Scaling Up Dynamic Optimization Problems: A Divide-and-Conquer Approach

Danial Yazdani, Mohammad Nabi Omidvar, Jürgen Branke, Trung Thanh Nguyen, and Xin Yao

***Abstract*—Scalability is a crucial aspect of designing efficient algorithms. Despite their prevalence, large-scale dynamic optimization problems are not well-studied in the literature. This paper is concerned with designing benchmarks and frameworks for the study of large-scale dynamic optimization problems. We start by a formal analysis of the moving peaks benchmark and show its nonseparable nature irrespective of its number of peaks. We then propose a composite moving peaks benchmark suite with exploitable modularity covering a wide range of scalable partially separable functions suitable for the study of large-scale dynamic optimization problems. The benchmark exhibits modularity, heterogeneity, and imbalance features to resemble real-world problems. To deal with the intricacies of large-scale dynamic optimization problems, we propose a decomposition-based coevolutionary framework which breaks a large-scale dynamic optimization problem into a set of lower dimensional components. A novel aspect of the framework is its efficient bi-level resource allocation mechanism which controls the budget assignment to components and the populations responsible for tracking multiple moving optima. Based on a comprehensive empirical study on a wide range of large-scale dynamic optimization problems with up to 200 dimensions, we show the crucial role of problem decomposition and resource allocation in dealing with these problems. The experimental results clearly show the superiority of the proposed framework over three other approaches in solving large-scale dynamic optimization problems.**

***Index Terms*—dynamic optimization problems, large-scale optimization problems, decomposition, multi-population, computational resource allocation, cooperative coevolutionary.**

I. Introduction

Change is an inescapable aspect of natural and artificial systems, and adaptation is central to their resilience [1], [2]. Optimization problems are no exception to this maxim. Indeed, viability of businesses and their operational success depends heavily on their effectiveness in responding to a change in the myriad of optimization problems they entail. For an optimization problem, this boils down to the efficiency of an algorithm to find and maintain a quality solution to an ever changing problem.

Ubiquity of dynamic optimization problems (DOPs) [3] demands extensive research into design and development of algorithms capable of dealing with various types of change [4]. These are often attributed to a change in the objective function, its number of decision variables, or constraints. Despite the large body of literature on dynamic optimization problems and algorithms, little attention has been given to their scalability. Indeed, the number of dimensions of a typical DOP studied in the literature rarely exceeds twenty. This is contrary to the emergence of high-dimensional dynamic optimization problems such as deep online learning [5]. Deep learning problems are large-scale by nature and the arrival of new training data makes online learning a dynamic problem. Online clustering of high-dimensional data is another example of a large-scale dynamic optimization problem [6], [7]. Many large-scale static optimization problems can also be regarded as dynamic due to unforeseen environmental changes. Large-scale crossing waypoints locating in air route networks is such a problem whose problem space changes by delayed airplanes, breakdowns, and extreme weather conditions [8].

Motivated by rapid technological advancements, large-scale optimization has gained popularity in recent years [9]. However, the exponential growth in the size of the search space, with respect to an increase in the number of decision variables, has made large-scale optimization a challenging task. For DOPs, however, the challenge is twofold. For such problems, not only should an algorithm be capable of finding the global optimum in the vastness of the search space but it should also be able to track it over time. For multi-modal DOPs, where several optima have the potential to turn into the global optimum after environmental changes, the cost of tracking multiple moving optima also adds to the complexity.

Exploiting problem structure is an effective way of approaching complex problems. Knowledge about the regularities and the structure of a problem allows us to devise more effective ways of solving them. This is common practice in many branches of optimization [10]–[13]. More recently, the term gray-box optimization has come to refer to the practice of incorporating problem structure into the optimization process [14]. In evolutionary computation, finding and exploiting the "hidden order" [15] by means of linkage learning has played a central role in designing scalable optimization

This work was supported in part by a Deans Scholarship by Faculty of Engineering and Technology, LJMU, a Newton Institutional Links grant no. 172734213, funded by the UK BEIS and delivered by the British Council, a NRCP grant no. NRCP1617-6-125 delivered by Royal Academy of Engineering, two EPSRC grants (nos. EP/P005578/1 and EP/J017515/1). Xin Yao was supported by a Royal Society Wolfson Research Merit Award and Honda Research Institute Europe.

D. Yazdani and T. T. Nguyen are with the Liverpool Logistics, Offshore and Marine Research Institute, Department of Maritime and Mechanical Engineering, Liverpool John Moores University, Liverpool L3 3AF, United Kingdom (e-mails: danial.yazdani@gmail.com, T.T.Nguyen@ljmu.ac.uk).

M. N. Omidvar and X. Yao are with the Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: m.omidvar@cs.bham.ac.uk, x.yao@cs.bham.ac.uk). Xin Yao is also with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China.

J. Branke is with the Operational Research and Management Sciences Group in Warwick Business school, University of Warwick, Coventry CV4 7AL, United Kingdom (email: Juergen.Branke@wbs.ac.uk).

The first two authors contributed equally to this work.

algorithms [16], [17]. Divide-and-conquer and problem decomposition techniques have also gained popularity in large-scale global optimization in recent years [18], [19]. However, scalability of dynamic optimization algorithms and the notion of exploiting problem structure are under-explored areas which we set out to study in this paper.

Moving peaks benchmark (MPB) [20] is the most popular benchmark in the field of dynamic optimization. Despite being scalable, MPB's lack of modularity limits its utility for the study of large-scale DOPs. Real-world problems often exhibit a modular structure with nonuniform imbalance among the contribution of its constituent parts to the objective value [21]. The modularity is caused by the interaction structure of the decision variables resulting in a wide range of structures from fully separable functions to fully nonseparable ones. Most problems exhibit some degree of sparsity in their interaction structure, which can be exploited by optimization algorithms. The imbalance property can be caused as a by-product of modularity or due to the heterogeneous nature of the input variables and their domains. For example, model predictive control (MPC) is a dynamic optimization problem with a wide range of applications in chemical power plants, robotics, and power systems and exhibits modularity and imbalance [22]. In this paper, we formally analyze the standard MPB and show that it is additively nonseparable. We then propose a new benchmark generator by composing several MPBs to account for modularity and imbalance.

In addition to the new benchmark, we draw on advances in large-scale global optimization and propose a decomposition-based framework for large-scale dynamic optimization problems. The idea is to first discover and exploit the underlying structure of a given problem by decomposing it into several components of smaller sizes, and then to tackle the subproblems simultaneously. The former can be achieved by a wide range of variable interaction analysis algorithms capable of identifying the underlying structure of a black-box problem with high efficiency and accuracy [18], [19], [23], [24], and the latter can be achieved by means of cooperative coevolution (CC) [25]–[27]. To deal with the imbalance problem, we also devise a new resource allocation policy, which takes the dynamic nature of the problem into account for an economical use of the limited computational resources. We empirically evaluate the proposed framework on a wide range of problem settings to validate the efficacy of various strategies such as problem decomposition, tracking multiple moving optima, and resource allocation. In short, this paper has the following major contributions:

1) A mathematical variable interaction analysis on the MPB benchmark to determine its interaction structure.
2) A large-scale benchmark suite with a modular heterogeneous structure allowing for imbalance among its components.
3) A decomposition-based algorithm for solving large-scale DOPs with a novel resource allocation mechanism.

The organization of this paper is as follows. Section II covers the background information and related work. Section III contains the analysis of the moving peaks benchmark and the details of the proposed large-scale benchmark function generator for DOPs. The details of the proposed decomposition-based algorithm and its resource allocation mechanism is given in Section IV. Section V is concerned with a comprehensive empirical analysis of the proposed algorithm. Finally, Section VI concludes the paper and outlines possible future directions.

II. Background and Literature Review

DOPs are usually represented as follows:

$$F(\mathbf{x}) = f\left(\mathbf{x}, \theta^{(t)}\right), \tag{1}$$

where f is the objective function, $\mathbf{x}$ is a design vector, $\theta^{(t)}$ is environmental parameters which change over time and t is the time index with $t \in [0, T]$ where T is the problem life cycle. In this paper, like most previous studies in the DOP domain, we investigate DOPs that change discretely over time, i.e., $t \in \{1, \ldots, T\}$. In this type of DOP, the environmental parameters change over time with stationary periods between changes. As a result, for a DOP with T environmental states, we have a sequence of T static environments:

$$F(\mathbf{x}) = \left[f(\mathbf{x}, \theta^{(1)}), f(\mathbf{x}, \theta^{(2)}), \ldots, f(\mathbf{x}, \theta^{(T)})\right], \tag{2}$$

where $\theta^{(i)}$ denotes the parameters of the ith environment.

A. Variable Interaction

Variable interaction or linkage refers to the extent to which the optimum of a variable depends on the values taken by other decision variables. For continuous optimization problems, variable interaction is defined as follows [18]:

Definition 1 (Mei et al. [18])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function. Decision variables x_i and x_j interact if a candidate solution $\mathbf{x}^\star$ exists, such that*

$$\frac{\partial^2 f(\mathbf{x}^\star)}{\partial x_i \partial x_j} \neq 0.$$

Some functions exhibit an underlying interaction structure such that groups of decision variables can be optimized independently. These functions, which are called *partially separable*, are defined as follows:

Definition 2 (Omidvar et al. [21])**.** *A function $f(\mathbf{x})$ is partially separable with m independent components iff:*

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \left(\arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \ldots), \ldots, \arg\min_{\mathbf{x}_m} f(\ldots, \mathbf{x}_m)\right),$$

where $\mathbf{x} = (x_1, \ldots, x_n)^\top$ is a decision vector of n dimensions, $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are disjoint sub-vectors of $\mathbf{x}$, and $2 \leq m \leq n$. The function is called fully separable when $m = n$.

Additive separability is a special type of partial separability, which is defined as follows:

Definition 3 (Omidvar et al. [21])**.** *A function is additively separable if it has the following general form:*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_i),\ m > 1,$$

Algorithm 1: $(x^\star, f^\star) = \texttt{CC}(f)$

```
/*Main Framework of CC*/
P ← randomized initial population;
c ← randomized initial context vector;
//grouping stage
G = Grouping(f);
//optimization stage
while Termination Condition is Not Satisfied do
    for κ = 1 to |G| do
        (P, c) = Optimizer(P, c, G_κ);
x* = c ; f* = f(x*) ;
return (x*, f*);
```

where $f_i(\cdot)$ is a nonseparable subfunction, and m is the number of nonseparable components of f. The definition of $\mathbf{x}$ and $\mathbf{x}_i$ is identical to what was given in Def. 2.

