A Data-Driven Evolutionary Transfer Optimization for Expensive Problems in Dynamic Environments

Ke Li\textsuperscript{a}, Senior Member, IEEE, Renzhi Chen, and Xin Yao, Fellow, IEEE

Abstract—Many real-world problems are computationally costly and the objective functions evolve over time. Data-driven, a.k.a. surrogate-assisted, evolutionary optimization has been recognized as an effective approach to tackle expensive black-box optimization problems in a static environment whereas it has rarely been studied under dynamic environments. This article proposes a simple yet effective transfer learning framework to empower data-driven evolutionary optimization to solve expensive dynamic optimization problems. Specifically, a hierarchical multioutput Gaussian process is proposed to capture the correlation among data collected from different time steps with a linearly increased number of hyperparameters. Furthermore, an adaptive source task selection along with a bespoke warm starting initialization mechanisms are proposed to better leverage the knowledge extracted from previous optimization processes. By doing so, the data-driven evolutionary optimization can jump start the optimization in the new environment with a very limited computational budget. Experiments on synthetic benchmark test problems and a real-world case study demonstrate the effectiveness of our proposed algorithm in comparison with nine state-of-the-art peer algorithms.

Index Terms—Data-driven evolutionary optimization, dynamic optimization, kernel methods, multioutput Gaussian processes (GPs), transfer optimization.

I. INTRODUCTION

REAL-WORLD black-box optimization problems pose unique challenges. Their complexity arises from their nonconvex, multimodal nature and the presence of constraints and multiple objectives. Besides, they are also subject to various uncertainties [1]. A specific type of these problems is the dynamic optimization problem (DOP). In a DOP, objective functions can undergo abrupt changes over time, leading to constantly evolving optima. This type of optimization problem is commonly encountered in many real-world scenarios. Take modern software systems as an example. These systems are typically equipped with a range of control features, such as CPU cap, thread pool size, and cache size. These features play a pivotal role in determining the system’s functional and nonfunctional properties. But what adds a layer of complexity is the need for these systems to cater to multiple users simultaneously. This necessitates on-the-fly dynamic configuration of these control features. Such configurations are largely dependent on variables in the runtime environments, such as bandwidth and response time [2].

Partially due to the population-based property and the self-adaptivity [3], evolutionary algorithms (EAs) have been widely recognized as an effective tool for DOPs [4], [5], [6], [7], [8]. Most EA developments implicitly assume that the objective function evaluations (FEs) are computationally cheap. Thus, it is common to see an EA routine costs tens or hundreds thousands or even more FEs in a single run. This assumption is problematic in real-world optimization scenarios. Here, FEs can be physically, computationally, or economically expensive. These are known as expensive black-box optimization problems. The challenge escalates when dealing with environments that change dynamically over time and are filled with uncertainties.

Data-driven optimization methods, including surrogate-assisted EAs (SAEAs) [9], efficient global optimization (EGO) [10], and Bayesian optimization (BO) [11], effectively handle expensive black-box optimization problems. These methods use data from limited, expensive FEs to create surrogate models of the physical objective function. Gaussian process (GP) models [12] are common examples of such surrogate models. They are also referred to as Kriging [13] or design and analysis of computer experiments stochastic process models [14]. However, existing data-driven optimization research primarily focuses on continuous space with static objective functions. DOPs receive far less attention, with few works [15], [16], [17], addressing them. This disparity can be explained by two main reasons.

1) The ultimate goal of dynamic optimization extends beyond simply approximating the global optimum. It aims to track the evolving optimum over time. Due to dynamic environments and rapidly changing problem landscapes, restarting the optimization routine from scratch, known as a “cold start,” is often impractical, ineffective, or inefficient when the environment changes [15].

2) While most data-driven optimization methods can adapt to changing search landscapes, it is unproductive to...
ignore these changes and let the optimization routine run unmodified. The time-varying nature of DOPs makes data collected in previous time steps stale, or even harmful, for current model building [15], [16], [17], [18]. Moreover, excessive data can make training and inference for a GP model computationally expensive.

DOPs inherently suggest a correlation between problem landscapes across successive time steps. This is because many applications experience gradual changes from one environment to the next. As such, it is logical to selectively utilize data from previous searches to “warm start” the optimization task. Warm starting a BO has been explored within the machine learning community. This concept has found applications in areas like hyperparameter optimization [19] and neural architecture search [20]. However, these implementations are usually “one-off” processes. Their aim is to use existing data to “jump start” the problem-solving process. This approach aligns with our findings.

We recently developed a simple BO variant, TBO [18], which employs a multioutput GP (MOGP) [21]. This model exploits correlations between different time steps, instead of modeling each time step individually. The proof-of-concept results demonstrate TBO’s effectiveness in solving DOPs compared to peer algorithms, especially under limited computational budgets. However, it is far from mature in view of the following three drawbacks.

1) The number of hyperparameters in a conventional MOGP is cubic in the number of tasks. In dynamic optimization, collecting abundant data at each time step is unlikely due to costly FEs. Consequently, the growth in data collected may not keep pace with the increasing number of hyperparameters, leading to underfitting over time.

