to the declaration of the pair as separable. To determine $\Delta^{(1)}$ and $\Delta^{(2)}$, it is essential to conduct the following evaluations to identify the interaction between the ath and bth dimensions:

$$\begin{cases} \Delta^{(1)} = f\left(\ldots, x'_a, \ldots\right) - f\left(x_1, \ldots, x_n\right) \\ \Delta^{(2)} = f\left(\ldots, x'_a, \ldots, x'_b, \ldots\right) - f\left(\ldots, x'_b, \ldots\right). \end{cases}$$

Conversely, if λ exceeds $e_{\sup}$, it is deemed a non-zero value, resulting in the pair being declared as interacting. For values of λ within the range $[e_{\inf}, e_{\sup}]$, ϵ is determined as a weighted average of the two bounds, considering the total number of instances identified as zeros and non-zeros. This decomposition process is conducted before the start of the optimization phase, which consumes a portion of the allocated resources.

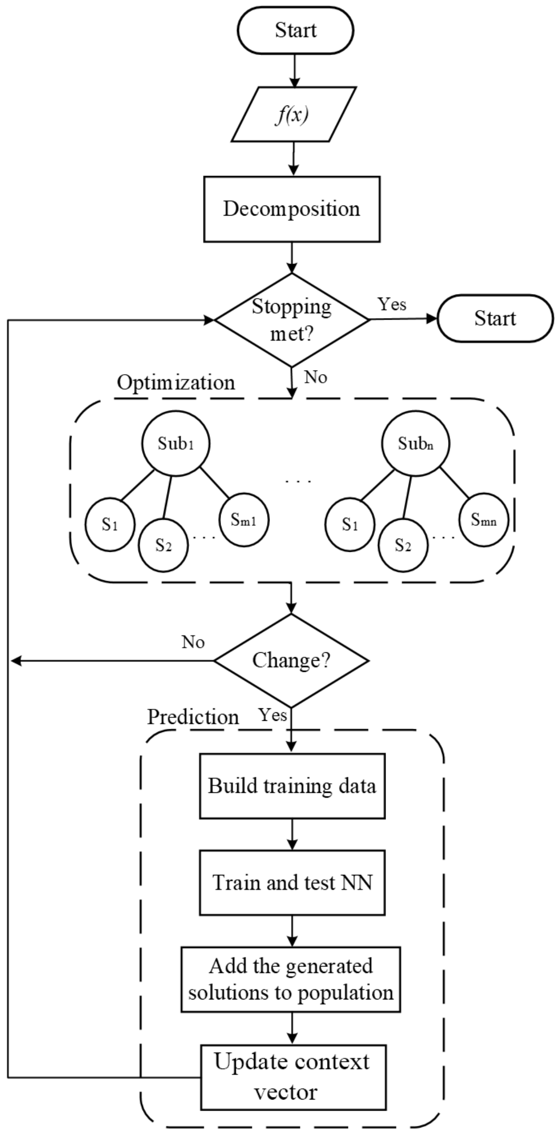

Fig. 2: General structure of the proposed algorithm

Algorithm 2 Proposed Algorithm

```
1: $components \leftarrow$ decomposition($f$);
2: Initialize an explorer population for each component;
3: Initialize the context vector;
4: while stopping condition is not met do
5:    for each component $c$ in components do
6:       $g_r^* \leftarrow$ optimize the explorer population;
7:       if explorer population converges to a peak then
8:          Convert it to an exploiter population;
9:          Create a new explorer;
10:      end if
11:      for each active exploiter $t$ do
12:         $g_t^* \leftarrow$ optimize the exploiter $t$;
13:         if diversity $< r_{deact}$ then
14:            Deactivate the $t^{th}$ exploiter;
15:         end if
16:      end for
17:   end for
18:   $H \leftarrow$ component with high progress;
19:   for each active exploiter $t$ in $H$ do
20:      $g_t^* \leftarrow$ optimize active trackers in $H$;
21:   end for
22:   for each component $c$ in components do
23:      $g^* \leftarrow$ optimize the best exploiter;
24:   end for
25:   if an environmental change is detected then
26:      Update context vector using best position of each population;
27:      Append $g_{past}^*$ and $g_{current}^*$ of all subpopulations to the training data;
28:      for each component $c$ in components do
29:         $net_c \leftarrow$ train neural network
30:         Predict promising solutions using the prediction model
31:         Increase diversity by predicted solutions and randomizing a portion of the population;
32:      end for
33:   end if
34: end while
```

3.2. Optimization Phase

As previously mentioned, one of the effective ways to address DOPs is by employing the multi-population approach [18], where an explorer subpopulation and several exploiter subpopulations are allocated to each subcomponent, (Algorithm 2, lines 5 to 17). The proposed algorithm incorporates a multi-population optimizer that iterates over subcomponents to find all peaks, tracks them, and determines the contribution of each subcomponent in enhancing the overall fitness value. Unlike the assumption in [57, 58], the number of peaks is not specified, and an explorer subpopulation is used to discover any uncovered peaks. The convergence of the explorer subpopulation is evaluated based on its diversity (i.e., the Euclidean distances between individuals [59, 60]), and if the diversity is less than a predefined threshold [28], the explorer subpopulation is assumed to have converged. Once converged, it is assumed that a promising region has been discovered. After that, exploiter subpopulations (including the converged explorer) are used in this region to track the peak, and another random explorer subpopulation will be generated. In lines 11 to 16 of Algorithm 2, an exploiter subpopulation undergoes deactivation when its diversity falls below a predetermined threshold, denoted as r_{deact}[28], calculated by evaluating the infinity norm distance between any pair of individuals within the population.

As the typical round-robin search strategy is not an effective way of dealing with the variable contribution of subproblems, contribution-based CC (CBCC) [61] is employed in this algorithm as it is more suitable for this type of problem [62]. Also, as inspired by the idea in [28], the resource allocation at both component and population levels is employed to save the computational budget due to inefficient design. This process is conducted by placing more emphasis (i.e., executing extra iterations) on the higher contributing subcomponent and the best exploiter subpopulation of each subcomponent , as described in Algorithm 2, lines 18–24.

3.3. Learning and prediction

In this phase, there are two tasks. Firstly, diversity loss due to environmental changes is addressed. To do this, at the start of each environment, once a change is identified, one of the individuals from the best-found region in the previous environment is selected. Then, some individuals are randomly selected around this region, with a radius of shift severity estimated based on each subpopulation's best-found global solutions at the end of the two previous environments [28].

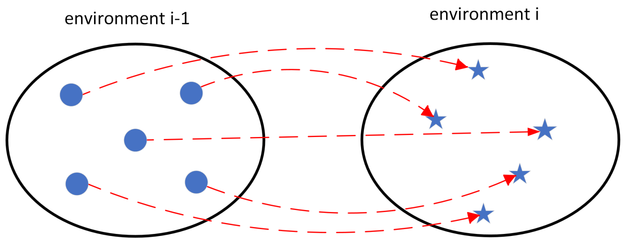

Fig. 3: Example of the movement of the best solution of the subpopulation in a subcomponent of DOP

Second, the solutions for the current environment are predicted using the solutions of the past environments. For this sake, a prediction model is integrated into the DOP optimizer. In the process, the final best solutions obtained by all subpopulations in each subproblem in each environment are archived. Then, a training data set is constructed by pairing these solutions, which represent the transitions of the subcomponents' optimum between any two consecutive environments. In the training data set, each row contains an input of one component that was optimum from a certain environment and an output, which is the optimum of the same element from the subsequent environment. Note that we use the subproblem solutions for each new environment from all the preceding environments (i.e., 1 to $t-1$) to build the training data set. An example of the movement of the best solution due to an environmental change is shown in Fig. 3, where solid circles represent the best solutions found by each subpopulation at environment $i-1$, while the stars are the best solutions at the environment i. The dashed arrows indicate the transitions of these solutions from environment $i-1$ to the environment i.