Definition 4 (Omidvar et al. [21])**.** *A function $f(\mathbf{x})$ is fully nonseparable if every pair of its decision variables interact.*

Let us provide two illustrative examples. Given the polynomial $f(\mathbf{x}) = x_1^2 + 3x_1x_2^2 + 2x_3^2x_4^3$, by applying Def. 1 we can show that $\frac{\partial f(\mathbf{x})}{\partial x_1 \partial x_2} = 6x_2$ which is clearly nonzero when $x_2 \neq 0$. Therefore, x_1 and x_2 interact. Conversely, the quantity $\frac{\partial f(\mathbf{x})}{\partial x_1 \partial x_3}$ is identically zero regardless of the choice of $\mathbf{x}$. Therefore, x_1 and x_3 are separable. It is clear that the nonlinearity of $f(\mathbf{x})$ is caused by the product terms, resulting in the following interaction groups: $\{x_1, x_2\}$ and $\{x_3, x_4\}$. Accordingly, we can rewrite $f(\mathbf{x})$ in terms of two subfunctions as follows: $f(\mathbf{x}) = f_1(x_1, x_2) + f_2(x_3, x_4)$. It is therefore clear that $f(\mathbf{x})$ is both partially separable and partially additively separable (Defs. 2 and 3). Another example is $g(\mathbf{x}) = \exp\{\sum_{i=1}^{n} x_i^2\}$. It is clear that all second-order partial derivatives of $g(\mathbf{x})$ are nonzero except at the origin, which forces all pairs of variables to interact (Def. 4). However, the optimal values for each dimension can still be found independently regardless of the values taken by other dimensions. This makes $g(\mathbf{x})$ fully separable according to Def. 2 where $m = n$.

B. Cooperative Coevolution

Cooperative coevolution (CC) has been proposed by Potter and De Jong [25] with the goal of allowing evolutionary algorithms the capacity to solve increasingly complex problems. The idea is based on decomposing a complex problem into subproblems of lower complexity which are coadapted within an evolutionary context. Algorithm 1 shows a high-level representation of CC. In the original implementation of CC, an n-dimensional problem is decomposed into n 1-dimensional problems each of which is optimized using a given optimizer in a round-robin fashion. In order to assign a fitness to each partial solution in a component, the individuals are evaluated within the context of a complete solution often referred to as the *context vector* [28].

The round-robin optimization of components assumes a uniform contribution from each component which is often not the case for various reasons [21]. The so-called *imbalance* among the contribution of components can be attributed to the following: 1) nonuniform dimensionality of the underlying component functions. 2) component functions with different landscapes and output ranges. 3) the dynamics of the optimizer, its convergence behavior, and stagnation. Contribution-based cooperative coevolution (CBCC) [29], [30] is an improved CC framework which addresses the imbalance issue by assigning more resources to components with higher overall contributions. An important aspect of a contribution-aware coevolutionary framework is maintaining an optimal balance between an exploration phase in which the contribution of components is updated, and an exploitation phase in which the most contributing component is optimized. This has resulted in many attempts to design various exploration/exploitation polices [31]–[33].

The original CC framework and its contribution-based counterpart have no explicit means of dealing with variable interactions. They only respond to interactions through the *cooperation* of individuals in updating the context vector, which acts as a message passing mechanism. The efficiency of this approach depends on the policy of constructing the context vector [34] as well as its update frequency [35]. To alleviate this problem, many variable interaction analysis algorithms have been proposed with the aim of decomposing a large-scale problem into smaller *independent* components. There have been many attempts on this [26], [36], among which the differential grouping family of algorithms showed the highest accuracy [18], [19], [23]. Differential grouping (DG) works on the basis of the following theorem:

Theorem 1 (Omidvar et al. [23])**.** *Let $f(\mathbf{x})$ be an additively separable function. $\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$, variables x_p and x_q interact if the following condition holds*

$$\Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_2}, \qquad (3)$$

where

$$\Delta_{\delta,x_p}[f](\mathbf{x}) = f(\dots, x_p + \delta, \dots) - f(\dots, x_p, \dots), \qquad (4)$$

refers to the forward difference of f with respect to variable x_p with interval δ.

The quantities in (3) are real-valued numbers; therefore, the equality check cannot be evaluated exactly over the floating-point number field on computer systems. Consequently, the equality check needs to be converted to an inequality check by introducing a sensitivity parameter: $|\Delta^{(1)} - \Delta^{(2)}| > \epsilon$. Here, $\Delta^{(1)}$ and $\Delta^{(2)}$ denote the left and right hand side of (3), respectively. In the absence of representation and roundoff errors, ϵ can be theoretically set to zero; however, this is not usually the case and the optimal value of ϵ is often a nonzero positive number. This parameterization makes DG sensitive to choices of ϵ whose optimal value may vary from function to function and is difficult to tune by practitioners. To alleviate this problem, Omidvar et al. proposed DG2 [24], a parameter-free version of DG, which automatically sets ϵ by estimating the bounds on the computational roundoff errors to maximize the accuracy of variable interaction detection. DG2 is the core decomposition algorithm used in this paper.

C. Tracking Moving Optimum

Tracking moving optimum is the most popular approach in the DOP domain in which algorithms try to find the optimum

and track it after each environmental change. One of the most important and challenging DOPs are problems with several competing local optima each having the potential to become the global optimum after an environmental change [4]. A multi-population strategy is one of the most effective approaches for solving this type of DOPs [37]. In this section, we only focus on the most relevant algorithms in which the multi-population strategy is ustilized for tracking multiple moving optima (TMMO) [3], [4], [37], [38].

Self organizing scouts (SOS) [39] is a multi-population approach which utilizes a large subpopulation for global search and a number of small subpopulations for tracking changes of the identified peaks. SOS is one of the first methods which proposed TMMO. This strategy with some modifications has also been used with other metaheuristics such as PSO [40]–[43], differential evolution (DE) [44], [45], and artificial fish swarm optimization [46], [47].

In [42], two multi-population methods, called MQSO and MCPSO, were proposed which use quantum and charged particles for maintaining diversity. The population size is equal for every sub-swarm, and the number of sub-swarms is fixed and predetermined. An anti-convergence method ensures continued search for possible better peaks. The problem with having a fixed number of subpopulations is that the algorithm either misses some peaks, or wastes computational resources due to redundant subpopulations. Although these two methods rely on an exclusion mechanism to avoid several populations to converge on the same peak, their reliance on knowing the actual number of peaks to determine the exclusion radius violates the black-box assumption.

Other methods used a dynamic number of subpopulations in which regrouping and splitting models were utilized for creating subpopulations [4], [37]. Algorithms such as species-based PSO (SPSO) [48] and the randomized regrouping multi-swarm PSO [49] regroups individuals every generation/iteration or when a predefined criterion is satisfied. In [50], [51], a method based on hierarchical clustering was proposed for developing subpopulations whenever a change is detected. The splitting approaches generate subpopulations by dividing a main population when a certain criterion is met [52], [53].

AMQSO [43] was the first adaptive number of subpopulations in which the algorithm performed a continual search for new peaks and adapts the number of subpopulations to the number of detected peaks. Different algorithms with adaptive number of subpopulations mechanisms have been proposed [46], [54]–[58]. Since AMQSO adapts the number of subpopulations to the number of peaks, it can adjust the exclusion radius without having access to the actual number of peaks. Unlike MQSO which uses quantum particles during the course of optimization, AMQSO only uses them after an environmental change [4], resulting in substantial savings of computational resources. The tracking moving optima with adaptive number of subpopulations and the exclusion mechanisms used in this paper are based on AMQSO. For diversification however, we use a simple random sampling mechanism around the best solution immediately prior to an environment change [40], [58], [59].

A multi-population DE (DynDE) was proposed in [44] for solving DOPs. DynDE uses Brownian particles around the best found position to improve exploitation. An improved version of DynDE was proposed by Plessis and Engelbrecht [45] by modifying its exclusion mechanism and adding a resource allocation mechanism which prioritizes the optimization of promising peaks. Although this mechanism results in a faster convergence within each environment, the computational resources are still wasted due to the continual optimization of an already stagnant population. This type of resource allocation only reduces the offline error which is based on averaging of the current error across all environments [20], but does not necessarily improve the overall performance by transferring the resources from an stagnant population to those that can still improve. Yazdani et al. [58] partially addressed this problem by using a hibernation mechanism to avoid optimization of stagnant populations; however, their approach does not take the imbalance and the relative contribution of tracker swarms into account. The resource allocation mechanism proposed in this paper addresses this issue.

Recently, in [60], for the first time, partially separable DOPs were investigated and a divide-and-conquer method was used in order to solve them. This method uses differential grouping [23] for detecting interactions between decision variables and uses a species-based PSO as its optimizer [48], [61]. A major drawback of this algorithm is the assumption that the number of peaks in each subfunction and the number of generations between successive environmental changes are known *a priori*, which violates the black-box assumption.

Despite the importance of modularity, heterogeneity, imbalance and high-dimensionality in many real-world problems [21], [62]–[64], very few studies are dedicated to address these issues in dynamic optimization. The only research in which modularity and high-dimensionality were investigated is [60]. However, this research did not consider heterogeneity and imbalance which are common in modular problems [21]. Limited research about the effect of scalability, modularity, imbalance and heterogeneity in dynamic optimization is partly due to a lack of suitable benchmarks, which is the topic of next section.

D. DOP Benchmarks

In this section, we review the well-known dynamic optimization benchmarks relevant to the current study, i.e., those which are continuous, single-objective, and unconstrained [4].

In [65] a switching function method was proposed in which two landscapes A and B are used to generate the following three types of change: 1) Linear translation of peaks in A; 2) Changing the location of the optimum randomly while the rest of the search space remain unchanged; and 3) Switching between landscapes A and B.

Branke's Moving Peaks Benchmark (MPB) [20], [66] is the most widely used benchmark in DOP. The search space generated by this benchmark consists of several peaks whose width, height, and location change over time. MPB is very flexible to generate functions with configurable dimensions, number of peaks, and peak dynamics. In the standard MPB, the widths and heights of peaks are changed by adding Gaussian noise.

Similar to MPB, DF1 [67], [68] generates problem instances in which the width, height, and location of peaks change over time. The nature of the changes can be controlled by a logistic function to generate fixed, chaotic, or bifurcated step sizes. Another benchmark whose landscape consists of several peaks is Gaussian peak [69]. In this benchmark, the location of peaks change in random directions and the step sizes are uniformly distributed over an interval controlled by two levels of severity called abrupt and gradual [69].