2) TBO uses a decay mechanism to discard some previously collected data, especially if they become less relevant to the current time step. This strategy mitigates the computational cost of MOGP. However, this decay mechanism is integrated as a hyperparameter of MOGP, raising the risk of overfitting with limited training data. It also fails to provide a controlled method for selecting the most relevant data source for MOGP model training.

3) A significant advantage of TBO is that it requires a limited number of initial samples to jump-start a new optimization task using MOGP. However, since this initial sampling is random, it can easily introduce bias. Furthermore, it does not fully leverage the knowledge gained from previous optimization tasks.

Bearing these considerations in mind, we propose a well-founded data-driven evolutionary transfer optimization framework, called DETO, for expensive DOPs. This article’s main contributions include the following.

1) To mitigate the underfitting problem of the conventional MOGP, we propose a hierarchical MOGP (HMOGP) that replaces the block matrices of the task related hyperparameters of the MOGP with a series of masked block matrices.

2) To reduce the risk of overfitting brought about by the augmented hyperparameters, we develop an adaptive source data selection mechanism. It proactively selects the most appropriate source task for MOGP model training.

3) To capitalize on insights from prior optimization tasks, we propose a custom initialization strategy. This approach uses the local optima collected from the estimated problem landscapes of previously completed optimization tasks, each represented by a separate GP model.

4) To tackle the nonconvex nature of optimizing the acquisition function, we develop a simple yet effective EA with local search. It helps identify the next point of merit.

5) Extensive experiments on synthetic benchmark problems with adjustable dynamic characteristics and a real-world DOP fully validate the efficacy of our proposed DETO. We compare it to nine other peer algorithms to support our findings.

In the remainder of this article, Section II provides some important background knowledge in order to understand this article along with a related literature review. The technical details of our proposed DETO are delineated in Section III and its performance is validated through a series of computational studies given in Section V. At the end, Section VI concludes this article and sheds some light on future directions.

II. PRELIMINARIES

This section begins by introducing some foundational concepts. These include the formal definition of the DOP that we are focusing on in this article, and the core principle of a basic data-driven optimization method that uses a GP model. Following this, we provide a pragmatic literature review on both data-driven methods (EGO and BO variants in particular) and EAs for solving DOPs.

A. Problem Definition

The DOP considered in this article is defined as

\[
\begin{align*}
\max_{\mathbf{x} \in \Omega} f(\mathbf{x}, t) = (x_1, \ldots, x_n) \in \mathbb{R}^n \subset \mathbb{R}^n = [x^L_1, x^U_1] \quad \subset \mathbb{R}^n = [x^L_t, x^U_t] \quad \subset \mathbb{R}^n
\end{align*}
\]

where \( \mathbf{x} = (x_1, \ldots, x_n)^T \) is a decision vector, \( \Omega = [x^L_1, x^U_1] \subset \mathbb{R}^n \) is the search space, \( t \in \{1, \ldots, T\} \) is a discrete time step and \( T > 1 \) is the total number of time steps. Note that the objective function is assumed to be computationally expensive and it changes over time so as its landscapes and the local/global optima. In this case, the ultimate goal of computationally expensive black-box optimization in dynamic environments is to track the time-varying optimum across \( T \) time steps under a strictly limited amount of overall computational budget.

**Theorem 1 (Karush–Kuhn–Tucker (KKT) Conditions [22]):** If \( \mathbf{x}^* \) is a local optimum of an objective function \( f(\mathbf{x}): \mathbb{R}^n \to \mathbb{R} \) with \( m \) constraints, \( g_i(\mathbf{x}) \leq 0 \) for \( i = 1, \ldots, m \), and the set of vectors \( \{g_i(\mathbf{x}^*)\} \) is linearly independent, then there exists a vector \( \mu = (\mu_1, \ldots, \mu_m)^T \) \( \in \mathbb{R}^m \) such that:

\[
\sum_{i=1}^{m} \mu_i g_i(\mathbf{x}^*) = 0
\]
Algorithm 1: Pseudocode for a Vanilla Data-Driven Optimization Based on GP Model

**Input:** Related hyperparameter settings.

**Output:** The best solution found so far.

1. Use an experimental design method to sample a set of initial solutions \( X \leftarrow \{x_i\}_{i=1}^{N_I} \) from \( \Omega \) and evaluate their objective function values \( Y \leftarrow \{f(x_i)\}_{i=1}^{N_I} \). Set the initial training dataset \( \mathcal{D} \leftarrow \{(x_i, f(x_i))\}_{i=1}^{N_I} \);

2. **while** termination criteria is not met **do**
   3. Build a GP model based on \( \mathcal{D} \);
   4. Optimize an acquisition function to obtain a candidate solution \( \hat{x}^* \);
   5. Evaluate the objective function values of \( \hat{x}^* \) and update \( \mathcal{D} \leftarrow \mathcal{D} \cup \{(\hat{x}^*, f(\hat{x}^*))\} \);

6. return \( \arg\min_{x \in \mathcal{D}} f(x) \)

Let \( \mathbb{R}^m \), such that
\[
\nabla f(x^*) + \sum_{j=1}^m \mu_j \nabla g_j(x^*) = 0 \\
\mu_j g_j(x^*) \mid_{j=1}^m = 0
\]  
where \( \mu_j \geq 0 \) \( \forall j \in \{1, \ldots, m\} \).