A neural network, which is a powerful tool to solve time series prediction problems, is utilized to enable learning from previous observations [55]. It uses the constructed training data to discover how solutions change between different environments. First, a suitable neural network architecture must be determined, such as the number of layers, neurons, and activation function. Then, for each component in the DOP, we utilize a three-layer neural network comprised of an input layer, a hidden layer, and an output layer. For an n-dimensional component, the input and output of the neural network will be a vector with n elements. All the networks use the Levenberg–Marquardt algorithm [63] for training with 100 iterations [64]. This is reasonable for dynamic optimization because of the time limitation in solving each environment.

After training, the learnt networks can be used to predict promising solutions for the new environment. The solutions for each component in the last environment $t-1$ are fed into the neural networks, and the outputs of all these networks form the prediction solutions. The promising solutions are then added to the previous group of individuals in the initial population, thereby assisting the optimizer in the new environment.

4. Experiments and Results

In this section, the experiments carried out to evaluate the performance of the proposed algorithm are discussed. Firstly, the experimental settings and performance measures are discussed. Then, the behavior of the proposed algorithm is tested by comparing the results with those obtained from the existing algorithms. Finally, the robustness of the proposed algorithm under different settings, including number of peaks, shift severity and change frequency, is analyzed.

4.1. Benchmark for Experimentation

In dynamic optimization, different test suites have been proposed to evaluate the efficacy of dynamic optimization algorithms [65, 66, 67, 68, 28, 69]. One of them is the moving peaks benchmark, which is considered the most popular and is employed in more than 70% of the DOP literature because of its ease of implementation and control [2]. Its landscapes consist of multiple peaks with randomly changing locations, heights and widths over time. The fitness of a solution x in a changing environment t is defined as:

$$f(\vec{x},t) = \underset{i\in\{1,\ldots,\#peaks\}}{max} \frac{h_i(t)}{1 + w_i(t)\sum_{j=1}^{D}(x_j - c_{i,j}(t))^2} \tag{1}$$

where x_j is the j^{th} dimension in a d-dimensional problem, $c_{i,j}(t)$ is the j^{th} dimension of the center position of the i^{th} peak. $h_i(t)$ and $w_i(t)$ are the height and width of the i^{th} peak, respectively.

For each peak, the height, width and center change from one environment to another as follows:

$$h_i(t) = h_i(t-1) + S_h.\mathcal{N}(0,1) \tag{2}$$

$$w_i(t) = w_i(t-1) + S_w.\mathcal{N}(0,1) \tag{3}$$

$$c_i(t) = c_i(t-1) + S\frac{\varepsilon}{\|\varepsilon\|} \tag{4}$$

where $\mathcal{N}(0,1)$ is a random number with mean 0 and variance 1, S_h, S_w and S are the height, width and length of the movement severities, respectively. In addition, ε is a vector of random values.

Generalized MPB (GMPB) [69] is another test suite whose landscapes are formed by combining multiple components. GMPB-generated components exhibit a wide range of characteristics, such as varying degrees of irregularities (smooth to extremely irregular), different types of modalities (unimodal and multimodal), as well as both symmetric and asymmetric features. In each instance of environmental change, the parameters of all components within the GMPB experience random dynamics. In addition, each component's degree of irregularity and variable interactions are controllable.

4.2. Experimental Settings

In order to evaluate the effectiveness of algorithms, we solved a set of 15 test functions which exhibit different characteristics, all of which were generated using the GMPB benchmark generator [70]. The test suite consists of functions with five distinct variable interaction structures, which were tested in spaces with dimensions of 50-D, 100-D, and 200- D (as indicated in Table 1). We adopted Particle Swarm Optimization (PSO) [71] as our component optimizer, since it has been widely used in the literature of dynamic optimization and has demonstrated its effectiveness [18]. The environmental change is explicitly detected, and the change frequency (i.e., the number of function evaluations for each environment) is set to $500 * D$. The average error of the best found solutions before each environmental change is employed as a performance metric. The results are based on 15 independent runs, each of which involves 30 environmental changes, and their best, median, mean, worst and standard errors are reported for comparison. To determine whether there are any significant differences among the algorithms, a non-parametric statistical comparison is performed using the Wilcoxon signed-rank test at a significance level of 0.05. The use of three symbols, namely $+$, -, and $\approx$, indicates whether the former algorithm is significantly better, worse, or not significantly different from the latter algorithm, respectively. Also, a Friedman ranking test [72] is used to rank all the algorithms according to their fitness values.

Table 1: The characteristics of the 15 GMPB Scenarios.

Fun	Dim	Nonseparable Components	Separable
f_1	50	2,3,5,6,6,8,10	10
f_2	50	2,3,5,5	35
f_3	50	2,2,3,5,5,5,5,8,10	0
f_4	50	-	50
f_5	50	50	0
f_6	100	2,2,3,5,5,6,6,8,8,10,10,15	20
f_7	100	2,2,3,3,5,5,10	70
f_8	100	2,2,2,2,3,3,5,5,5,5,5,5,8,8,10,10,20	0
f_9	100	-	100
f_{10}	100	100	0
f_{11}	200	2,2,3,5,5,6,6,8,8,10,10,15,20,20,30	50
f_{12}	200	2,3,5,10,20,30	130
f_{13}	200	2,2,2,3,5,5,5,5,5,8,8,10,10,10,20,20,30,50	0
f_{14}	200	-	200
f_{15}	200	200	0

4.3. Experimental Results

The proposed algorithm is compared with CTR [28] and TMMO [28], a monolithic multi-population framework that represents the typical approach to tracking multiple moving optima. TMMO does not explicitly rely on a divide-and-conquer mechanism; instead, it utilizes a large subpopulation for global search and several smaller subpopulations to track changes in identified peaks. Additionally, GCM-PSO [50], a hybrid particle swarm optimization algorithm, is considered. GCM-PSO decomposes the problem into sub-problems, optimizes them using species-based PSO with nearest-better clustering to divide the swarm into species, and incorporates a memory strategy. The experimental results obtained on GMPB are shown in Table 2.

The results obviously demonstrate that the proposed algorithm exhibits superior performance compared to the other algorithms on a large number of the functions. This clearly illustrates the benefit of integrating the multi-population approach with a prediction method. Also, decomposition-based algorithms (i.e., the proposed CTR and GCM-PSO ones) outperform TMMO on all fully and partially separable functions, clearly showing the importance of problem decomposition for handling LSDOPs. As shown in Table 2, the GCM-PSO algorithm demonstrates superior performance compared to other algorithms in some functions, particularly those that are nonseparable.