Generalized dynamic benchmark generator (GDBG) [70], [71] can be instantiated into the binary space, real space and combinatorial space. GDBG provided six properties of the environmental dynamics including small step change, large step change, random change, recurrent change, recurrent change with noise, and chaotic change. These environmental dynamics were used in some other studies such as [72], [73], in which the width and height of each peak changed using them.

Dynamic rotation uses rotation for creating dynamic changes [74]. In this benchmark, the landscape is combined with a visibility mask which allows a percentage of the search space to be masked with a predefined fitness value. The rotation dynamic benchmark generator (RDBG) [71], [75] is another benchmark generators that uses rotation to generate environmental changes in continuous space. The magnitude of change in RDBG is defined using a rotation angle.

Although all the previous benchmarks are scalable, they all lack modularity which is an important feature of many real-world problems. One way of modularizing benchmarks is through summation of several independent benchmarks which is a common practice in large-scale global optimization [21], [75], [76]. However, in addition to modularity, the benchmark should exhibit heterogeneity and imbalance features to resemble real-world problems [21], [62]–[64], [75]. In [60] a modularized MPB was proposed; however, the generated problem instances lack heterogeneity and imbalance. Generating problem instances to resemble real-world problems is the motivation behind proposing a new benchmark in this paper.

III. The Proposed Benchmark Generator

The moving peaks benchmark (MPB) [20] is the most popular benchmark suite in dynamic optimization. MPB generates a landscape containing several peaks whose height, width, and location change over time. As a result, each peak can become the global optimum after an environmental change according to its current height and width. Standard baseline function of MPB is as follows:

$$f^{(t)}(\mathbf{x}) = \max_{i\in\{1,\dots,m\}} \left\{ h_i^{(t)} - w_i^{(t)} \left\| \mathbf{x} - \mathbf{c}_i^{(t)} \right\| \right\}, \qquad (5)$$

where m is the number of peaks, $\mathbf{x}$ is a solution in the problem space, $h_i^{(t)}$, $w_i^{(t)}$ and $\mathbf{c}_i^{(t)}$ are the height, width, and the center of the ith peak in the tth environment, respectively.

Although MPB can be scaled to any number of dimensions, its lack of modularity limits its capacity for large-scale DOPs. This limitation comes from the nonseparable nature of the benchmark. Section S-I in the supplementary document shows a mathematical variable interaction analysis on the MPB benchmark to demonstrate its nonseparable nature.

One way of modularizing MPB is through summation of several independent MPBs. This is customary in many large-scale global optimization benchmarks [75], [76] and has been recently used in [60] to propose a modularized MPB. Three major shortcomings of this benchmark are: a lack of imbalance among components, uniform component sizes, and unrealistic homogeneous structures. Many real-world problems, however, are heterogeneous in nature which is caused by the coexistence of separable and nonseparable components, each having a different share in improving the objective function [21].

In this paper, we address these shortcomings by proposing a new scalable benchmark, Composite MPB (CMPB), through heterogeneous composition of several MPBs. CMPB uses the standard MPB (Equation (5)) as its component function and has the following general form:

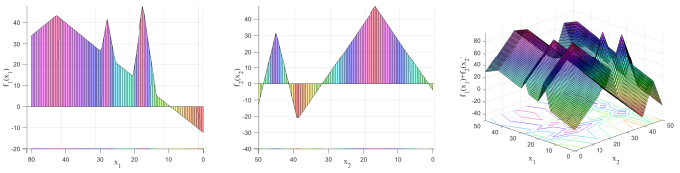
$$F^{(t)}(\mathbf{x}) = \sum_{i=1}^{k} \left(\omega_i f_i^{(t)}(\mathbf{x}_i) \right) + \sum_{j=k+1}^{k+l} \left(\omega_j \gamma f_j^{(t)}(x_j) \right), \qquad (6)$$

where the first summation term generates k nonseparable components, and the second summation term generates an l-dimensional separable component. Here f_i is the ith nonseparable subfunction which is a d_i-dimensional MPB ($d_i > 1$), f_j is the jth 1-dimensional MPB, $\mathbf{x}$ is the decision vector of D dimensions, $\mathbf{x}_i$ is a disjoint sub-vector of $\mathbf{x}$ with $d_i \geq 2$, x_j is a 1-dimensional scalar variable, ω_i and ω_j control the contribution of each component (for generating imbalance), and γ is a regulatory factor controlling the dominance of the separable component which is the reciprocal of the average dimensionality of the nonseparable components:

$$\gamma = \frac{k}{\sum_{i=1}^{k} d_i}. \qquad (7)$$

According to (5), the contribution of various MPBs is almost identical. This is because the height and the width parameters are usually sampled from the same distribution for different instances of MPB and the use of the max function also dampens the contrasts between various instances of MPB. Therefore, in (6) a large number of separable variables can easily dominate the final function value, $F^{(t)}$, which limits the utility of the benchmark to study a wide range of scenarios. To alleviate this issue, γ is used to regulate the dominance of one component over another. As can be seen, γ is a function of k and d_i and is calculated automatically when the number of nonseparable components and their dimensions are chosen. Only after this regularization, the imbalance coefficients (ω_i and ω_j) make intuitive sense and can be freely picked by the user to generate different imbalance patterns. By assigning different values to ω_i and ω_j, it is possible to generate problem instances in which different subfunctions have different contributions to the total fitness value which resembles the imbalance characteristics of some real-world problems.

For each MPB in the CMPB, the height, width, and center of a peak change from one environment to the next by the



(a) 1-dimensional MPB with 3 peaks (b) 1-dimensional MPB with 2 peaks (c) 2D CMPB with 6 peaks by adding (a) and (b).

Fig. 1. Exponentially growing number of peaks by composing MPBs.

following equations:

$$h_i^{(t+1)} = h_i^{(t)} + \alpha_i \mathcal{N}(0,1), \tag{8}$$

$$w_i^{(t+1)} = w_i^{(t)} + \beta_i \mathcal{N}(0,1), \tag{9}$$

$$\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \tag{10}$$

$$\mathbf{v}_i^{(t+1)} = \frac{s_i \mathbf{r}}{\|\mathbf{r}\|}, \tag{11}$$

where $\mathcal{N}(0,1)$ is a random number drawn from a Gaussian distribution with mean 0 and variance 1, α_i is the height severity, β_i is the width severity, s_i is the shift severity of the ith peak, and the components of the vector $\mathbf{r}$ are uniformly drawn from $[-0.5, 0.5]$. The reason that each peak has its own width, height, and shift severity is to simulate problems in which different regions change with different intensity. The parameter settings of CMPB are shown in Table I.

An interesting and natural consequence of CMPB's design is the exponential growth in the total number of peaks as the number of multi-modal components increases. This is a new challenge never addressed in either large-scale global optimization or dynamic optimization. In CMPB, when we compose several MPBs according to (6), the total number of peaks is:

$$M = \prod_{i=1}^{k+l} m_i, \tag{12}$$

where m_i is the number of peaks in the ith MPB subfunction (represented by f_i and f_j in (6)). It should be noted that M is the maximum number of peaks that can exist in the landscape, which may change over time due to coverage of smaller peaks by larger ones. For the sake of clarity, we provide an illustrative example. In Fig. 1(a) and Fig. 1(b), two 1-dimensional MPBs with 2 and 3 peaks are shown. The 2-dimensional function constructed based on (6) with $\omega_1 = \omega_2 = 1$ results in a total of $2 \times 3 = 6$ peaks. A consequence of this is that even for low-dimensional functions of this form, variable interaction analysis and problem decomposition can significantly simplify the problem. Indeed, an ideal decomposition can reduce the maximum number of peaks down to $\sum_{i=1}^{k+l} m_i$ which is significantly smaller than (12) for problems with large number of peaks and components. In the next section, we propose a decomposition-based framework that has this feature.

IV. The Proposed Framework

In this section, we propose a cooperative coevolutionary multi-population framework for solving large-scale DOPs. We first provide an overview of the framework with an emphasis on its high-level structure and the resource allocation policy (Section IV-A). We then focus on the details of our multi-population part of the framework and address dynamic issues such as convergence detection of populations, avoiding mutual convergence of populations onto the same peak, diversity control, and detection and handling of environmental changes (Section IV-B).

A. The high-level structure of the proposed framework

Algorithm 2 shows the structure of the proposed framework. The framework has three major parts – decomposition, search and resource allocation, and change management – which are explained next. In addition to Algorithm 2, Fig. S-1 in the supplementary document illustrates the flow chart of the proposed framework.

1) Decomposition: The framework starts by decomposing a given dynamic optimization problem into its constituent independent components (Algorithm 2, line 1). This is done using a variable interaction analysis algorithm. In this paper, we use the state-of-the-art DG2 algorithm [24] introduced in Section II. After problem decomposition, a multi-population dynamic optimizer is initialized for each of the identified components (Algorithm 2, lines 2-3). It should be noted that each component contains partial solutions which cannot be evaluated directly using the objective function. Due to the black-box nature of the objective function, these partial solutions can only be evaluated within the context of a complete solution referred to as a *context vector* [28] which is randomly initialized on line 4.

Next, the framework enters its main loop and optimizes the identified lower-dimensional components in an iterative manner (Algorithm 2, lines 5-41). The framework has three major phases: 1) exploration, 2) exploitation, and 3) change management. In the first phase (Algorithm 2, lines 6-23), the framework cycles over all components with the aim of tracking optima, discovering any emerging optima, and estimating the contribution of each component in improving the overall objective function value. For this purpose, the framework maintains a free population and a set of tracker populations for each component. The primary purpose of a free population is to find uncovered peaks. When a free population has converged to a peak, it will change to a tracker population whose primary purpose is to do exploitation and track it after each environmental change. For better use of the limited computational resources between successive environmental changes, the framework detects and deactivates the converged tracker populations based on a mechanism which will be explained in the next section.

2) Search and Resource Allocation: The exploitation phase (Algorithm 2, lines 24-29) is a crucial step in improving the efficiency of the framework. First, the tracking of multiple moving optima is inherently expensive. Second, the contribution of components are not uniform, making the classic round-robin optimization policy very inefficient. The imbalance in the contribution happens for several reasons. Two major factors are nonuniform change severity of components after

an environmental change, and discrepancy in the convergence behavior of populations.