**Remark 1:** The objective and constraint functions are assumed to be continuously differentiable in the KKT conditions.

**Remark 2:** The objective function considered in this article does not include constraints, thus we ignore the \( \sum_{j=1}^m \mu_j \nabla g_j (x^*) \) part of (2) in the latter derivations.

B. Vanilla Data-Driven Optimization Based on GP Model

Data-driven optimization based on GP model represents a category of sequential model-based optimization methods. They are particularly effective for solving black-box optimization problems that have computationally expensive objective functions. As shown in line 1 of Algorithm 1, this process begins with solutions sampled uniformly using a space-filling experimental design, such as Latin hypercube sampling. Following this, the process iteratively updates its next sample until the computational resources have been fully utilized. Specifically, its iteration has two primary components: 1) a GP-based surrogate model approximates the expensive objective function and 2) an infill criterion, driven by the optimization of an acquisition function, determines the next sample point for evaluation by the costly objective function. Let us briefly discuss these two components as follows.

1) Surrogate Model: This article considers using GP regression (GPR) model to serve the surrogate modeling (line 3 of Algorithm 1). Given a set of training data \( \mathcal{D} = \{(x_i, f(x_i))\}_{i=1}^{N_D} \), the GPR model considered in this aim is designed to learn a noise-free latent function with a constant 0 mean. For each testing input vector \( z^* \in \Omega \), the mean and variance of the target \( f(z^*) \) are predicted as
\[
\begin{align*}
\mu(z^*) &= k^* \top K^{-1} f \\
\sigma(z^*) &= k(k^*, z^*) - k^* \top K^{-1} k^*
\end{align*}
\]
where \( X = (x^1, \ldots, x^N)^\top \) and \( f = (f(x^1), \ldots, f(x^N))^\top \), \( k^* = (k(x^1, z^*), \ldots, k(x^N, z^*))^\top \) is the covariance vector between \( X \) and \( z^* \), and \( K \) is the covariance matrix of \( X \). In particular, a covariance function, also known as a kernel function, is used to measure the similarity between a pair of two data points \( x \) and \( x' \in \Omega \). Here, we use the radial basis function (RBF) kernel in this article and it is defined as
\[
k(x, x') = \gamma \exp \left(-\frac{\|x - x'\|}{\ell}\right)^2
\]
with two hyperparameters \( \gamma \) and length scale \( \ell \). The predicted mean \( \mu(z^*) \) is directly used as the prediction of \( f(z^*) \), and the predicted variance \( \sigma(z^*) \) quantifies the uncertainty. As recommended in [12], the hyperparameters associated with the mean and covariance functions are learned by maximizing the log marginal likelihood function defined as
\[
\log p(f|X) = -\frac{1}{2} f^\top K^{-1} f - \frac{1}{2} \log |K| - \frac{N}{2} \log 2\pi.
\]

2) Infill Criterion: The search for the next point of merit, \( \hat{x}^* \), is driven by an acquisition function rather than by optimizing the surrogate objective function. This approach balances the exploitation of the predicted minimum and the exploration of model uncertainty
\[
\hat{x}^* = \arg\max_{x \in \Omega} f^{acquire}(x)
\]
where \( f^{acquire}(x) = \mu(x) + \sigma(x) \) represents the commonly used upper confidence bound (UCB) in this article [23]. Here, \( \omega > 0 \) is a parameter that manages the tradeoff between exploration and exploitation. The optimization of (6) can be executed by an EA or a gradient descent method (refer to lines 4 and 5 of Algorithm 1).

C. Related Works

EAs, renowned for their population-based and self-adaptive features, are highly effective for DOPs. Over recent decades, numerous studies have explored the use of EAs for solving DOPs. For a comprehensive literature review, we suggest excellent survey papers, such as [4], [5], [6], [7], and [8]. However, because EAs typically demand a substantial number of FEs to achieve a satisfactory solution, they may not be directly applicable or computationally feasible for expensive DOPs. An interesting recent development is an EA proposed by Luo et al. [24], designed for data-driven dynamic optimization. This approach begins by storing elite solutions found in previous time steps, which are then clustered to build surrogate models. Following this, two variants of particle swarm optimization with memory are proposed to solve DOPs. Despite its utilization of surrogate models for objective FE, this approach remains unfeasible for expensive DOPs in our context, as it demands \( 10000 \times n \) FEs at each time step.

On the other hand, while data-driven optimization is a well-established field for expensive optimization [9], research specifically targeting expensive dynamic optimization from a data-driven perspective remains sparse, as outlined in Section I. Kushner [25] made an early attempt in this area, proposing a stochastic processes model to represent a
dynamically changing function. However, the focus on 1-D problems in [25] limits its practical applicability. The field then remained quiet until Morales-Enciso and Branke’s work, which proposed eight strategies to tackle expensive DOPs within the EGO framework [15]. The first four strategies, including random sampling, evolutionary mutation, restart, and ignorance, were simple but limited in their ability to handle complex dynamics. The remaining strategies sought to reuse and transfer previous knowledge to the ongoing problem-solving process but relied on prior knowledge or assumptions about the problem characteristics.