Based on the Wilcoxon test, Table 3 shows comparisons of the best, median and mean values obtained by the proposed and other algorithms using

Table 2: Results by the proposed algorithm, CTR, TMMO and GCM-PSO for the GMPB problems with change frequency = 500D.

Fun	Proposed algorithm					CTR					TMMO					GCM-PSO				
	Best	Median	Mean	Worst	STD	Best	Median	Mean	Worst	STD	Best	Median	Mean	Worst	STD	Best	Median	Mean	Worst	STD
f_1	5.04E+00	9.24E+00	1.02E+01	1.81E+01	1.04E+00	5.32E+00	1.08E+01	1.02E+01	1.67E+01	9.01E-01	3.45E+01	5.10E+01	5.09E+01	7.52E+01	3.03E+00	7.72E+00	1.18E+01	1.27E+01	2.07E+01	1.11E+00
f_2	1.29E+00	2.98E+00	2.81E+00	3.39E+00	1.42E-01	2.11E+00	3.09E+00	3.31E+00	5.88E+00	2.62E-01	2.00E+01	2.42E+01	2.46E+01	3.17E+01	7.52E-01	5.45E+00	7.01E+00	7.13E+00	8.94E+00	2.28E-01
f_3	4.30E+00	9.80E+00	9.54E+00	1.55E+01	8.46E-01	5.23E+00	9.73E+00	9.64E+00	1.39E+01	7.49E-01	3.85E+01	5.96E+01	5.94E+01	8.63E+01	3.86E+00	5.44E+00	1.31E+01	1.36E+01	3.95E+01	2.15E+00
f_4	3.14E-01	1.32E+00	2.91E+00	1.41E+01	9.88E-01	3.59E-01	1.53E+00	3.30E+00	1.54E+01	1.10E+00	1.48E+01	1.70E+01	1.69E+01	1.85E+01	2.35E-01	6.61E+00	7.36E+00	7.41E+00	8.32E+00	1.31E-01
f_5	2.48E+01	1.19E+02	1.45E+02	4.58E+02	2.97E+01	4.69E+01	1.83E+02	2.19E+02	6.29E+02	4.31E+01	1.53E+01	8.88E+01	1.21E+02	4.31E+02	2.69E+01	9.51E+00	4.63E+01	7.09E+01	2.83E+02	1.84E+01
f_6	1.27E+01	1.66E+01	2.81E+01	1.21E+02	7.94E+00	1.18E+01	1.74E+01	2.95E+01	1.95E+02	1.19E+01	8.25E+01	1.05E+02	1.05E+02	1.24E+02	3.23E+00	1.19E+01	2.15E+01	2.05E+01	3.15E+01	1.46E+00
f_7	2.55E+00	4.48E+00	5.75E+00	1.56E+01	8.79E-01	3.15E+00	4.48E+00	5.75E+00	2.06E+01	1.11E+00	2.87E+01	3.28E+01	3.30E+01	4.07E+01	9.14E-01	7.02E+00	8.31E+00	9.99E+00	2.37E+01	1.18E+00
f_8	1.40E+01	2.27E+01	2.42E+01	4.36E+01	2.30E+00	1.48E+01	2.23E+01	2.49E+01	3.86E+01	2.32E+00	1.01E+02	1.51E+02	1.44E+02	2.13E+02	9.63E+00	1.91E+01	3.17E+01	3.53E+01	7.37E+01	4.24E+00
f_9	3.67E-01	8.59E+00	8.46E+00	2.09E+01	1.93E+00	4.61E-01	9.65E+00	9.07E+00	2.14E+01	1.95E+00	1.95E+01	2.19E+01	2.19E+01	2.31E+01	2.29E-01	6.52E+00	7.19E+00	7.16E+00	7.82E+00	8.76E-02
f_{10}	7.09E+01	3.46E+02	4.12E+02	1.52E+03	9.00E+01	2.09E+02	8.08E+02	8.06E+02	1.52E+03	1.07E+02	6.44E+01	3.73E+02	4.53E+02	1.24E+03	8.46E+01	1.91E+01	1.04E+02	2.45E+02	8.94E+02	6.75E+01
f_{11}	2.81E+01	4.86E+01	7.21E+01	3.70E+02	2.18E+01	2.81E+01	5.17E+01	7.82E+01	4.20E+02	2.47E+01	1.48E+02	2.46E+02	2.36E+02	3.05E+02	1.49E+01	1.90E+01	5.86E+01	5.56E+01	9.12E+01	5.34E+00
f_{12}	1.78E+01	4.02E+01	6.71E+01	1.81E+02	1.52E+01	1.90E+01	5.31E+01	9.08E+01	2.75E+02	2.27E+01	5.81E+01	7.81E+01	9.19E+01	1.65E+02	8.69E+00	2.37E+01	3.75E+01	4.68E+01	9.60E+01	5.82E+00
f_{13}	4.01E+01	9.23E+01	1.07E+02	2.23E+02	1.33E+01	5.86E+01	1.30E+02	1.39E+02	2.29E+02	1.40E+01	2.52E+02	3.36E+02	3.84E+02	6.88E+02	2.98E+01	4.93E+01	1.11E+02	1.09E+02	2.23E+02	1.24E+01
f_{14}	6.55E-01	6.86E+00	1.21E+01	2.69E+01	2.77E+00	8.78E-01	8.26E+00	1.30E+01	2.76E+01	2.88E+00	2.58E+01	2.76E+01	2.77E+01	2.92E+01	2.67E-01	6.83E+00	7.33E+00	7.27E+00	7.66E+00	6.25E-02
f_{15}	1.93E+02	6.52E+02	7.57E+02	1.73E+03	1.11E+02	5.52E+02	2.28E+03	2.30E+03	4.28E+03	2.68E+02	2.40E+02	8.40E+02	9.31E+02	1.82E+03	1.14E+02	4.00E+01	1.89E+02	3.47E+02	1.65E+03	1.06E+02

Table 3: Results from the Wilcoxon signed rank test

Algorithm	Criteria	Better	Equal	Worse	p-value	Decision
CTR	Best	13	0	2	5.00E-03	$+$
	Median	13	0	2	3.00E-03	$+$
	Mean	14	0	1	1.00E-03	$+$
TMMO	Best	13	0	2	1.00E-03	$+$
	Median	14	0	1	3.00E-03	$+$
	Mean	14	0	1	2.00E-03	$+$
GCM-PSO	Best	10	0	5	7.33E-01	$\approx$
	Median	10	0	5	4.96E-01	$\approx$
	Mean	7	0	8	1.91E-01	$\approx$

Table 4: Results obtained from Friedman rank test

Algorithm	Criteria	Rank	Order
Proposed	Mean	1.67	1
CTR		2.73	3
TMMO		3.73	4
GCM-PSO		1.87	2
Proposed	Median	1.57	1
CTR		2.50	3
TMMO		3.73	4
GCM-PSO		2.20	2

the test functions previously indicated. It can be seen that the proposed algorithm is evaluated as superior, inferior, or similar to others for 108, 27 and 0 instances, respectively, out of a total of 135 instances. Therefore, it can be stated that the proposed algorithm performs better than the others for 80.0% of the instances, and it is worse for only 20.0%. Furthermore, the Friedman rank test presented in Table 4, based on the mean and median results obtained, shows that the proposed algorithm is ranked first (having the smallest value), highlighted in bold, followed in order by GCM-PSO, CTR, and TMMO.