For best use of the available resources, the exploitation phase occurs at two levels: component level, and population level. At the component level, the best contributing component is selected and all its active tracker populations are executed for an extra iteration (Algorithm 2, lines 24-26). The amount that each component improves the objective value at the end of the exploration phase is taken as its contribution. This often happens for the component experiencing the most intense environmental change. Therefore, by allocating more computational resource to such populations, the algorithm accelerates the optimization process by prioritizing components with higher importance or higher change severities. Finally, at the population level, the best tracker of each component is executed for one more iteration (Algorithm 2, lines 27-29). This step not only gives more resources to the best performing tracker, but also keeps the information about the best partial solutions up-to-date for the purpose of updating the context vector.

3) Change Management: Finally, the last phase deals with the environmental changes and updating of the context vector. Two events trigger the updating of the context vector. The first and most obvious case is the detection of an environmental change (Algorithm 2, lines 30-37). The second is prior to the deployment of a solution (Algorithm 2, lines 38-40). In DOPs, the algorithm is given a predefined time frame within which it has to respond to an environmental change. We denote this with η which is the maximum number of function evaluations available to the optimizer before providing a solution for deployment. It should be noted that in classic CC, the context vector is updated at every coevolutionary cycle. This is a costly operation because all solutions whose fitness were calculated with a previous version of the context vector have to be re-evaluated. However, owing to the grouping accuracy of DG2 and the independent nature of the components, this operation can be delayed until it becomes necessary (due to a dynamic change).

Algorithm 2: The Proposed Framework

1 $\mathcal{G} = \texttt{Grouping}(f)$;
2 **forall** $\mathcal{G}$ **do**
3 $\mathbf{P}_{\text{free}} \leftarrow$ Initialize the free population;
4 $\mathbf{c} \leftarrow$ Randomly initialize the context vector;
5 **repeat**
6 **forall** $\mathcal{G}$ **do**
7 $(\mathbf{P}_{\text{free}}, \mathbf{g}^{\star}_{\text{free}}) = \texttt{Optimizer}(\mathbf{P}_{\text{free}}, \mathbf{g}^{\star}_{\text{free}})$;
8 **if** *diversity of the free population is* $< r_{\text{conv}}$ **then**
9 Change its status to tracker population;
10 $\mathbf{P}_{\text{free}} \leftarrow$ Initialize a new free population;
11 **foreach** *tracker population* i **do**
12 **if** $\|\mathbf{g}^{\star}_{\text{free}} - \mathbf{g}^{\star}_{i}\| < r_{\text{excl}}$ **then**
13 $\mathbf{P}_{\text{free}} \leftarrow$ Reinitialize the free population;
14 **if** i^{th} *tracker population is active* **then**
15 $(\mathbf{P}_i, \mathbf{g}^{\star}_i) = \texttt{Optimizer}(\mathbf{P}_i, \mathbf{g}^{\star}_i)$;
16 **if** *the diversity is* $< r_{\text{deact}}$ **then**
17 Deactivate the tracker population;
18 **foreach** *tracker populations* j **do**
19 **if** $\|\mathbf{g}^{\star}_{i} - \mathbf{g}^{\star}_{j}\| < r_{\text{excl}}$ **then**
20 **if** $f(\mathbf{g}^{\star}_{i}) < f(\mathbf{g}^{\star}_{j})$ **then**
21 Remove i^{th} tracker population;
22 **else if** $f(\mathbf{g}^{\star}_{i}) > f(\mathbf{g}^{\star}_{j})$ **then**
23 Remove j^{th} tracker population ;
24 Determine the component H with the highest progress;
25 **forall** *active tracker populations* i *in* H **do**
26 $(\mathbf{P}_i, \mathbf{g}^{\star}_i) = \texttt{Optimizer}(\mathbf{P}_i, \mathbf{g}^{\star}_i)$;
27 **foreach** $\mathcal{G}$ **do**
28 Determine the best tracker population b;
29 $(\mathbf{P}_b, \mathbf{g}^{\star}_b) = \texttt{Optimizer}(\mathbf{P}_b, \mathbf{g}^{\star}_b)$;
30 **if** *an environmental change is happened* **then**
31 $\mathbf{c} \leftarrow$ Update context vector using best found position in each population $\mathbf{g}^{\star}$;
32 **forall** $\mathcal{G}$ **do**
33 Re-evaluate all individuals of the free population;
34 **forall** *trackers* **do**
35 Update estimated shift severity by Eq. (15);
36 Activate if is deactivated;
37 Increase diversity by Eq. (14);
38 **if** *computational budget* η *is finished* **then**
39 $\mathbf{c} \leftarrow$ Update context vector using best found position in each population $\mathbf{g}^{\star}$;
40 Re-evaluate all individuals in all populations;
41 **until** *stopping criterion is met*;

B. Dynamic Considerations

The aim of the framework on each component is to find all peaks and track them. However, due to the lack of information about the number of peaks, and also the coverage of some smaller peaks by larger ones in some of the environments, a free population needs to constantly search for possible uncovered peaks. Once a new optimum is found by a free population, it changes to a tracker population. To test the convergence of a free population, we use the procedure in which the Euclidean distances between all pairs of individuals are calculated. If all calculated distances are smaller than a given threshold (r_{conv}), it is assumed that the free population is converged [43]. When a free population becomes a tracker population, a new free population will be initialized immediately in the search space in order to search for another uncovered peak. It is possible that a free population converges to a peak already covered by a tracker population. Tracking a peak by multiple populations wastes a considerable amount of computational resource. Therefore, a mutual exclusion principle is enforced to avoid more than one population to cover the same peak. To establish the mutual exclusion, we use the mechanism proposed in [43]. According to the exclusion mechanism, when Euclidean distances between the global best of the free population and a tracker population is less than a threshold (r_{excl}), the algorithm assumes that the free population has converged to a covered peak. In this situation, the free population will be re-initialized. The value of r_{excl} is calculate as follows:

$$r_{\text{excl}} = 0.5 \frac{\text{SR}}{\sqrt[D]{\text{TN}}}, \tag{13}$$

where SR is the range of search space and TN is the number of trackers.

A similar conflict can also happen to two trackers when a peak is covered by a larger peak. Therefore, its tracker loses its own peak and starts converging to the larger peak's center. A similar situation happens when the convergence of a free population is detected before it enters into the mutual

exclusion area of a covered peak. As a result, it becomes a tracker population and moves toward the peak's center. This is another case where the exclusion principle is enforced to control the computational overhead. To do so, the tracker with the second best found position's fitness value $f(\mathbf{g}^\star)$ will be removed. For determining tracker populations which are under exclusion condition, the Euclidean distance between all pairs of trackers' $\mathbf{g}^\star$ position is calculated and compared with r_{excl} based on (13).

Another critical challenge of the population-based optimization algorithms in DOPs is diversity loss. According to [4], there are two main groups of methods to address this challenge. First is the reaction methods which introduce diversity after each environmental change, and second is diversity maintenance methods which try to keep the diversity of population above a certain level over time.

Our multi-population part of the framework uses a reaction type method in which the trackers' diversities are increased at the beginning of each environment. When a change is detected, for each tracker, one of the individuals is located on the $\mathbf{g}^\star$ position from the previous environment and other individuals are randomized around the $\mathbf{g}^\star$ position with radius of shift severity of the peak by (14):

$$\mathbf{P}_{i,j} = (s_i \cdot \mathbf{r}) + \mathbf{g}_i^{\star(t-1),\text{end}}, \tag{14}$$

where $\mathbf{P}_{i,j}$ is the position of the jth individual of the ith tracker population and $\mathbf{g}_i^{\star(t-1),\text{end}}$ is its best found position at the end of the previous environment, s_i is the shift severity of the peak which is under cover of the ith tracker population, and $\mathbf{r}$ is a uniformly distributed random number vector in range $[-1, 1]$. The reason for using s_i in (14) is that the new location of the peak after environmental change is expected to be inside that radius from the previous peak center. In (14), the $\mathbf{g}^*$ from the end of the previous environment is used instead of previous peak center position. Therefore, the diversity is introduced to the population of each tracker as much as needed. The shift severity of each peak is estimated by (15):

$$\hat{s}_i = \frac{1}{t - b_i - 1} \cdot \sum_{k=b_i+1}^{t-1} \left\| \mathbf{g}_i^{\star k,\text{end}} - \mathbf{g}_i^{\star(k-1),\text{end}} \right\|, \tag{15}$$

where $\hat{s}_i$ is the estimated shift severity of the peak covered by ith tracker population, b_i is the time index of the environment that ith tracker population has started tracking the peak, t is the current environment time index and $\mathbf{g}_i^{\star k,\text{end}}$ is the global best position of the ith tracker population at the end of the kth environment.

Another diversity related issue is detection and deactivation of converged trackers to save computational resources. When a tracker population gets sufficiently close to the center of a peak, it should be deactivated until the next environment. A tracker population is deactivated when its diversity drops below a certain threshold. To measure the diversity of a tracker population the infinity norm distance between all pairs of individuals is calculated. If all distances fall below a predefined value (r_{deact}), the tracker population will be deactivated which means that its individuals freeze until another environmental change is detected. r_{deact} is a positive constant number. A positive attribute of using infinity norm distance here is that it is independent from dimension number.

TABLE I
PARAMETER SETTINGS OF CMPB

Parameter	Symbol	Value
Number of peaks	m	Randomized* between 1 to 10
Dimension	D	1-200
Evaluations between changes	$\mathfrak{f}$	$500D$
Shift severity	s	Randomized* $\in [0.5, 3]$
Height severity	α	Randomized* $\in [3, 10]$
Width severity	β	Randomized* $\in [0.5, 1.5]$
Peaks shape	–	Cone
Peaks location range	SR	[-50,50]
Peak height	h	[30,70]
Peak width	w	[1,12]
Initial height value	–	50
Initial width value	–	6
Number of environments	–	100
Weight	ω	Randomized* $\in [0.5, 2]$

* Randomized with uniform distribution.

Another challenge of DOPs is outdated memory which happens after each environmental change due to the outdated stored fitness values of positions such as $\mathbf{g}^\star$. For addressing this issue, after each environmental change, the fitness values of all necessary positions (depends on the component optimizer) of the free population will be re-evaluated. For tracker populations, after re-diversification, the fitness values of the necessary positions are evaluated. For example, if PSO is embedded into the framework, the fitness values of particle positions are evaluated and the personal best positions are set to the particle positions.

The final main challenge is change detection. Since detecting a change is a separate issue and in most real-world dynamic environments the occurrence of a change is obvious (e.g., arrival of new order, change in temperature) [38], in this paper, we assume that the framework will be informed when an environmental change happens. However, it should be noted that environmental changes can be detected easily in many problems (including the ones that we investigate in this paper) by re-evaluating some beacons [4].