More recent work includes that of Nyikosa et al. [16], who introduced an adaptive BO method. This approach employs a product kernel to augment a spatial-temporal covariance function for managing expensive DOPs, utilizing a time-related length-scale parameter as a forgetting factor. Richter et al. [17] proposed two approaches to adapt BO for DOPs: 1) the window approach, which uses only the most recent observations for model training and 2) the time-as-covariate approach, which incorporates time into the covariance function. Finally, our recent work [18] introduced a BO variant, offering proof-of-concept results for the use of a MOGP to exploit correlations between current and past optimization tasks, using a sliding window strategy to focus on the most recent time steps and manage computational costs.

A related line of research concerns the “warm starting” of data-driven optimization, where additional information beyond FE is employed. This approach has seen significant attention in the BO community. Many researchers have championed the use of meta-learning [26], [27] and acquisition functions [28], [29]. Others have suggested knowledge transfer techniques, such as weighted aggregation of surrogate models across multiple tasks [19], [30], [31], and multitasking strategies [32]. These methods have proven effective in hyperparameter optimization. However, they typically rely on large volumes of historical training data to discern the traits of a task class. This requirement may be hard to fulfill in expensive dynamic optimization considered in this article.

The last related line of research is online optimization, which focuses on real-time learning and decision making [33]. Unlike the DOP discussed in this article, online optimization considers \( f(x, t) \) would change with each FE and is highly correlated. Furthermore, online optimization often presumes certain regularity, such as an unknown reward function within a compact and convex subset or Lipschitz continuity. For instance, Bogunovic et al. [34] introduced two variants of GP-UCB algorithms that utilize spatial and temporal correlations among time-varying reward functions. Liu et al. [35] considered online simulation optimization with streaming input data and suggested an online adaptation of the Bayesian risk optimization framework. Letham and Bakshy [36] proposed using multitask BO to calibrate online machine learning systems through online-offline experimentation.

### III. PROPOSED ALGORITHM

The pseudo code for DETO is in Algorithm 2. While it largely mirrors the structure of Algorithm 1, there are notable differences. These distinct elements (highlighted in blue color) include the following:

1. The use of a HMOGP with fewer hyperparameters, serving as the surrogate model that capitalizes on information gathered from prior time steps.
2. An adaptive source data selection mechanism designed to lessen the risk of overfitting.
3. A custom initialization process that effectively “warm starts” the new optimization task.
4. A simple yet potent EA, combined with a local search, to find the next point for executing the expensive FE.

#### A. Surrogate Model Building

Given a DOP, we denote \( \mathcal{D} = \{ D^1, \ldots, D^T \} \) as a set of observations collected so far where \( D^t = \{ (x_i, t), f((x_i, t)) \}_{i=1}^{N_t}, t \in \{ 1, \ldots, T \}, T \geq 1 \) and \( N_t > 1 \) is the number of observations evaluated at the \( t \)th time step. This article proposes to apply a MOGP model to exploit the correlations between the observations collected at different time steps. Comparing to the conventional GPR model, the MOGP model is featured with the multivariate covariance function, also known as linear model of coregionalization (LMC) [21]. It is defined as

\[
k((x, t), (x', t')) = \sum_{i=1}^{T} [A_i]_t \cdot k_i(x, x')
\]

### Algorithm 2: Pseudocode of DETO

**Input:** Related hyperparameter settings.

**Output:** The optima found at all time steps \( X^* \).

1. \( t \leftarrow 1, \mathcal{D} \leftarrow \emptyset, X^* \leftarrow \emptyset; \)
2. Use an experimental design method to sample a set of initial solutions \( X^1 \leftarrow \{ (x_i, t) \}_{i=1}^{N_t} \) from \( \Omega \) and evaluate their objective function values \( \mathcal{Y} \leftarrow \{ f((x_i, t), t) \}_{i=1}^{N_t} \). Set the initial training dataset \( \mathcal{D}^t \leftarrow \{ (x_i, t), f((x_i, t), t) \}_{i=1}^{N_t}, N_t \leftarrow N_t + 1 \); 
3. if termination criterion is not met then
   while \( N_t < N_{FE} \) do
      Build a MOGP model based on \( \mathcal{A} \cup \mathcal{D}^t \) with a low-rank approximation of covariance matrix;
      Optimize an acquisition function to obtain a candidate solution \( \hat{x}^*, t \);
      Evaluate the objective function values of \( \hat{x}^*, t \) and set \( \mathcal{D}^t \leftarrow \mathcal{D}^t \cup \{ (\hat{x}^*, t), f((\hat{x}^*, t), t) \} \);
      \( N_t \leftarrow N_t + 1 \); 
      \( \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}^t \); 
      \( X^* \leftarrow X^* \bigcup \{ \arg\min_{(x, t) \in \mathcal{D}^t} f((x, t), t) \} \); 
      \( t \leftarrow t + 1 \);
   Use the adaptive source data selection to pick up \( k \) related datasets from \( \mathcal{D} \); 
   Use the warm starting initialization to generate \( k \) augmented datasets to construct \( \mathcal{A} \); 
   \( N_t \leftarrow 0, \mathcal{D}^t \leftarrow \emptyset \);
15. return \( X^* \)
where \((x, t)\) is a candidate solution sampled at the current time step \(t\) while \((x', t')\) is an observation at the \(t'\)th time step \((1 \leq t' \leq T)\) chosen from \(\mathcal{D}\). \(k_t(\cdot, \cdot)\) is the covariance function at the \(t\)th time step \((1 \leq t \leq T)\) as in the conventional GPR. \(A_t = B_tB_t^T \in \mathbb{R}^{T \times T}\) is used to characterize the correlation between different time steps where