4.4. Robustness to Dynamic Changes

This section presents an evaluation of the performance of the proposed algorithm, using a test suite with different dynamic changes, including a varied number of peaks for each component, stronger shift severity and faster change frequency.

Table 5 shows the results obtained by both the proposed and the CTR algorithms, in which a distinct number of peaks were randomly generated for each component within the ranges of $\{1,...,5\}$ and $\{1,...,10\}$. Increasing the number of peaks allocated to each component usually increases the complexity of the landscape. Therefore, the results reveal that the performance of both algorithms worsen as the number of assigned peaks increases. In spite of this, the proposed algorithm maintains its superior performance over CTR for the two different peaks scenarios. According to the Wilcoxon test, the proposed algorithm is significantly better than CTR for both cases, as presented in Table 6. Out of 90 cases, our proposed approach is superior, inferior and similar to CTR for 79, 11 and 0 instances, respectively.

Also, it is clear that obtaining high-quality solutions within a high change frequency (i.e., a short computational time) is essential for real-world dynamic optimization problems. Table 7 displays the results of the proposed and the CTR algorithm with a lower number of fitness evaluations between any consecutive environments ($200D$). The results indicate that the performance of both algorithms deteriorates with a high change frequency, compared to those with slower changes. This is expected, since time constraints often make problems more difficult. However, the proposed algorithm still outperforms CTR on the majority of test instances. Also, the Wilcoxon test results shown in Table 8 clearly show that the proposed algorithm is significantly better than those of the CTR algorithm in the median and mean results.

Table 5: Results by the proposed algorithm and CTR for the GMPB problems with the number of peaks for each component.

# Peaks	Fun	CTR					Proposed algorithm				
		Best	Median	Mean	Worst	STD	Best	Median	Mean	Worst	STD
{1,...,5}	f_1	5.26E+00	8.18E+00	9.47E+00	1.97E+01	9.23E-01	4.40E+00	8.65E+00	9.07E+00	1.27E+01	6.02E-01
	f_2	1.16E+00	2.39E+00	2.42E+00	3.62E+00	1.66E-01	1.10E+00	2.17E+00	2.20E+00	3.32E+00	1.65E-01
	f_3	3.28E+00	9.03E+00	8.65E+00	1.38E+01	7.27E-01	4.62E+00	8.41E+00	8.88E+00	1.30E+01	6.15E-01
	f_4	2.78E-01	4.47E-01	4.48E-01	6.21E-01	2.26E-02	1.42E-01	2.66E-01	2.70E-01	3.92E-01	1.57E-02
	f_5	4.25E+01	3.01E+02	3.35E+02	8.72E+02	5.27E+01	4.22E+01	1.57E+02	1.67E+02	4.48E+02	2.65E+01
	f_6	7.55E+00	1.19E+01	1.37E+01	2.43E+01	1.20E+00	7.35E+00	1.44E+01	1.51E+01	2.65E+01	1.38E+00
	f_7	2.22E+00	4.73E+00	4.98E+00	7.84E+00	4.51E-01	1.85E+00	4.42E+00	4.60E+00	7.36E+00	4.48E-01
	f_8	9.58E+00	1.55E+01	1.70E+01	3.03E+01	1.51E+00	1.03E+01	1.46E+01	1.62E+01	2.76E+01	1.43E+00
	f_9	2.64E-01	4.28E-01	5.07E-01	1.73E+00	8.87E-02	2.32E-01	2.80E-01	3.67E-01	1.60E+00	8.83E-02
	f_{10}	1.67E+02	1.03E+03	9.99E+02	1.89E+03	1.45E+02	7.51E+01	3.93E+02	4.66E+02	1.15E+03	8.19E+01
	f_{11}	2.67E+01	4.79E+01	6.03E+01	2.21E+02	1.23E+01	2.48E+01	4.46E+01	4.59E+01	8.95E+01	4.63E+00
	f_{12}	1.67E+01	6.11E+01	1.04E+02	2.64E+02	2.31E+01	1.34E+01	5.53E+01	7.71E+01	2.10E+02	1.56E+01
	f_{13}	5.07E+01	1.07E+02	1.09E+02	1.99E+02	2.31E+01	3.06E+01	8.18E+01	8.70E+01	1.64E+02	1.00E+01
	f_{14}	3.40E-01	4.31E-01	4.35E-01	5.63E-01	1.84E-02	2.48E-01	2.78E-01	2.84E-01	3.46E-01	6.85E-03
	f_{15}	7.03E+02	2.46E+03	2.73E+03	5.97E+03	4.13E+02	2.65E+02	8.10E+02	1.26E+03	3.55E+03	2.91E+02
{1,...,10}	f_1	4.55E+00	9.68E+00	9.95E+00	1.67E+01	7.97E-01	5.02E+00	9.09E+00	8.80E+00	1.41E+01	5.91E-01
	f_2	1.27E+00	2.44E+00	2.50E+00	4.55E+00	2.07E-01	1.11E+00	1.83E+00	2.09E+00	3.74E+00	2.04E-01
	f_3	3.40E+00	8.37E+00	8.49E+00	1.29E+01	7.33E-01	2.59E+00	8.22E+00	8.51E+00	1.44E+01	8.21E-01
	f_4	2.64E-01	4.27E-01	4.19E-01	6.91E-01	3.06E-02	2.33E-01	3.26E-01	3.34E-01	4.31E-01	1.55E-02
	f_5	6.28E+01	3.39E+02	3.58E+02	8.74E+02	5.48E+01	3.89E+01	1.77E+02	2.14E+02	5.72E+02	3.86E+01
	f_6	7.97E+00	1.47E+01	1.61E+01	2.65E+01	1.29E+00	1.06E+01	1.47E+01	1.58E+01	2.48E+01	1.05E+00
	f_7	2.93E+00	5.18E+00	5.74E+00	9.94E+00	6.02E-01	1.28E+00	5.06E+00	5.46E+00	1.08E+01	8.01E-01
	f_8	1.20E+01	1.75E+01	2.17E+01	5.02E+01	2.60E+00	1.24E+01	1.61E+01	1.94E+01	3.59E+01	1.89E+00
	f_9	3.66E-01	4.79E-01	5.02E-01	8.50E-01	3.27E-02	2.30E-01	3.23E-01	3.26E-01	4.35E-01	1.66E-02
	f_{10}	2.25E+02	9.55E+02	9.27E+02	1.77E+03	1.23E+02	1.21E+02	4.06E+02	4.42E+02	1.11E+03	6.37E+01
	f_{11}	1.90E+01	3.76E+01	4.24E+01	6.52E+01	3.50E+00	1.79E+01	3.78E+01	3.52E+01	4.85E+01	2.36E+00
	f_{12}	3.52E+01	7.50E+01	8.81E+01	2.47E+02	1.37E+01	1.68E+01	5.32E+01	6.48E+01	1.99E+02	1.21E+01
	f_{13}	6.50E+01	9.76E+01	1.50E+02	4.39E+02	2.75E+01	3.50E+01	8.27E+01	9.64E+01	3.00E+02	1.74E+01
	f_{14}	3.75E-01	4.48E-01	4.50E-01	5.57E-01	1.36E-02	2.53E-01	2.93E-01	2.95E-01	3.37E-01	6.63E-03
	f_{15}	5.68E+02	2.78E+03	2.70E+03	5.68E+03	4.29E+02	1.77E+02	8.78E+02	1.19E+03	3.52E+03	2.77E+02

Table 6: Results from Wilcoxon signed rank test

# Peaks	Criteria	Better	Equal	Worse	p-value	Decision
{1,...,5}	Best	13	0	2	1.70E-02	+
	Median	13	0	2	1.10E-02	+
	Mean	13	0	2	9.00E-03	+
{1,...,10}	Best	12	0	3	2.70E-02	+
	Median	14	0	1	1.00E-03	+
	Mean	14	0	1	1.00E-03	+

Table 7: Results by the proposed algorithm and CTR for the GMPB problems with a change frequency = 200D.