V. EXPERIMENTS AND ANALYSIS

The experiments in this section are based on different scenarios of the CMPB framework described in Section III. Section V-A covers the experimental settings, and Section V-B covers the experimental analysis which contains three sets of experiments. The first set is concerned with investigating the efficacy of decomposition and resource allocation in the proposed framework (Section V-B1), the second set is concerned with investigating the robustness of the proposed framework with respect to various aspects of DOPs, such as the number of peaks, shift severities, and change frequencies (Section V-B2), and the third is to investigate the effect of different component optimizers on the relative performance of the framework (Section V-B3) to demonstrate that the results hold independent of the component optimizer used.

A. Experimental Design

TABLE II
SUMMERY OF UTILIZED APPROACHES IN THE FRAMEWORKS

Framework	Cooperative coevolutionary	Tracking multiple moving optima	Resource allocation
CTR	✓	✓	✓
CT	✓	✓	×
C	✓	×	×
T	×	✓	×

1) Compared Frameworks: To study the effectiveness of different components of the proposed framework, i.e., cooperative coevolution (indexed by C), tracking of moving optima (indexed by T), and resource allocation (indexed by R), we generated four different frameworks which take these components into account in isolation as well as together. These cases are summarized in Table II.

The first framework (T) has no decomposition or resource allocation mechanism and is a representative of the classic TMMO algorithms. For a fair comparison, this framework uses the multi-population approach presented in Section IV-B. The second framework (C) is a simple CC framework which uses DG2 for problem decomposition and uses a single-population optimizer for each component. This framework represents large-scale static methods with no designated multiple optima tracking mechanism. After each environmental change, it simply re-initializes the subpopulations while maintaining the best solution. The third framework (CT) is the combination of the previous two cases, which is identical to our proposed framework (CTR) with the exception of the resource allocation mechanism. CT represents the state-of-the-art GCM-PSO [60] and replicates its major features and unifies its underlying decomposition and the multi-population optima tracking mechanisms for a fair comparison. The last framework (CTR) represents our proposed framework with all three features active.

All the frameworks presented above can be used with any component optimizer. For our empirical analysis, we use four popular component optimizers: PSO [77], jDE [78], [79], DynDE [44], [80], and CMAES [81]. For the experiments in Sections V-B1 and V-B2, we use PSO as the component optimizer due to its popularity in the dynamic optimization literature [4], [37]. The remaining algorithms are used in Section V-B3 to test the effect of different component optimizers on the proposed frameworks and to show that main conclusions are independent of the component optimizer used.

2) Parameter Settings: The parameter settings of all algorithms and frameworks are given in Table III. The right column shows whether the settings are taken from the original reference or from the sensitivity analysis results reported in the supplementary document. The parameters common to all algorithms are also listed at the bottom of the table. For all frameworks, the context vector is updated only after environmental changes and when the computational budget η is used. The default value of η is $\mathfrak{f}-1$ which means we fetch the solution at the end of each environment.

3) Performance Indicator: To measure the efficiency of algorithms, the average error of the updated context vector at the time of deployment (determined by η) after each environmental change is used as the measure of performance:

$$P = \frac{1}{T}\sum_{t=1}^{T}\left(f^{(t)}\left(\text{Optimum}^{(t)}\right) - f^{(t)}\left(\mathbf{c}^{(t),\eta}\right)\right), \quad (16)$$

where $\mathbf{c}^{(t),\eta}$ is the context vector at the tth environment which is updated after η fitness evaluations since the beginning of the new environment.

TABLE III
PARAMETER SETTINGS

Alg.	Param.	Frameworks				Ref.
		CTR	CT	T	C	
CMAES	λ	5			30	Tables S-IV, S-VIII
	μ	$\lfloor\lambda/2\rfloor$				[81]
jDE	NP	7			20	Tables S-II, S-VI
	Cr	self adaptive $\in [0, 1]$				[78]
	F	self adaptive $\in [0.1, 0.9]$				[78]
	strategy	$DE/rand/1/bin$				[78]
DynDE	NP	10			60	Tables S-III, S-VII
	Brownian	3				Tables S-III, S-VII
	F, Cr	random uniform $\in [0, 1]$				[80]
	strategy	$DE/best/2/bin$				[80]
PSO	$C_1 = C_2$	2.05				[82]
	χ	0.729843788				[82]
	swarm size	5 for $d \leq 5$			50	Tables S-I, S-V
		7 for $5 < d \leq 7$			50	Tables S-I, S-V
		10 for $d > 10$			50	Tables S-I, S-V
Common Parameters						
	r_{deact}	0.1	–	–	–	Table S-XI
	r_{conv}	r_{excl}			–	[43]

B. Empirical Analysis

To compare the performance of the four algorithms, we test them on 20 functions with various characteristics created using the CMPB benchmark generator. The suite contains functions with five different variable interaction structures tested in 25-, 50-, 100-, and 200-dimensional spaces (Table IV). The statistical results are based on 31 independent runs and their median are reported for comparison (mean and standard error are reported in the supplementary document). To test the statistical significance, we perform a multiple comparison test based on a series of pairwise Wilcoxon signed-rank tests with Holm-Bonferroni p-value correction with $\alpha = 0.05$. Highlighted entries are not statistically different from the best result.

1) The Overall Comparison: For the experiments in this section, the dynamic parameters of the functions listed in Table IV are set according to the default values reported in Table I. The obtained results by PSO_{CTR}, PSO_{CT}, PSO_{T}, and PSO_{C} on f_1 to f_{20} are summarized in Table V. The table clearly shows that PSO_{CTR} performs significantly better than all other algorithms on majority of the functions. Exceptions are the fully nonseparable functions (f_5, f_{10}, f_{15}, and f_{20}) for which no decomposition happens. It is notable that other decomposition-based algorithms, PSO_{CT} and PSO_{C}, also perform better than PSO_{T} which does not benefit from a decomposition mechanism. This clearly shows the benefit of problem decomposition for solving large-scale DOPs. Two

TABLE IV
BENCHMARK SCENARIOS BASED ON CMPB.

F	D	Dimensionality of Nonseparable Components	Separable
f_1	25	$\{2, 4, 6, 8\}$	5
f_2	25	$\{2, 5\}$	18
f_3	25	$\{2, 4, 5, 6, 8\}$	0
f_4	25	—	25
f_5	25	$\{25\}$	0
f_6	50	$\{2, 3, 5, 6, 7, 8, 10\}$	10
f_7	50	$\{2, 3, 5, 5\}$	35
f_8	50	$\{2, 2, 3, 5, 5, 5, 5, 5, 8, 10\}$	0
f_9	50	—	50
f_{10}	50	$\{50\}$	0
f_{11}	100	$\{2, 2, 3, 5, 5, 6, 6, 8, 8, 10, 10, 15\}$	20
f_{12}	100	$\{2, 2, 3, 3, 5, 5, 10\}$	70
f_{13}	100	$\{2, 2, 2, 2, 3, 3, 5, 5, 5, 5, 5, 5, 8, 8, 10, 10, 20\}$	0
f_{14}	100	—	100
f_{15}	100	$\{100\}$	0
f_{16}	200	$\{2, 2, 3, 5, 5, 6, 6, 8, 8, 10, 10, 15, 20, 20, 30\}$	50
f_{17}	200	$\{2, 3, 5, 10, 20, 30\}$	130
f_{18}	200	$\{2, 2, 2, 3, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 20, 20, 30, 50\}$	0
f_{19}	200	—	200
f_{20}	200	$\{200\}$	0

TABLE V
COMPARATIVE RESULTS OF $\mathrm{PSO_{CTR}}$, $\mathrm{PSO_{CT}}$, $\mathrm{PSO_C}$, AND $\mathrm{PSO_T}$ ON f_1 TO f_{20}. THE HIGHLIGHTED ENTRIES ARE SIGNIFICANTLY BETTER USING PAIR-WISE WILCOXON SIGNED-RANK TEST WITH HOLM p-VALUE ADJUSTMENT ($\alpha = 0.05$).

Dim	Function	$\mathrm{PSO_{CTR}}$	$\mathrm{PSO_{CT}}$	$\mathrm{PSO_C}$	$\mathrm{PSO_T}$
25D	f_1	2.35e+00	3.95e+00	5.93e+01	6.43e+01
	f_2	2.23e+00	3.24e+00	3.17e+01	9.55e+01
	f_3	1.19e+00	2.87e+00	5.71e+01	4.30e+01
	f_4	3.00e+00	3.19e+00	1.10e+01	3.21e+02
	f_5	1.84e+00	2.94e+00	1.38e+01	1.24e+00
50D	f_6	4.56e+00	8.77e+00	1.14e+02	1.77e+02
	f_7	4.42e+00	6.53e+00	6.68e+01	2.47e+02
	f_8	3.64e+00	5.95e+00	1.14e+02	2.07e+02
	f_9	6.08e+00	6.73e+00	2.17e+01	8.16e+02
	f_{10}	7.76e+00	8.38e+00	1.66e+01	8.05e+00
100D	f_{11}	1.05e+01	2.02e+01	2.13e+02	6.45e+01
	f_{12}	1.19e+01	1.51e+01	1.31e+02	5.71e+02
	f_{13}	1.05e+01	1.70e+01	1.98e+02	5.00e+02
	f_{14}	1.18e+01	1.32e+01	4.35e+01	2.14e+03
	f_{15}	3.61e+01	4.43e+01	3.47e+01	4.80e+01
200D	f_{16}	3.63e+01	5.19e+01	3.39e+02	9.78e+02
	f_{17}	3.77e+01	5.60e+01	2.92e+02	5.79e+02
	f_{18}	2.79e+01	3.77e+01	2.27e+02	1.17e+03
	f_{19}	2.38e+01	2.29e+01	8.14e+01	5.01e+03
	f_{20}	1.49e+02	1.98e+02	8.92e+01	1.75e+02

reasons can be attributed to the poor performance of $\mathrm{PSO_T}$ on majority of the functions. First is the scalability issue. It is clear that in the absence of problem decomposition, the dimensionality of a given problem can easily exceed the capacity of the optimizer. Second, is the exponential growth in the number of peaks when no decomposition is used (see Fig. 1).