\[
B_t = \begin{bmatrix}
    b_{1,1} & b_{1,2} & \cdots & b_{1,r} \\
    b_{2,1} & b_{2,2} & \cdots & b_{2,r} \\
    \vdots & \vdots & \ddots & \vdots \\
    b_{T,1} & b_{T,2} & \cdots & b_{T,r}
\end{bmatrix}
\]  

(8)

where \(b_{j,k} (1 \leq j \leq T \text{ and } 1 \leq k \leq r)\) is the coefficient (as the hyperparameter) that aggregates the covariance between outputs across different time steps. For each testing input vector sampled at the \(t\)th time step \((z^*, t)\), the mean and variance of the target \(f((z^*, t), t)\) are predicted as

\[
\begin{align*}
    \mu((z^*, t)) &= k_t^T K_t^{-1} f_t \\
    \sigma((z^*, t)) &= k((z^*, t), (z^*, t)) - k_t^T K_t^{-1} k_t
\end{align*}
\]  

(9)

where \(f_t\) is a vector of the objective values of all solutions in \(\mathcal{D}\), \(k_t^\top\) is the vector of covariance functions between \((z^*, t)\) and solutions in \(\mathcal{D}\), and \(K_t\) is the covariance matrix of solutions in \(\mathcal{D}\).

From (7) and (8), it is clear that the LMC’s hyperparameters increase cubically as the number of time steps, \(T\), grows. Given that FEs in this study are computationally expensive, we may not gather a substantial amount of data across varying time steps. Consequently, this could potentially lead to underfitting the LMC within our expensive dynamic optimization context. To counteract this risk, we are introducing a HMOGP, which maintains the same covariance function as (7). However, each element of the task correlation matrix \(A_t\) is redefined in the following way

\[
\tilde{A}_t = \begin{cases}
    1, & \text{if } t \geq i, t' \geq i \\
    0, & \text{otherwise.}
\end{cases}
\]  

(10)

Alternatively, we can write \(\tilde{A}_t\) as

\[
\tilde{A}_t = \begin{bmatrix}
    0_{(i-1) \times (i-1)} & 1_{(i-1) \times (T-i+1)} \\
    1_{(T-i+1) \times (i-1)} & 1_{(T-i+1) \times (T-i+1)}
\end{bmatrix}
\]  

(11)

More specifically, we have

\[
\tilde{A}_1 = \begin{bmatrix}
    1 & \cdots & 1 \\
    \vdots & \ddots & \vdots \\
    1 & \cdots & 1
\end{bmatrix}, \quad \tilde{A}_2 = \begin{bmatrix}
    0 & 1 & \cdots & 1 \\
    1 & 1 & \cdots & 1 \\
    \vdots & \vdots & \ddots & \vdots \\
    1 & 1 & \cdots & 1
\end{bmatrix}, \quad \cdots
\]  

(12)

\[
\tilde{A}_T = \begin{bmatrix}
    0 & \cdots & 0 & 1 \\
    \vdots & \ddots & \vdots & \vdots \\
    0 & \cdots & 0 & 1 \\
    1 & \cdots & 1 & 1
\end{bmatrix}
\]

By doing so, the number of hyperparameters of this HMOGP is significantly reduced by two orders of magnitude to be linear to \(T\), comparing against the conventional LMC.

B. Adaptive Source Data Selection Mechanism

In a dynamic optimization process, the evaluated solutions incrementally accumulate over time. Despite this being counter-intuitive, especially given the limited data typically collected in expensive optimization, amassing an “excessive” quantity of data may inadvertently lead to overfitting. We demonstrate this in Fig. 1 using the moving peak function from our empirical study in Section IV. For illustration purposes, we consider \(T = 8\) time steps, with 19 data points per step, sampled uniformly within the moving peak function’s \(x\) range. As evident in Fig. 1, including data from all eight source datasets results in the HMOGP overfitting and generating superfluous local optima. To alleviate this, we suggest an adaptive source data selection mechanism. This mechanism proactively selects the most fitting source data from past collections to construct the HMOGP at each time step, comprising four distinct steps.

Step 1: Build a GPR model \(GP(D^t), t \in \{1, \ldots, T-1\}\), for each of the previous time steps.

Step 2: Store the hyperparameters \(h'^t\) of \(GP(D^t)\) into \(H = \{h'^t\}_{t=1}^{T-1}\).

Step 3: Use the classic \(k\)-means clustering algorithm to divide \(H\) into \(k > 1\) clusters.

Step 4: For each cluster, pick up the hyperparameters closest to its centroid and the corresponding dataset is chosen as a source data.