Fun	CTR					Proposed algorithm				
	Best	Median	Mean	Worst	STD	Best	Median	Mean	Worst	STD
f_1	1.01E+01	1.98E+01	2.07E+01	3.69E+01	2.15E+00	1.13E+01	2.10E+01	2.10E+01	3.49E+01	1.70E+00
f_2	4.23E+00	6.83E+00	6.55E+00	1.01E+01	3.97E-01	4.57E+00	6.31E+00	6.19E+00	1.03E+01	3.66E-01
f_3	8.08E+00	1.58E+01	1.85E+01	3.11E+01	1.70E+00	8.70E+00	1.76E+01	1.77E+01	2.63E+01	1.54E+00
f_4	1.01E+00	3.14E+00	5.01E+00	1.94E+01	1.39E+00	1.14E+00	2.69E+00	4.90E+00	1.97E+01	1.38E+00
f_5	7.96E+01	2.35E+02	2.94E+02	7.33E+02	4.75E+01	3.67E+01	1.44E+02	1.78E+02	4.58E+02	3.40E+01
f_6	2.32E+01	3.29E+01	5.06E+01	2.86E+02	1.69E+01	2.30E+01	3.14E+01	4.23E+01	2.09E+02	1.20E+01
f_7	5.91E+00	1.04E+01	1.20E+01	4.45E+01	2.52E+00	4.87E+00	9.36E+00	1.00E+01	2.40E+01	1.27E+00
f_8	2.35E+01	3.85E+01	4.27E+01	7.55E+01	3.98E+00	2.05E+01	3.34E+01	3.75E+01	6.73E+01	3.61E+00
f_9	1.14E+00	1.36E+01	1.21E+01	2.55E+01	2.36E+00	1.12E+00	1.18E+01	1.14E+01	2.48E+01	2.28E+00
f_{10}	3.46E+02	1.24E+03	1.19E+03	2.21E+03	1.52E+02	8.86E+01	4.09E+02	4.70E+02	8.94E+02	6.63E+01
f_{11}	5.61E+01	1.13E+02	1.53E+02	7.58E+02	4.41E+01	6.15E+01	8.38E+01	1.31E+02	6.81E+02	4.00E+01
f_{12}	3.15E+01	1.45E+02	1.54E+02	3.66E+02	2.71E+01	3.53E+01	9.04E+01	1.22E+02	3.22E+02	2.39E+01
f_{13}	1.05E+02	2.03E+02	2.04E+02	3.10E+02	1.61E+01	6.08E+01	1.48E+02	1.51E+02	2.46E+02	1.33E+01
f_{14}	1.76E+00	9.95E+00	1.63E+01	3.37E+01	3.41E+00	2.28E+00	9.34E+00	1.54E+01	3.25E+01	3.18E+00
f_{15}	8.24E+02	2.93E+03	2.72E+03	4.79E+03	3.10E+02	2.81E+02	9.35E+02	1.01E+03	1.71E+03	1.15E+02

Table 8: Results from the Wilcoxon signed rank test

	Better	Equal	Worse	p-value	Decision
Best	8	0	7	4.27E-01	$\approx$
Median	13	0	2	8.00E-03	+
Mean	14	0	1	1.00E-03	+

Table 9: Results by the proposed algorithm and CTR for the GMPB problems with a shift severity in the range [3, 5].

Fun	CTR					Proposed algorithm				
	Best	Median	Mean	Worst	STD	Best	Median	Mean	Worst	STD
f_1	7.10E+00	1.23E+01	1.24E+01	1.77E+01	9.01E-01	6.73E+00	1.17E+01	1.33E+01	2.59E+01	1.44E+00
f_2	2.03E+00	3.91E+00	3.98E+00	5.52E+00	2.27E-01	1.91E+00	3.26E+00	3.44E+00	4.81E+00	1.92E-01
f_3	5.45E+00	1.24E+01	1.18E+01	1.71E+01	9.69E-01	6.93E+00	1.26E+01	1.24E+01	1.92E+01	1.04E+00
f_4	3.53E-01	1.74E+00	3.69E+00	1.73E+01	1.24E+00	4.19E-01	1.54E+00	3.15E+00	1.48E+01	1.06E+00
f_5	5.20E+01	1.78E+02	2.06E+02	5.70E+02	3.18E+01	3.49E+01	9.64E+01	1.34E+02	2.64E+02	2.05E+01
f_6	1.43E+01	2.11E+01	3.69E+01	2.02E+02	1.28E+01	1.53E+01	1.92E+01	3.25E+01	1.45E+02	8.84E+00
f_7	2.83E+00	5.11E+00	6.49E+00	2.08E+01	1.13E+00	2.95E+00	5.23E+00	6.98E+00	2.33E+01	1.29E+00
f_8	1.76E+01	2.87E+01	2.88E+01	4.51E+01	1.95E+00	1.45E+01	2.53E+01	2.74E+01	5.24E+01	2.58E+00
f_9	5.13E-01	1.04E+01	1.00E+01	2.36E+01	2.18E+00	4.57E-01	8.78E+00	8.86E+00	2.21E+01	1.96E+00
f_{10}	1.76E+02	7.22E+02	8.23E+02	2.16E+03	1.32E+02	8.73E+01	3.80E+02	4.91E+02	1.92E+03	1.15E+02
f_{11}	3.31E+01	5.39E+01	9.74E+01	6.38E+02	3.89E+01	2.78E+01	5.24E+01	8.14E+01	4.74E+02	2.84E+01
f_{12}	2.38E+01	5.60E+01	9.62E+01	3.00E+02	2.19E+01	1.32E+01	4.67E+01	7.10E+01	2.21E+02	1.61E+01
f_{13}	6.11E+01	1.55E+02	1.48E+02	2.32E+02	1.11E+01	4.71E+01	8.60E+01	1.10E+02	1.92E+02	1.25E+01
f_{14}	1.05E+00	8.78E+00	1.40E+01	2.96E+01	3.10E+00	7.98E-01	7.45E+00	1.25E+01	2.71E+01	2.80E+00
f_{15}	7.68E+02	2.15E+03	2.16E+03	3.73E+03	2.47E+02	1.83E+02	7.20E+02	9.92E+02	2.68E+03	1.90E+02

Furthermore, the robustness of the proposed algorithm is tested with stronger shift severities that make the tracking process more difficult. Table 9 shows the obtained results with stronger shift severities randomized in the range [3, 5], which shows that the results of the proposed algorithm are better than those of CTR. Also, Table 10 confirms the superiority of the proposed algorithm over CTR on stronger shift severities, as it is superior, similar and inferior to CTR in 36, 0 and 9 cases, respectively. The reason for the effectiveness of the proposed algorithm for solving dynamic problems with different dynamic changes is due to the expectation of some good solutions that help handling new environmental changes.