In addition to problem decomposition, resource allocation is another major feature of $\mathrm{PSO_{CTR}}$. The effectiveness of the resource allocation mechanism can be checked by comparing it with $\mathrm{PSO_{CT}}$ whose only difference lies within its resource allocation policy. Table V clearly shows the superiority of $\mathrm{PSO_{CTR}}$ over $\mathrm{PSO_{CT}}$. Although problem decomposition plays a crucial role in simplifying a large-scale problem,

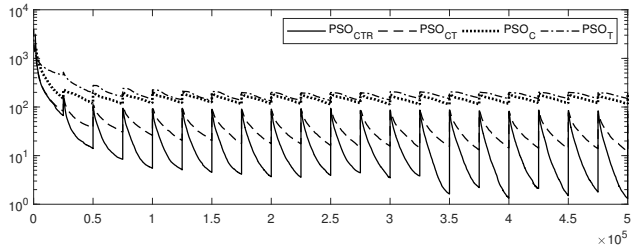
TABLE VI
OBTAINED RESULTS BY ALGORITHMS ON f_6 TO f_{10} WITH DIFFERENT NUMBER OF PEAKS m FOR EACH COMPONENT RANDOMIZED IN THE FOLLOWING RANGES $\{1, \ldots, 5\}$, $\{1, \ldots, 10\}$, AND $\{1, \ldots, 20\}$. OTHER PARAMETERS OF CMPB ARE SET AS SHOWN IN TABLE I.

# Peaks	$F(\mathbf{x})$	$\mathrm{PSO_{CTR}}$	$\mathrm{PSO_{CT}}$	$\mathrm{PSO_C}$	$\mathrm{PSO_T}$
$m \in \{1, \ldots, 5\}$	f_6	2.54e+00	5.57e+00	1.03e+02	1.90e+02
	f_7	2.36e+00	4.61e+00	6.01e+01	2.63e+02
	f_8	2.14e+00	3.01e+00	1.04e+02	2.08e+02
	f_9	3.41e+00	3.64e+00	1.92e+01	8.61e+02
	f_{10}	6.17e+00	8.08e+00	1.45e+01	7.75e+00
$m \in \{1, \ldots, 10\}$	f_6	4.56e+00	8.77e+00	1.14e+02	1.77e+02
	f_7	4.42e+00	6.53e+00	6.68e+01	2.47e+02
	f_8	3.64e+00	5.95e+00	1.14e+02	2.07e+02
	f_9	6.08e+00	6.73e+00	2.17e+01	8.16e+02
	f_{10}	7.76e+00	8.38e+00	1.66e+01	8.05e+00
$m \in \{1, \ldots, 20\}$	f_6	9.52e+00	1.33e+01	1.12e+02	2.02e+02
	f_7	7.04e+00	9.32e+00	6.73e+01	2.62e+02
	f_8	8.07e+00	1.21e+01	1.29e+02	2.55e+02
	f_9	9.55e+00	1.11e+01	2.47e+01	8.38e+02
	f_{10}	6.99e+00	8.42e+00	1.89e+01	8.36e+00

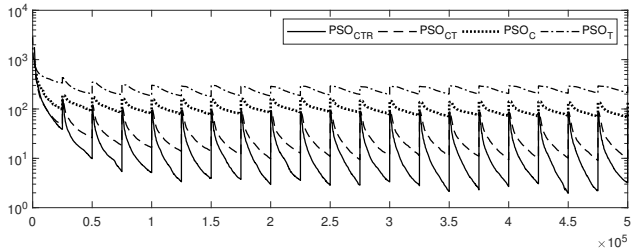
the existence of numerous components can impose a computational overhead on the algorithm. Additionally, use of a multi-population algorithm to optimize the components also adds to the computational complexity. The component-level and population-level resource allocation policies of $\mathrm{PSO_{CTR}}$ allow for an economical use of resources while preserving the simplifying effects of problem decomposition. The population-level mechanism prevents over-exploitation of trackers and releases more resources to be used by the best trackers to improve the overall solution quality. The component-level mechanism accelerates the convergence by allocating more resources to the component with maximum impact on the overall solution quality. On the fully nonseparable functions however (f_5, f_{10}, f_{15}, and f_{20}), the only active resources allocation mechanism is the population level. The relative high dimensionality of the only available component causes the population-level mechanism to lose its efficiency because of slow convergence and existence of many active populations.

Another interesting observation is a sharp contrast between the performance of multi-population methods ($\mathrm{PSO_{CTR}}$ and $\mathrm{PSO_{CT}}$) and the only single-population method ($\mathrm{PSO_C}$). These are all decomposition based where each component is optimized independently. $\mathrm{PSO_{CTR}}$ and $\mathrm{PSO_{CT}}$ use multiple populations for each component whereas $\mathrm{PSO_C}$ uses a single population for each component. All these methods benefit from an ideal decomposition which eliminates the issue of exponentially growing number of peaks. However, the comparison clearly shows that a special mechanism for tracking multiple moving optima should be in place to obtain acceptable results. In other words, simple mechanisms such re-initialization and injection of the best found solution into the population are not sufficient for efficient handling of environmental changes.

Fig. 2 shows the convergence plot of the four algorithms on f_6 and f_7. The convergence plots are based on current error of the context vector after each function evaluation for the first 20 environments. The figure shows that the algorithms try to find better solutions until the end of an environment where the error jumps due to an environmental



(a) Convergence plot for f_6



(b) Convergence plot for f_7

Fig. 2. Convergence plot of $\mathrm{PSO_{CTR}}$, $\mathrm{PSO_{CT}}$, $\mathrm{PSO_C}$ and $\mathrm{PSO_T}$ based on the average current error of 31 runs on f_6 and f_7 for the first 20 environments.

change. $\mathrm{PSO_{CTR}}$ and $\mathrm{PSO_{CT}}$ which are decomposition-based and track multiple optima outperform $\mathrm{PSO_T}$ and $\mathrm{PSO_C}$ across all environments. $\mathrm{PSO_{CTR}}$ has a clear advantage over $\mathrm{PSO_{CT}}$ due to its efficient resource allocation mechanism. As can be seen in Fig. 2, for the first environment, algorithms try to find uncovered peaks to track them after environmental changes. That is why the results obtained in the first environment are worse. This circumstance is more obvious for $\mathrm{PSO_{CTR}}$ and $\mathrm{PSO_{CT}}$ which suffer from uncovered peaks until the fifth environment. After this phase, the algorithms are more stable and their results are improved because most peaks are identified and the trackers can converge faster to the new optimum after each environmental change.

2) Robustness to Dynamic Changes: Table VI shows the results obtained by the four algorithms on f_6-f_{10} with different number of peaks[1]. The results show that the performance of all algorithms deteriorates as the number of peaks increases. However, $\mathrm{PSO_{CTR}}$ maintains the best performance across all three cases. For the multi-population algorithms ($\mathrm{PSO_{CTR}}$, $\mathrm{PSO_{CT}}$, and $\mathrm{PSO_T}$) the increase in the number of peaks results in more tracker populations, which increases the computational overhead of these algorithms. Among these methods, $\mathrm{PSO_T}$ has the worst performance and experiences an exponential growth in the number of peaks due to its lack of decomposition (see Fig. 1). $\mathrm{PSO_{CTR}}$ performs better than $\mathrm{PSO_{CT}}$ thanks to its resource allocation mechanism, which makes it less susceptible to an increase in the number of peaks (hence more trackers). $\mathrm{PSO_C}$, which maintains a single population, also suffers from an increase in the number of peaks. The reason is that the increased number of peaks adds to the complexity of the landscape and increasing the likelihood of a premature convergence.

Table VII shows the obtained results by the four algorithms on f_6-f_{10} with different shift severities[1]. It is clear that stronger shift severities, i.e., larger displacement in the location of a peak, makes tracking more difficult and time consuming. Table VII shows that $\mathrm{PSO_{CTR}}$ has the best overall performance across all three severity levels. The results clearly show that $\mathrm{PSO_{CTR}}$ has a better competitive advantage on problems with stronger shift severities. The resource allocation mechanism of $\mathrm{PSO_{CTR}}$ allows it to prioritize its limited computational resources for tracking of important peaks. On simpler problems with a smaller shift magnitude, other algorithms with no resource allocation mechanism such $\mathrm{PSO_{CT}}$ can also track the peaks with a relatively good efficiency and accuracy. This is because the amount of available function evaluations between successive environmental changes is large enough to track all the peaks accurately.

Table VIII shows the obtained results by the four algorithms on f_6-f_{10} with different change frequencies (ƒ)[1]. The results show that the performance of all methods declines when the change frequency is high (i.e., when the number of fitness evaluations between successive environmental changes is lower). A high change frequency means that the algorithm has limited time to do an accurate global and local search, which leads

TABLE VII
OBTAINED RESULTS BY $\mathrm{PSO_{CTR}}$, $\mathrm{PSO_{CT}}$, $\mathrm{PSO_C}$, AND $\mathrm{PSO_T}$ ON f_6 TO f_{10} WITH DIFFERENT SHIFT SEVERITY VALUES FOR EACH PEAK IN EACH COMPONENT. THE VALUES ARE RANDOMIZED IN THE FOLLOWING RANGES $[0.5, 1]$, $[0.5, 3]$, AND $[0.5, 5]$. OTHER PARAMETERS OF CMPB ARE SET AS SHOWN IN TABLE I.

Shift Severity	$F(\mathbf{x})$	$\mathrm{PSO_{CTR}}$	$\mathrm{PSO_{CT}}$	$\mathrm{PSO_C}$	$\mathrm{PSO_T}$
$S \in [0.5, 1]$	f_6	3.67e+00	6.74e+00	1.10e+02	1.50e+02
	f_7	3.79e+00	4.26e+00	6.26e+01	2.12e+02
	f_8	3.53e+00	4.02e+00	1.20e+02	1.73e+02
	f_9	7.63e+00	5.04e+00	1.87e+01	7.02e+02
	f_{10}	6.05e+00	6.43e+00	1.65e+01	6.53e+00
$S \in [0.5, 3]$	f_6	4.56e+00	8.77e+00	1.14e+02	1.77e+02
	f_7	4.42e+00	6.53e+00	6.68e+01	2.47e+02
	f_8	3.64e+00	5.95e+00	1.14e+02	2.07e+02
	f_9	6.08e+00	6.73e+00	2.17e+01	8.16e+02
	f_{10}	7.76e+00	8.38e+00	1.66e+01	8.05e+00
$S \in [0.5, 5]$	f_6	6.20e+00	1.14e+01	1.13e+02	2.49e+02
	f_7	5.84e+00	7.75e+00	7.01e+01	3.12e+02
	f_8	5.05e+00	7.60e+00	1.24e+02	2.89e+02
	f_9	5.93e+00	7.97e+00	2.25e+01	9.53e+02
	f_{10}	9.48e+00	1.03e+01	1.66e+01	9.48e+00

TABLE VIII
OBTAINED RESULTS BY $\mathrm{PSO_{CTR}}$, $\mathrm{PSO_{CT}}$, $\mathrm{PSO_C}$, AND $\mathrm{PSO_T}$ ON f_6 TO f_{10} WITH DIFFERENT CHANGE FREQUENCIES: $200D$, $500D$, AND $1000D$. OTHER PARAMETERS OF CMPB ARE SET AS SHOWN IN TABLE I.