Remark 3: In this article, we use the hyperparameters of \(GP(D^t)\) as the implicit features to represent the characteristics of the problem landscape of \(f(x, t)\) where \(t \in \{1, \ldots, T-1\}\).

Remark 4: In practice, it is not necessary to build all \(T-1\) GPR models from scratch in steps 1 and 2 each time. Instead, we only need to build the GPR model for the previous time step at the outset of each new optimization task.

Remark 5: In the realms of transfer and multitask learning, the common approach to promote knowledge transfer and mitigate adverse transfer is identifying the most similar tasks [37]. However, in the context of dynamic optimization, the early stages of a new time step often yield insufficient data. This
limitation can potentially introduce unexpected bias if we aim to align previous tasks with the current one. With this in mind, our proposed adaptive source data selection strives to identify the most representative tasks rather than the most similar ones. Consider the examples in Figs. 2 and 3 to better understand this process. Fig. 2 demonstrates that when using the k-means clustering algorithm, the hyperparameters of 10 GPR models (where $\ell$ represents normalized $\ell$ within the range $[0, 1]$) built from distinct source datasets are grouped into three clusters. Looking at the estimated fitness landscapes of corresponding GPR models in Fig. 3, we can see that the datasets $\{D^1, D^3, D^6\}$, $\{D^5, D^8, D^9\}$, and $\{D^2, D^7, D^{10}\}$ are quite similar within each cluster. Meanwhile, these clusters complement those from different clusters, enhancing diversity, and broad representation.

C. Warm Starting Initialization Mechanism

In traditional BO and SAEA, the initialization step typically involves using an experimental design method to randomly sample a set of solutions in a strategic manner. However, in the context of DOPs, where recurrent patterns can emerge across different time steps, valuable knowledge can be gleaned from previous optimization tasks. It is counter-intuitive to disregard such knowledge and start the new problem-solving process from scratch. This is particularly inefficient in the case of expensive optimization scenarios, where computational resources are highly limited at each time step. To address this concern, we introduce a tailored initialization mechanism that capitalizes on the insights gained from related previous tasks identified by the adaptive source task selection. This mechanism serves to warm start the underlying optimization task and consists of two key steps.

Step 1: For each of the $k$ tasks recommended by the adaptive source task selection discussed in Section III-B, identify the set of local optima $L^i$ where $i \in \{1, \ldots, k\}$ w.r.t. the corresponding GPR model.

Step 2: For each task $i \in \{1, \ldots, k\}$, sort the samples in $L^i$ based on the predicted objective function values. Use the following three-step procedure to pick up $1 \leq \tilde{N} < |L^i|$ samples to constitute the augmented dataset $A^i$ (started from an empty set) for the $i$th task.

Step 2.1: Pick up the current best sample and calculate its Euclidean distance(s) w.r.t. all samples in $A^i$.

Step 2.2: If any of the distances is less than a threshold $\epsilon_\ell$, remove this sample from $L^i$ and move to the next sample then go to step 2.1. Otherwise, this sample is removed from $L^i$ and it is added into $A^i$.

Step 2.3: Repeat steps 2.1 and 2.2 until the size of $A^i$ reaches $\tilde{N}$.

Step 3: $\mathcal{A} = \bigcup_{i=1}^{\tilde{N}} A^i$.

Remark 6: Since the GPR model is continuously differentiable, we can derive the first order derivative of the predicted mean function w.r.t. a solution $x$ as

$$\frac{\partial \hat{g}(x)}{\partial x} = \frac{\partial k^{\mathbf{X}^*}}{\partial x} K^{-1} \mathbf{f}.$$ (13)

Based on the KKT conditions in Theorem 1 and (13), we can derive the saddle points of the corresponding GPR model by identifying the solutions that satisfy $([\partial k^{\mathbf{x}}/\partial x]) K^{-1} \mathbf{f} = 0$. They are thereafter used as the local optima in step 1.

Remark 7: Instead of re-evaluating the objective function values of the selected samples in $A^i$ where $i \in \{1, \ldots, k\}$, we only use the corresponding GPR model to assign a pseudo objective function value to each sample.

Remark 8: In step 2, a naive idea is to pick the top $\kappa$ samples from $L^i$ where $i \in \{1, \ldots, k\}$. However, it can be less representative if the selected local optima are crowded within a small niche. Thus, we try to enforce certain level of diversity by making sure that the inter distance of samples in $A^i$ is at least larger than $\epsilon_\ell = 10^{-2} \times \max_{1 \leq i \leq N} (x_i^l - x_i^r)$.

D. Optimization of Acquisition Function by a Hybrid EA

As discussed in [38], the optimization of UCB is nontrivial given its nonconvex properties. In view of the outstanding performance of differential evolution (DE) [39] for global optimization, here we propose a simple yet effective DE variant with local search to serve a proof-of-concept purpose. Specifically, it consists of the following six steps.

Step 1: Initialize a population of solutions $P = \{x^i\}_{i=1}^{N}$.

Step 2: Use DE to generate a population of offspring $Q$ and let $\overline{P} = P \cup Q$.