4.5. Variability in Dynamic Changes

To introduce a predictable movement for the components, Equation (4) is replaced by

$$c_i(t) = c_i(t-1) + S_i(t) \tag{5}$$

where $S_i(t)$ represents the peaks' center movement which is calculated as follows:

$$S_i(t) = \frac{S}{\|(1-\lambda)\varepsilon + \lambda S_i(t-1)\|}\left((1-\lambda)\varepsilon + \lambda S_i(t-1)\right) \tag{6}$$

Here, $\lambda \in [0, 1]$ is the correlation coefficient factor. A $\lambda = 0$ makes a completely random direction of the peak's movement whereas if $\lambda = 1$, the movement's direction is fully correlated with the direction of the previous move (i.e., linear shift). The parameter $\lambda = 0.5$ indicates that the peak movements exhibit a moderate level of correlation [69]. The results obtained from both the proposed and CTR algorithms are presented in Table 11. The proposed algorithm demonstrates its superior performance consistently compared to the CTR algorithm across most functions. Moreover, the statistical analysis conducted using the Wilcoxon test, as depicted in Table 12, shows that the proposed algorithm significantly outperforms the CTR algorithm in terms of median and mean results.

5. Conclusions and Future Work

In this paper, an algorithm for solving large-scale dynamic problems is proposed. The algorithm consists of three components: (1) problem decomposition to mitigate the curse of dimensionality: this was done by incorporating a variable interaction grouping and CC methods; (2) performing

Table 10: Results from the Wilcoxon signed rank test

	Better	Equal	Worse	p-value	Decision
Best	11	0	4	2.50E-02	$+$
Median	13	0	2	1.00E-03	$+$
Mean	12	0	3	5.00E-03	$+$

Table 11: Results by the proposed algorithm and CTR for the GMPB problems with variability in dynamic changes.

Fun	CTR					Proposed algorithm				
	Best	Median	Mean	Worst	STD	Best	Median	Mean	Worst	STD
f_1	3.09E+00	8.20E+00	7.66E+00	1.26E+01	6.78E-01	4.23E+00	8.01E+00	9.94E+00	2.02E+01	1.23E+00
f_2	1.26E+00	2.38E+00	2.58E+00	4.98E+00	2.36E-01	9.94E-01	1.88E+00	2.03E+00	4.09E+00	2.32E-01
f_3	3.80E+00	7.75E+00	7.79E+00	1.40E+01	7.85E-01	2.65E+00	6.34E+00	6.84E+00	1.33E+01	8.12E-01
f_4	7.35E-01	4.87E+00	5.96E+00	1.41E+01	1.25E+00	2.26E-01	5.07E+00	5.65E+00	1.37E+01	1.31E+00
f_5	6.34E+01	3.14E+02	2.83E+02	6.78E+02	4.70E+01	1.77E+01	1.48E+02	1.57E+02	3.32E+02	2.57E+01
f_6	6.24E+00	1.13E+01	2.45E+01	2.03E+02	1.28E+01	6.60E+00	1.06E+01	1.72E+01	9.39E+01	5.55E+00
f_7	2.41E+00	4.22E+00	5.66E+00	1.88E+01	1.16E+00	2.26E+00	3.76E+00	6.08E+00	1.79E+01	1.25E+00
f_8	7.82E+00	1.60E+01	1.91E+01	4.04E+01	2.23E+00	5.74E+00	1.64E+01	1.87E+01	3.54E+01	2.36E+00
f_9	7.15E-01	6.33E+00	7.80E+00	1.87E+01	1.72E+00	2.89E-01	6.55E+00	7.62E+00	1.89E+01	1.79E+00
f_{10}	3.01E+02	7.39E+02	8.17E+02	1.51E+03	9.68E+01	1.04E+02	2.91E+02	3.60E+02	8.90E+02	5.75E+01
f_{11}	1.72E+01	3.87E+01	5.95E+01	3.46E+02	2.07E+01	1.76E+01	3.42E+01	5.34E+01	2.37E+02	1.37E+01
f_{12}	1.80E+01	4.53E+01	6.27E+01	1.54E+02	1.04E+01	2.16E+01	4.39E+01	5.20E+01	1.28E+02	7.90E+00
f_{13}	6.67E+01	9.90E+01	1.14E+02	1.97E+02	9.61E+00	4.10E+01	6.91E+01	8.45E+01	1.87E+02	9.64E+00
f_{14}	1.41E+00	8.84E+00	1.08E+01	2.31E+01	1.83E+00	8.02E-01	8.93E+00	1.10E+01	2.40E+01	1.93E+00
f_{15}	6.38E+02	2.07E+03	2.21E+03	4.27E+03	2.92E+02	1.81E+02	7.48E+02	8.69E+02	2.60E+03	1.53E+02

Table 12: Results from the Wilcoxon signed rank test

	Better	Equal	Worse	p-value	Decision
Best	11	0	4	6.10E-02	$\approx$
Median	11	0	4	8.00E-03	$+$
Mean	12	0	3	9.00E-03	$+$

an evolutionary process through a multi-population algorithm that adopted PSO and its operators, where an explorer and exploiter subpopulations are assigned to each subproblem to be evolved in a contribution-based manner; and (3) a prediction mechanism based on a neural network to predict new solutions when a dynamic change happens. The proposed algorithm was tested on the generalized moving peaks benchmark problems with 50, 100 and 200 decision variables. The results presented indicate that the proposed algorithm is capable of locating the optimum in a new environment more quickly. Interestingly, the experimental analysis revealed that the proposed algorithm outperforms state-of-the-art algorithms in 80.0% of all instances. Also, the proposed algorithm achieves superior performance on difficult scenarios, including stronger shift severity and faster change frequency. As part of our future work, we are interested in designing a dynamic optimization method to handle constrained DOPs, as many real-world DOPs are constrained.

Acknowledgment

This research is supported by the ARC Discovery Project (No. DP210102939). Carlos A. Coello Coello gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (Investigación en Fronteras de la Ciencia 2016).

References

[1] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (2012) 1–24. doi:https://doi.org/10.1016/j.swevo.2012.05.001.

[2] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, X. Yao, A survey of evolutionary continuous dynamic optimization over two decades—part b, IEEE Transactions on Evolutionary Computation 25 (2021) 630–650. doi:10.1109/TEVC.2021.3060012.

[3] J. K. Kordestani, A. E. Ranginkaman, M. R. Meybodi, P. Novoa-Hernández, A novel framework for improving multi-population algorithms for dynamic optimization problems: A scheduling approach, Swarm and Evolutionary Computation 44 (2019) 788–805. doi:https://doi.org/10.1016/j.swevo.2018.09.002.

[4] C. Bu, W. Luo, L. Yue, Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies, IEEE Transactions on Evolutionary Computation 21 (2017) 14–33. doi:10.1109/TEVC.2016.2567644.