Frequency	$F(\mathbf{x})$	$\mathrm{PSO_{CTR}}$	$\mathrm{PSO_{CT}}$	$\mathrm{PSO_C}$	$\mathrm{PSO_T}$
ƒ $= 200D$	f_6	1.83e+01	2.90e+01	1.93e+02	2.76e+02
	f_7	1.61e+01	2.00e+01	1.05e+02	3.32e+02
	f_8	1.33e+01	2.34e+01	1.73e+02	3.18e+02
	f_9	1.74e+01	2.79e+01	5.82e+01	1.05e+03
	f_{10}	1.20e+01	1.51e+01	1.97e+01	1.36e+01
ƒ $= 500D$	f_6	4.56e+00	8.77e+00	1.14e+02	1.77e+02
	f_7	4.42e+00	6.53e+00	6.68e+01	2.47e+02
	f_8	3.64e+00	5.95e+00	1.14e+02	2.07e+02
	f_9	6.08e+00	6.73e+00	2.17e+01	8.16e+02
	f_{10}	7.76e+00	8.38e+00	1.66e+01	8.05e+00
ƒ $= 1000D$	f_6	1.90e+00	2.84e+00	9.08e+01	1.62e+02
	f_7	2.16e+00	1.91e+00	4.55e+01	2.30e+02
	f_8	2.24e+00	2.13e+00	9.89e+01	1.84e+02
	f_9	3.54e+00	1.33e+00	1.16e+01	7.30e+02
	f_{10}	6.03e+00	6.12e+00	1.77e+01	6.75e+00

to degraded performance. Despite this, a desired property of PSO_{CTR} is its good performance on problems with a high change frequency. The results clearly show that the PSO_{CTR} gains a significant competitive edge over other algorithms on such problems. This can be attributed to its resource allocation mechanism which allows it to benefit from the saved resources to respond to rapid environmental changes more efficiently. This property is less crucial for problems with low change frequencies, due to the availability of sufficient time between environmental changes for accurate tracking of all the peaks.

3) The Effect of Component Optimizers: In this part, we investigate the influence of several component optimizers on the performance of the proposed framework. We compare the performance of the frameworks when PSO [82], jDE [78], [79], DynDE [44], [80], and CMAES [81], [83] are used as the optimizer. For all multi-population algorithms, we use the same mechanism described in Section IV-B. The reason is that our aim here is to confirm that our conclusions are independent of the component optimizer used, not the effect of different dynamic handling mechanisms.

The core procedure of CMAES [81] is very different from PSO and DE. Therefore, for embedding it in the frameworks, we need to carry out some modifications. Instead of the best found position, the mean position is used. For calculating the diversity of each population in $\text{CMAES}_{\text{CTR}}$, CMAES_{CT} and CMAES_{T}, we calculate the Euclidean distance between all pairs of offspring (i.e., prior to selection). After each environmental change, the mean positions of all free and tracker populations are re-evaluated. Moreover, for the free population, all other state variables remain unchanged. For trackers however, all state variables relating to the covariance matrix and the evolution path are reinitialized since the directions towards the new optima are unknown. For the ith tracker, the step-size (σ_i) is set to $\hat{s}_i/2$ where $\hat{s}_i$ is the estimated shift severity of its covered peak. The reason for choosing $\hat{s}_i/2$ for σ_i is that the offspring are normally distributed and approximately 95.4% of them are located within 2 standard deviations from the mean. This makes the diversity of new samples in CMAES similar to those of PSO and DE trackers after re-diversification by (14).

Table IX shows the results obtained by the four frameworks using PSO, CMAES, jDE and DynDE as component optimizers on f_6-f_{10} (see Table IV). The results clearly show that our proposed framework (CTR) consistently outperforms other cases independent of the chosen optimizer. Additionally, comparing the component optimizers across different frameworks shows that PSO has the best performance with CTR; however, comparing the efficacy of component optimizers is beyond the scope of this study. More comparative results obtained by the four frameworks from Table II with the above mentioned component optimizers on different 100 and 200-dimensional problems with different dynamic configurations of CMPB, can be found in Section S-V of the supplementary document.

[1]Other parameters of CMPB are set based on Table I.

TABLE IX
OBTAINED RESULTS BY THE FOUR FRAMEWORKS FROM TABLE II WITH DIFFERENT OPTIMIZERS INCLUDING PSO, JDE, DYNDE, AND CMAES ON f_6 TO f_{10} WITH DEFAULT PARAMETER SETTING OF CMPB (TABLE I).

Optimizer	F	Framework			
		CTR	CT	C	T
PSO	f_6	4.56e+00	8.77e+00	1.14e+02	1.77e+02
	f_7	4.42e+00	6.53e+00	6.68e+01	2.47e+02
	f_8	3.64e+00	5.95e+00	1.14e+02	2.07e+02
	f_9	6.08e+00	6.73e+00	2.17e+01	8.16e+02
	f_{10}	7.76e+00	8.38e+00	1.66e+01	8.05e+00
CMA-ES	f_6	6.70e+00	1.18e+01	8.22e+01	1.43e+03
	f_7	7.55e+00	1.25e+01	7.09e+01	1.16e+03
	f_8	4.20e+00	6.92e+00	1.02e+02	2.03e+03
	f_9	1.36e+01	1.67e+01	1.68e+02	2.62e+03
	f_{10}	1.11e+00	1.33e+00	1.08e+01	5.90e+02
DynDE	f_6	5.47e+00	1.04e+01	6.94e+01	1.43e+02
	f_7	4.84e+00	9.61e+00	4.14e+01	2.05e+02
	f_8	4.53e+00	5.74e+00	9.13e+01	1.64e+02
	f_9	4.86e+00	5.10e+00	1.25e+01	6.88e+02
	f_{10}	4.53e+00	4.92e+00	1.26e+01	5.20e+00
jDE	f_6	2.11e+01	3.43e+01	7.80e+01	1.19e+02
	f_7	1.62e+01	2.36e+01	5.76e+01	1.35e+02
	f_8	1.75e+01	2.50e+01	7.53e+01	1.31e+02
	f_9	1.08e+01	1.77e+01	8.75e+00	3.56e+02
	f_{10}	2.91e+00	2.60e+00	1.15e+01	3.95e+00

VI. CONCLUSION

In this paper, we presented a thorough investigation of large-scale dynamic optimization problems (DOPs). A formal analysis of the moving peaks benchmark (MPB) showed that its lack of modularity limits its applicability to the study of large-scale DOPs. A new benchmark generator based on MPB was proposed for large-scale DOPs. The benchmark was made by composing several weighted MPBs in which an automated weight regulates the equilibrium between fully separable and nonseparable components, and a manual weight creates artificial imbalance among the contributions of different components.

We also proposed a cooperative coevolutionary multi-population framework which benefits from a bi-level computational resource allocation mechanism capable of saving resources at both component and sub-population levels. We investigated the performance of the proposed framework against three other frameworks on a wide range of problems having different dimensions, interaction structures, shift severities, number of peaks, and change frequencies. The results showed that the proposed framework not only outperforms the peer frameworks, but also gains even a greater competitive advantage on more difficult problems with higher dimensionality, number of peaks, change frequency or shift severity.

Despite improving the baseline with up to two orders of magnitude, the proposed framework has several shortcomings that can limit its applicability in certain situations. The decomposition algorithm used in this paper cannot exploit the structure of problems with overlapping components. Many overlapping problems have sparse interaction matrices [21], [84]; however, the proposed framework does not have the necessary mechanisms to exploit this sparsity; therefore, treating them as fully nonseparable. Optimal decomposition of overlapping functions is an open question even in static

optimization, which becomes a greater challenge in dynamic environments. Another decomposition related issue is optimal grouping of separable variables. Although one may consider a full decomposition of separable variables into a series of 1-dimensional subproblems an obvious choice, empirical evidence suggests that such decomposition is suboptimal and increases the computational overhead of cooperative coevolution [85]. Grouping of separable variables may decrease this computational overhead; however, imbalance considerations and the phenomenon of exponentially growing "pseudo" peaks discussed in Section III makes finding an effective decomposition a nontrivial task. Finally, dealing with fully nonseparable problems with no apparent exploitable modularity is another important issue missing form the current study.

On the dynamic side, the proposed framework is primarily designed for tracking moving optima with no consideration about the cost of deploying a new solution after an environmental change. In many real-world situations however, frequent deployment of new solutions can incur additional cost [86], [87]. Although the ability to find and track moving optima precedes any deployment consideration, designing a framework capable of finding robust solutions over time has strong practical merits. Other practical considerations such as dealing with dynamic constraints [88], [89] and multiple conflicting objectives [90], [91] are important topics deserving further investigation.

REFERENCES

[1] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[2] H. Spencer, *The Principles of Biology, Volume 1 (of 2)*. Litres, 2017.

[3] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing*, vol. 15, no. 7, pp. 1427–1448, 2011.

[4] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[5] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: Learning deep neural networks on the fly," in *International Joint Conference on Artificial Intelligence*, 2018, pp. 2660–2666.

[6] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for projected clustering of high dimensional data streams," in *International Conference on Very Large Data Bases*, 2004, pp. 852–863.

[7] J. Beringer and E. Hüllermeier, "Online clustering of parallel data streams," *Data & Knowledge Engineering*, vol. 58, no. 2, pp. 180–204, 2006.

[8] X. Mingming, Z. Jun, C. Kaiquan, C. Xianbin, and T. Ke, "Cooperative co-evolution with weighted random grouping for large-scale crossing waypoints locating in air route network," in *International Conference on Tools with Artificial Intelligence*, 2011, pp. 215–222.

[9] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Information Sciences*, vol. 295, pp. 407–428, 2015.

[10] T. Hogg, "Exploiting problem structure as a search heuristic," *International Journal of Modern Physics C*, vol. 9, no. 01, pp. 13–29, 1998.

[11] U. Aickelin and K. A. Dowsland, "Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem," *Journal of Scheduling*, vol. 3, no. 3, pp. 139–153, 2000.