Step 3: Pick up the best-$\kappa$ candidates from $\overline{P}$, according to the UCB value, to constitute an archive $A$.

Step 4: For each candidate $\mathbf{x} \in A$ do

Step 4.1: Use stochastic gradient descent (SGD) [40] on $\mathbf{x}$ to obtain $\bar{\mathbf{x}}$, then set $\overline{P} = P \cup (\bar{\mathbf{x}} \setminus \{\mathbf{x}\})$.

Step 4.2: If $\|\bar{\mathbf{x}} - \mathbf{x}\|_2 < \epsilon_\ell$, then set $\kappa = \max(\kappa - 1, 1)$; otherwise, set $\kappa = \min(\kappa + 1, 2N)$.

Step 5: Pick up the best-$N$ solutions from $\overline{P}$ to constitute $P$ according to their UCB values.

Step 6: If the termination criterion is met, then stop and output $(\mathbf{x}^*, \ell) = \arg\max_{\mathbf{x} \in P} \text{acq}(\mathbf{x})$. Otherwise, go to step 2.
against nine peer algorithms. 

Remark 9: An essential attribute of the acquisition function is its multimodal landscape. This DE variant effectively combines the global search capability of DE with the local search characteristics of SGD, striking a harmonious balance between exploration and exploitation.

Remark 10: $\kappa$ is a hyperparameter governs the scale of the local search. In this article, we set $\kappa = 5$ in step 1, while it will be adaptively tuned in step 4.2.

Remark 11: In step 4.2, if $\tilde{x}$ is very similar to $\hat{x}$, i.e., their Euclidean distance (denoted as $\|\cdot\|_2$) is smaller than a threshold $\epsilon_d$, it implies the local search might not be helpful any longer. Therefore, $\kappa$ will be reduced in the next round.

E. Time Complexity Analysis

DETO consists of four algorithmic components delineated in Sections III-A–III-D. We now provide a breakdown of the computational complexity for each component to derive the overall time complexity of DETO. Regarding the surrogate modeling introduced in Section III-A (line 5 of Algorithm 2), its complexity is $O((\sum_{i=1}^{T} |D|^3))$. For the optimization of the acquisition function presented in Section III-D (line 6 of Algorithm 2), its complexity is dominated by DE, i.e., $O(N^2)$, where $N$ is the population size. Since DETO operates sequentially, these two operations are iterated within the while loop for $|D|$ times at each time step. Consequently, the complexity of lines 4 to 8 is $O((\sum_{i=1}^{T} |D|^4))$. In the adaptive source selection in Section III-B (line 12 of Algorithm 2), the training and inference of GPR in step 1 for each time step incur a complexity of $O(|D|^3)$, while the $k$-means clustering complexity is linear to the dataset size. Consequently, the dominant complexity in adaptive source selection stems from the GPR models. For the warm start proposed in Section III-C (line 13 of Algorithm 2), the complexity is linear due to the GPR model’s continuous differentiability. In summary, the time complexity of DETO is $O((\sum_{i=1}^{T} |D|^4))$.

IV. EXPERIMENTAL SETUP

In this section, we introduce the experimental settings for validating the effectiveness of our proposed DETO compared against nine peer algorithms.

A. Benchmark Test Problems

In our experiments, we use the moving peaks benchmark (MPB) [41], the most popular benchmark test suite in the dynamic optimization literature, to constitute the synthetic benchmark problems. MPB generates a landscape that consists of several components. Each component has a peak whose height, width and location change over time with the environment changes. MPB is flexible and can generate scalable functions with a configurable number of components, each of which is able to become the global optimum at the current time step. The mathematical definitions and visual interpretations of the fitness landscapes of MPB can be found in the Appendix A of the supplemental document of this article.

B. Peer Algorithms

In this article, we consider nine peer algorithms in our experiments, including restart BO (RBO), continuous BO (CBO), discounted information via noise sampling (DIN) [15], model-based optimization (MBO) [17], time varying GP bandit optimization (TV-GP-UCB) [34], BO with box mechanism (Box+GP) [42], ranking-weighted GP ensemble (RGPE) [43], boosted hierarchical GP (BHGP) [44], and transfer BO (TBO) [18]. Their differences w.r.t. DETO are clarified in Appendix A of the supplemental document.

C. Performance Metrics

Since the landscape and optimum change over time in DOP, it is not adequate to simply evaluate the quality of the best solution found by an algorithm in performance comparison. Instead, we need to take more information (e.g., how the algorithms track the moving optima) into account. In this article, we consider the following two performance metrics.

1) Offline Error [45]: It measures the average deviation from the global optimum over all FEs

$$\hat{e}_t = \frac{1}{N_{FE}} \sum_{i=1}^{N_{FE}} [f(x^\star, t) - f_i(x^\star, t)]$$

where $N_{FE}$ is the total number of FEs, $f(x^\star, t)$ is the global optimum at the $t$th time step, $t \in \{1, \ldots, T\}$, and
2) A average of the best error before changes [46]: it measures the average accuracy achieved over all time steps

\[ \bar{\epsilon}_t = \frac{1}{T} \sum_{t=1}^{T} f(\hat{x}^*, t) - f(\bar{x}^*, t) \]  

(15)

where \( f(\hat{x}^*, t) \) is the true global optimum at the \( t \)th time step, \( t \in \{1, \ldots, T\} \), and \( f(\bar{x}^*, t) \) is the best solution found at the end of the \( t \)th time step.