[5] N. Hamza, R. Sarker, D. Essam, Sensitivity-based change detection for dynamic constrained optimization, IEEE Access 8 (2020) 103900–103912. doi:10.1109/ACCESS.2020.2999161.

[6] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, X. Yao, A survey of evolutionary continuous dynamic optimization over two decades—part a, IEEE Transactions on Evolutionary Computation 25 (2021) 609–629. doi:10.1109/TEVC.2021.3060014.

[7] J. Branke, Evolutionary optimization in dynamic environments, volume 3, Springer Science & Business Media, 2012.

[8] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Transactions on Evolutionary Computation 10 (2006) 440–458. doi:10.1109/TEVC.2005.859468.

[9] K. Deb, U. B. Rao N., S. Karthik, Dynamic multi-objective optimization and decision-making using modified nsga-ii: A case study on hydro-thermal power scheduling, in: S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, T. Murata (Eds.), Evolutionary Multi-Criterion Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 803–817.

[10] S. Das, A. Mandal, R. Mukherjee, An adaptive differential evolution algorithm for global optimization in dynamic environments, IEEE Transactions on Cybernetics 44 (2014) 966–978. doi:10.1109/TCYB.2013.2278188.

[11] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Transactions on Evolutionary Computation 10 (2006) 459–472. doi:10.1109/TEVC.2005.857074.

[12] W. Luo, J. Sun, C. Bu, H. Liang, Species-based particle swarm optimizer enhanced by memory for dynamic optimization, Applied Soft Computing 47 (2016) 130–140. doi:https://doi.org/10.1016/j.asoc.2016.05.032.

[13] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), volume 3, 1999, pp. 1875–1882 Vol. 3. doi:10.1109/CEC.1999.785502.

[14] M. Jiang, Z. Huang, L. Qiu, W. Huang, G. G. Yen, Transfer learning-based dynamic multiobjective optimization algorithms, IEEE Transactions on Evolutionary Computation 22 (2018) 501–514. doi:10.1109/TEVC.2017.2771451.

[15] A. Meier, O. Kramer, Prediction with recurrent neural networks in evolutionary dynamic optimization, in: K. Sim, P. Kaufmann (Eds.), Applications of Evolutionary Computation, Springer International Publishing, Cham, 2018, pp. 848–863.

[16] A. Ahrari, S. Elsayed, R. Sarker, D. Essam, C. A. Coello Coello, Adaptive multilevel prediction method for dynamic multimodal optimization, IEEE Transactions on Evolutionary Computation 25 (2021) 463–477. doi:10.1109/TEVC.2021.3051172.

[17] J. Branke, T. Kaussler, C. Smidt, H. Schmeck, A multi-population approach to dynamic optimization problems, in: I. C. Parmee (Ed.), Evolutionary Design and Manufacture, Springer London, London, 2000, pp. 299–307.

[18] M. Mavrovouniotis, C. Li, S. Yang, A survey of swarm intelligence for dynamic optimization: Algorithms and applications, Swarm and Evolutionary Computation 33 (2017) 1–17. doi:https://doi.org/10.1016/j.swevo.2016.12.005.

[19] X. Chen, W. Du, R. Qi, F. Qian, H. Tianfield, Hybrid gradient particle swarm optimization for dynamic optimization problems of chemical processes, Asia-Pacific Journal of Chemical Engineering 8 (2013) 708–720. doi:https://doi.org/10.1002/apj.1712. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/apj.1712.

[20] N. Dige, U. Diwekar, Efficient sampling algorithm for large-scale optimization under uncertainty problems, Computers Chemical Engineering 115 (2018) 431–454. doi:https://doi.org/10.1016/j.compchemeng.2018.05.007.

[21] X. Mingming, Z. Jun, C. Kaiquan, C. Xianbin, T. Ke, Cooperative co-evolution with weighted random grouping for large-scale crossing waypoints locating in air route network, in: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, 2011, pp. 215–222. doi:10.1109/ICTAI.2011.40.

[22] W. Dong, T. Chen, P. Tino, X. Yao, Scaling up estimation of distribution algorithms for continuous optimization, IEEE Transactions on Evolutionary Computation 17 (2013) 797–822.

[23] M. Bhattacharya, R. Islam, J. Abawajy, Evolutionary optimization: a big data perspective, Journal of Network and Computer Applications 59 (2016) 416–426.

[24] M. A. Potter, K. A. De Jong, A cooperative coevolutionary approach to function optimization, in: Y. Davidor, H.-P. Schwefel, R. Männer (Eds.), Parallel Problem Solving from Nature — PPSN III, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 249–257.

[25] M. Omidvar, X. Li, X. Yao, A review of population-based metaheuristics for large-scale black-box global optimization: Part b, IEEE Transactions on Evolutionary Computation (2021).

[26] M. N. Omidvar, X. Li, X. Yao, A review of population-based metaheuristics for large-scale black-box global optimization: Part a, IEEE Transactions on Evolutionary Computation (2021) 1–1. doi:10.1109/TEVC.2021.3130838.

[27] M. Meselhi, R. Sarker, D. Essam, S. Elsayed, A decomposition approach for large-scale non-separable optimization problems, Applied Soft Computing 115 (2022) 108168. doi:https://doi.org/10.1016/j.asoc.2021.108168.

[28] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, X. Yao, Scaling up dynamic optimization problems: A divide-and-conquer approach, IEEE Transactions on Evolutionary Computation 24 (2020) 1–15. doi:10.1109/TEVC.2019.2902626.

[29] X.-F. Liu, Z.-H. Zhan, T.-L. Gu, S. Kwong, Z. Lu, H. B.-L. Duh, J. Zhang, Neural network-based information transfer for dynamic opti-

mization, IEEE Transactions on Neural Networks and Learning Systems 31 (2020) 1557–1570. doi:10.1109/TNNLS.2019.2920887.

[30] C. Rossi, M. Abderrahim, J. C. Díaz, Tracking moving optima using kalman-based predictions, Evolutionary Computation 16 (2008) 1–30. doi:10.1162/evco.2008.16.1.1.

[31] A. Simões, E. Costa, Prediction in evolutionary algorithms for dynamic environments, Soft Computing 18 (2014) 1471–1497.

[32] C.-K. Goh, K. C. Tan, A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization, IEEE Transactions on Evolutionary Computation 13 (2009) 103–127. doi:10.1109/TEVC.2008.920671.

[33] H. G. Cobb, An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments, Technical Report, Naval Research Lab Washington DC, 1990.

[34] Y. G. Woldesenbet, G. G. Yen, Dynamic evolutionary algorithm with variable relocation, IEEE Transactions on Evolutionary Computation 13 (2009) 500–513. doi:10.1109/TEVC.2008.2009031.

[35] J. J. Grefenstette, et al., Genetic algorithms for changing environments, in: Ppsn, volume 2, Citeseer, 1992, pp. 137–144.

[36] C. Li, S. Yang, A general framework of multipopulation methods with clustering in undetectable dynamic environments, IEEE Transactions on Evolutionary Computation 16 (2012) 556–577. doi:10.1109/TEVC.2011.2169966.

[37] H. G. Cobb, J. J. Grefenstette, Genetic algorithms for tracking changing environments., Technical Report, Naval Research Lab Washington DC, 1993.