[12] C. J. Price and P. L. Toint, "Exploiting problem structure in pattern search methods for unconstrained optimization," *Optimisation Methods and Software*, vol. 21, no. 3, pp. 479–491, 2006.

[13] N. Ho-Nguyen and F. Kılınç-Karzan, "Exploiting problem structure in optimization under uncertainty via online convex optimization," *Mathematical Programming*, pp. 1–35, 2018.

[14] R. Santana, "Gray-box optimization and factorized distribution algorithms: where two worlds collide," *arXiv preprint arXiv:1707.03093*, 2017.

[15] J. Holland, *Hidden Order: How Adaptation Builds Complexity*. Perseus, 1995.

[16] G. R. Harik, "Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms," Ph.D. dissertation, University of Michigan, 1997.

[17] Y. Chen, T. Yu, K. Sastry, and D. Goldberg, "A survey of linkage learning techniques in genetic and evolutionary algorithms," *IlliGAL Report*, vol. 2007014, 2007.

[18] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Transactions on Mathematical Software*, vol. 42, no. 2, p. 13, 2016.

[19] Y. Sun, M. Kirley, and S. K. Halgamuge, "A recursive decomposition method for large scale continuous optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 647–661, 2018.

[20] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Congress on Evolutionary Computation*, 1999, pp. 1875–1882.

[21] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Information Sciences*, vol. 316, pp. 419–436, 2015.

[22] F. Wang, P. Bahri, P. L. Lee, and I. Cameron, "A multiple model, state feedback strategy for robust control of non-linear processes," *Computers & Chemical Engineering*, vol. 31, no. 5-6, pp. 410–418, 2007.

[23] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2014.

[24] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929–942, 2017.

[25] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[26] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[27] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2012.

[28] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[29] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Genetic and Evolutionary Computation Conference*, 2011, pp. 1115–1122.

[30] M. N. Omidvar, B. Kazimipour, X. Li, and X. Yao, "CBCC3 – a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance," in *Congress on Evolutionary Computation*, 2016, pp. 3541–3548.

[31] M. Yang, M. N. Omidvar, C. Li, X. Li, Z. Cai, B. Kazimipour, and X. Yao, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 493–505, 2017.

[32] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Multilevel framework for large-scale global optimization," *Soft Computing*, vol. 21, no. 14, pp. 4111–4140, 2017.

[33] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization," *Applied Intelligence*, vol. 47, no. 3, pp. 888–913, 2017.

[34] R. P. Wiegand, W. C. Liles, and K. A. D. Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *Genetic and Evolutionary Computation Conference*, 2001, pp. 1235–1242.

[35] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Congress on Evolutionary Computation*, 2010, pp. 1–8.

[36] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving from Nature*, 2010, pp. 300–309.

[37] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[38] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, University of Birmingham, 2011.

[39] J. Branke, T. Kaussler, C. Smidt, and H. Schmeck, "A multipopulation approach to dynamic optimization problems," in *Evolutionary Design and Manufacture*, 2000, pp. 299–307.

[40] D. Yazdani, B. Nasiri, R. Azizi, A. Sepas-Moghaddam, and M. R. Meybodi, "Optimization in dynamic environments utilizing a novel method based on particle swarm optimization," *International Journal of Artificial Intelligence*, vol. 11, pp. 170–192, 2013.

[41] C. Li and S. Yang, "Optimization in dynamic environments utilizing a novel method based on particle swarm optimization," in *International Conference on Natural Computation*, 2008, pp. 624–628.

[42] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[43] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," in *Swarm Intelligence: Introduction and Applications*, C. Blum and D. Merkle, Eds. Springer Lecture Notes in Computer Science, 2008, pp. 193–217.

[44] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," in *Congress on Evolutionary Computation*, vol. 3, 2005, pp. 2808–2815.

[45] M. C. du Plessis and A. P. Engelbrecht, "Using competitive population evaluation in a differential evolution algorithm for dynamic environments," *European Journal of Operational Research*, vol. 218, no. 1, pp. 7–20, 2012.

[46] D. Yazdani, A. Sepas-Moghaddam, A. Dehban, and N. Horta, "A novel approach for optimization in dynamic environments based on modified artificial fish swarm algorithm," *International Journal of Computational Intelligence and Applications*, vol. 15, no. 02, pp. 1 650 010–1 650 034, 2016.

[47] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. R. Meybodi, and M. Akbarzadeh-Totonchi, "mNAFSA: A novel approach for optimization in dynamic environments with global changes," *Swarm and Evolutionary Computation*, vol. 18, pp. 38–53, 2014.

[48] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 04, pp. 440–458, 2006.

[49] Y. Jiang, W. Huang, and L. Chen, "Applying multi-swarm accelerating particle swarm optimization to dynamic continuous functions," in *Knowledge Discovery and Data Mining*, 2009, pp. 710–713.

[50] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *Congress on Evolutionary Computation*, 2009, pp. 439–446.

[51] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 06, pp. 959–974, 2010.

[52] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A hibernating multi-swarm optimization algorithm for dynamic environments," in *World Congress on Nature and Biologically Inspired Computing*, 2010, pp. 363–369.

[53] C. Li and S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," in *International Conference on Natural Computation*, vol. 7, 2008, pp. 624–628.

[54] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multi-population framework for locating and tracking multiple optima," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 05, pp. 590–605, 2016.

[55] S. K.Nseef, S. Abdullah, A. Turky, and G. Kendall, "An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems," *Knowledge-Based Systems*, vol. 104, pp. 14–23, 2016.

[56] C. Li, T. T. Nguyen, M. Yang, S. Yang, and S. Zeng, "Multi-population methods in unconstrained continuous dynamic environments: The challenges," *Information Sciences*, vol. 296, pp. 95–118, 2015.

[57] C. Li, S. Yang, and M. Yang, "An adaptive multi-swarm optimizer for dynamic optimization problems," *Evolutionary Computation*, vol. 22, no. 4, pp. 559–594, 2014.

[58] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, "A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," *Applied Soft Computing*, vol. 13, no. 04, pp. 2144–2158, 2013.

[59] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Transactions on Evolutionary Computation*, 2018.

[60] W. Luo, B. Yang, C. Bu, and X. Lin, "A hybrid particle swarm optimization for high-dimensional dynamic optimization," in *Simulated Evolution and Learning*, Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin, Eds. Springer Lecture Notes in Computer Science, 2017, vol. 10593, pp. 981–993.

[61] M. Preuss, "Niching the CMA-ES via nearest-better clustering," in *Genetic and Evolutionary Computation Conference*, 2010, pp. 1711–1718.

[62] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, 1960.

[63] Z. Michalewicz, M. Schmidt, M. Michalewicz, and C. Chiriac, *Adaptive Business Intelligence*. Springer-Verlag Berlin Heidelberg, 2006.

[64] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep., 2010.

[65] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *International Conference on Genetic Algorithms*, 1993, pp. 523–530.

[66] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. Springer Natural Computing Series, 2003, pp. 239–262.

[67] R. W. Morrison and K. A. D. Jong, "A test problem generator for non-stationary environments," in *Congress on Evolutionary Computation*, vol. 3, 1999, pp. 2047–2053.

[68] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*. Springer-Natural Computing Series, 2004.

[69] J. J. Grefenstette, "Evolvability in dynamic fitness landscapes: a genetic algorithm approach," in *Congress on Evolutionary Computation*, vol. 3, 1999, pp. 2031–2038.

[70] C. Li and S. Yang, "A generalized approach to construct benchmark problems for dynamic optimization," in *Simulated Evolution and Learning*, X. L. et al., Ed. Springer Lecture Notes in Computer Science, 2013, vol. 5361, pp. 391–400.

[71] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for cec'2009 competition on dynamic optimization," Center for Computational Intelligence, Tech. Rep., 2008.

[72] Y. Huang, Y. Ding, K. Hao, and Y. Jin, "A multi-objective approach to robust optimization over time considering switching cost," *Information Sciences*, vol. 394-395, pp. 183–197, 2017.

[73] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Robust optimization over time: problem difficulties and benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 731–745, 2015.

[74] K. Weicker and N. Weicker, "Dynamic rotation and partial visibility," in *Congress on Evolutionary Computation*, 2000, p. 11251131.

[75] X. Li, K. Tang, M. N. Omidvar, Z. Yang, , and K. Qin, "Benchmark functions for the CEC2013 special session and competition on large-scale global optimization," RMIT University, Tech. Rep., 2013.

[76] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC2010 special session and competition on large-scale global optimization," Nature Inspired Computation and Applications Laboratory, Tech. Rep., 2009.

[77] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, vol. 04, 1995, pp. 1942–1948.

[78] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[79] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer, "Dynamic optimization using self-adaptive differential evolution," in *Congress on Evolutionary Computation*, 2009, pp. 415–422.

[80] M. C. du Plessis and A. P. Engelbrecht, "Using competitive population evaluation in a differential evolution algorithm for dynamic environments," *European Journal of Operational Research*, vol. 218, no. 1, pp. 7–20, 2012.

[81] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[82] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Congress on Evolutionary Computation*, vol. 1, 2001, pp. 84–88.

[83] C.-K. Au and H.-F. Leung, "An empirical comparison of CMA-ES in dynamic environments," in *Parallel Problem Solving from Nature*, 2012, vol. 7491, pp. 529–538.

[84] H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.

[85] M. N. Omidvar, Y. Mei, and X. Li, "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms," in *Congress on Evolutionary Computation*, 2014, pp. 1305–1312.

[86] D. Yazdani, J. Branke, M. N. Omidvar, T. T. Nguyen, and X. Yao, "Changing or keeping solutions in dynamic optimization problems with switching costs," in *Genetic and Evolutionary Computation Conference*, 2018, pp. 1095–1102.
[87] Y. Jin, K. Tang, X. Yu, B. Sendhoff, and X. Yao, "A framework for finding robust optimal solutions over time," *Memetic Computing*, vol. 5, no. 01, pp. 3–18, 2013.
[88] C. Bu, W. Luo, and L. Yue, "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 14–33, 2017.
[89] T. T. Nguyen and X. Yao, "Continuous dynamic constrained optimizationthe challenges," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 769–786, 2012.
[90] J. Zou, Q. Li, S. Yang, H. Bai, and J. Zheng, "A prediction strategy based on center points and knee points for evolutionary dynamic multi-objective optimization," *Applied Soft Computing*, vol. 61, pp. 806–818, 2017.
[91] R. Azzouz, S. Bechikh, L. B. Said, and W. Trabelsi, "Handling time-varying constraints and objectives in dynamic evolutionary multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 39, pp. 222–248, 2018.