[38] X. Xu, Y. Tan, W. Zheng, S. Li, Memory-enhanced dynamic multi-objective evolutionary algorithm based on lp decomposition, Applied Sciences 8 (2018) 1673.

[39] X.-F. Liu, X.-X. Xu, Z.-H. Zhan, Y. Fang, J. Zhang, Interaction-based prediction for dynamic multiobjective optimization, IEEE Transactions on Evolutionary Computation 27 (2023) 1881–1895. doi:10.1109/TEVC.2023.3234113.

[40] T. T. Nguyen, X. Yao, Evolutionary optimization on continuous dynamic constrained problems - an analysis, in: S. Yang, X. Yao (Eds.), Evolutionary Computation for Dynamic Optimization Problems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 193–217.

[41] R. I. Lung, D. Dumitrescu, Evolutionary swarm cooperative optimization in dynamic environments, Natural Computing 9 (2010) 83–94.

[42] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Transactions on Evolutionary Computation 16 (2012) 210–224. doi:10.1109/TEVC.2011.2112662.

[43] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling up fast evolutionary programming with cooperative coevolution, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546), volume 2, 2001, pp. 1101–1108 vol. 2. doi:10.1109/CEC.2001.934314.

[44] Z. Yang, K. Tang, X. Yao, Differential evolution for high-dimensional function optimization, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 3523–3530. doi:10.1109/CEC.2007.4424929.

[45] M. Yang, A. Zhou, C. Li, X. Yao, An efficient recursive differential grouping for large-scale continuous problems, IEEE Transactions on Evolutionary Computation 25 (2021) 159–171. doi:10.1109/TEVC.2020.3009390.

[46] Y. Mei, M. N. Omidvar, X. Li, X. Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. 42 (2016). doi:10.1145/2791291.

[47] Y. Sun, M. Kirley, S. K. Halgamuge, A recursive decomposition method for large scale continuous optimization, IEEE Transactions on Evolutionary Computation 22 (2018) 647–661. doi:10.1109/TEVC.2017.2778089.

[48] M. N. Omidvar, X. Li, K. Tang, Designing benchmark problems for large-scale continuous optimization, Information Sciences 316 (2015) 419–436. doi:https://doi.org/10.1016/j.ins.2014.12.062, nature-Inspired Algorithms for Large Scale Global Optimization.

[49] G. A. Trunfio, P. Topa, J. Wąs, A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution, Information Sciences 372 (2016) 773–795. doi:https://doi.org/10.1016/j.ins.2016.08.080.

[50] W. Luo, B. Yang, C. Bu, X. Lin, A hybrid particle swarm optimization for high-dimensional dynamic optimization, in: Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, Y. Jin (Eds.), Simulated Evolution and Learning, Springer International Publishing, Cham, 2017, pp. 981–993.

[51] M. N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Transactions on Evolutionary Computation 18 (2014) 378–393. doi:10.1109/TEVC.2013.2281543.

[52] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Transactions on Evolutionary Computation 10 (2006) 440–458. doi:10.1109/TEVC.2005.859468.

[53] Z.-H. Zhan, J.-Y. Li, S. Kwong, J. Zhang, Learning-aided evolution for optimization, IEEE Transactions on Evolutionary Computation 27 (2023) 1794–1808. doi:10.1109/TEVC.2022.3232776.

[54] Y. Jiang, Z.-H. Zhan, K. C. Tan, J. Zhang, Knowledge learning for evolutionary computation, IEEE Transactions on Evolutionary Computation (2023) 1–1. doi:10.1109/TEVC.2023.3278132.

[55] R. Rojas, Neural networks: a systematic introduction, Springer Science & Business Media, 2013.

[56] M. N. Omidvar, M. Yang, Y. Mei, X. Li, X. Yao, Dg2: A faster and more accurate differential grouping for large-scale black-box optimization, IEEE Transactions on Evolutionary Computation 21 (2017) 929–942. doi:10.1109/TEVC.2017.2694221.

[57] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. R. Meybodi, A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization, Applied Soft Computing 13 (2013) 2144–2158. doi:https://doi.org/10.1016/j.asoc.2012.12.020.

[58] T. Blackwell, Particle swarm optimization in dynamic environments, in: Evolutionary computation in dynamic and uncertain environments, Springer, 2007, pp. 29–49.

[59] K. Trojanowski, Properties of quantum particles in multi-swarms for dynamic optimization, Fundamenta Informaticae 95 (2009) 349–380.

[60] A. Sepas-Moghaddam, A. Arabshahi, D. Yazdani, M. M. Dehshibi, A novel hybrid algorithm for optimization in multimodal dynamic environments, in: 2012 12th International Conference on Hybrid Intelligent Systems (HIS), IEEE, 2012, pp. 143–148.

[61] M. N. Omidvar, B. Kazimipour, X. Li, X. Yao, Cbcc3 — a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 3541–3548. doi:10.1109/CEC.2016.7744238.

[62] S. Mahdavi, S. Rahnamayan, M. E. Shiri, Multilevel framework for large-scale global optimization, Soft Computing 21 (2017) 4111–4140.

[63] M. Hagan, M. Menhaj, Training feedforward networks with the marquardt algorithm, IEEE Transactions on Neural Networks 5 (1994) 989–993. doi:10.1109/72.329697.

[64] J. S. Smith, B. Wu, B. M. Wilamowski, Neural network training with levenberg–marquardt and adaptable weight compression, IEEE Transactions on Neural Networks and Learning Systems 30 (2019) 580–587. doi:10.1109/TNNLS.2018.2846775.

[65] R. Morrison, K. De Jong, A test problem generator for non-stationary environments, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), volume 3, 1999, pp. 2047–2053 Vol. 3. doi:10.1109/CEC.1999.785526.

[66] B. Nasiri, M. Meybodi, M. Ebadzadeh, History-driven particle swarm optimization in dynamic and uncertain environments, Neurocomputing 172 (2016) 356–370. doi:https://doi.org/10.1016/j.neucom.2015.05.115.

[67] C. Li, S. Yang, A generalized approach to construct benchmark problems for dynamic optimization, in: X. Li, M. Kirley, M. Zhang, D. Green, V. Ciesielski, H. Abbass, Z. Michalewicz, T. Hendtlass, K. Deb, K. C. Tan, J. Branke, Y. Shi (Eds.), Simulated Evolution and Learning, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 391–400.

[68] G. Pamparà, A. P. Engelbrecht, A generator for dynamically constrained optimization problems, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1441–1448. doi:10.1145/3319619.3326798.

[69] D. Yazdani, M. N. Omidvar, R. Cheng, J. Branke, T. T. Nguyen, X. Yao, Benchmarking continuous dynamic optimization: Survey and generalized test suite, IEEE Transactions on Cybernetics (2020) 1–14. doi:10.1109/TCYB.2020.3011828.

[70] M. N. Omidvar, D. Yazdani, J. Branke, X. Li, S. Yang, X. Yao, Generating large-scale dynamic optimization problem instances using the generalized moving peaks benchmark, 2021. `arXiv:2107.11019`.

[71] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, volume 4, 1995, pp. 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968.

[72] E. Theodorsson-Norheim, Friedman and quade tests: Basic computer program to perform nonparametric two-way analysis of variance and multiple comparisons on ranks of several related samples, Computers in biology and medicine 17 (1987) 85–99.