D. Statistical Tests

To have a statistical interpretation of the significance of comparison results, we use the following three statistical measures in our empirical study.

1) Wilcoxon Signed-Rank Test [47]: This is a nonparametric statistical test that makes little assumption about the underlying distribution of the data and has been recommended in many empirical studies in the EA community [48]. In particular, the significance level is set to \( p = 0.05 \) in our experiments.

2) Scott-Knott Test [49]: Instead of merely comparing the raw metric values, we apply the Scott-Knott test to rank the performance of different peer techniques over 31 runs on each test problem. In a nutshell, the Scott-Knott test uses a statistical test and effect size to divide the performance of peer algorithms into several clusters. The performance of peer algorithms within the same cluster are statistically equivalent. Note that the clustering process terminates until no split can be made. Finally, each cluster can be assigned a rank according to the mean metric values achieved by the peer algorithms within the cluster. In particular, the smaller the rank is, the better performance of the algorithm achieves.

3) \( A_{12} \) Effect Size [50]: To ensure the resulted differences are not generated from a trivial effect, we apply \( A_{12} \) as the effect size measure to evaluate the probability that one algorithm is better than another. Specifically, given a pair of peer algorithms, \( A_{12} = 0.5 \) means they are equivalent. \( A_{12} > 0.5 \) denotes that one is better for more than 50% of the times. \( 0.56 \leq A_{12} < 0.64 \) indicates a small effect size while \( 0.64 \leq A_{12} < 0.71 \) and \( A_{12} \geq 0.71 \) mean a medium and a large effect size, respectively. Note that both Wilcoxon signed-rank test and \( A_{12} \) effect size are also used in the Scott-Knott test for generating clusters.

E. Parameter Settings

The parameter settings used in our experiments are summarized as follows.

1) Test Problem Settings: We consider different number of variables as \( n \in \{3, 5, 8, 10\} \). The total number of peaks in MPB and MPBG is \( m = 5 \). In our experiments, we consider both small and large changes by setting the control parameters as \( (\bar{h} = 1.0, \bar{s} = 1.0) \) and \( (\bar{h} = 5.0, \bar{s} = 7.0) \) respectively.

2) Computational Budget: In our experiments, we consider \( T = 100 \) time steps in total and each time step is allocated with an extremely limited FE. Specifically, \( N_{FE} = 2 \times (11n - 1) \) at the first time step with \( 11n - 1 \) for initial sampling. As for the follow up time steps, \( N_{FE} = 9n \) of which \( 2n \) FEs are used for the initialization purpose.

V. Experimental Results

Our empirical study conducted in this section aims to answer the following research questions (RQs).

RQ1: How is the performance of DETO compared against the other peer algorithms at different dimensions?

RQ2: What is the effect brought by the significance of change of environments?

RQ3: What is the benefit of the hierarchical version of MOGP versus the conventional LMC?

RQ4: What is the benefit of the adaptive source data selection?

RQ5: What is the benefit of the warm starting initialization mechanism?

RQ6: What is the benefit of the hybrid DE with local search for the optimization of acquisition function?

RQ7: How is the performance of DETO on solving a real-world problem in dynamic environments?

A. Performance Comparisons With Peer Algorithms

In this section, we investigate the performance comparison of our proposed DETO against the other peer algorithms.

1) Methods: To address RQ1 and RQ2, the results are presented from the following five aspects.

1) First, the statistical comparison results on \( \bar{\epsilon}_t \) and \( \bar{\epsilon}_i \) are given in Table I in the Appendix B of the supplemental document.

2) Then, we apply the \( A_{12} \) effect size to further understand the performance difference between DETO and the other peer algorithms on \( \bar{\epsilon}_t \) and \( \bar{\epsilon}_i \). Given the large number of comparisons, we pull \( 9 \times 16 \) results collected from \( A_{12} \) effect size together and calculate the percentage of different effect sizes obtained by a pair of dueling algorithms. These are presented as bar charts in Fig. 4.

3) To have an overall comparison across different benchmark test problems, we apply the Scott-Knott test to rank the performance of different algorithms. To facilitate a better interpretation of many comparison results, we pull \( 9 \times 16 \) comparison results collected from the Scott-Knott test together along with their variances as the bar charts with error bars in Fig. 5.

4) To facilitate the understanding of the convergence rates of different algorithms during the dynamic optimization process, we develop two dedicated metrics as follows.

a) First, we keep a record of the loss w.r.t. the corresponding optimum at a given FE as

\[ L(x, t) = f(x^*, t) - f(\bar{x}^*, t) \]  

(16)

where \( (x^*, t) \) is the best solution found at the corresponding FE of the \( t \)th time step.
b) The other metric evaluates the difference of the budget required by a peer algorithm w.r.t. the best algorithm at each time step. It is calculated as

\[
\rho_c = \frac{1}{T} \sum_{t=1}^{T} \frac{N_{FE}^c}{N_{FE}^*} \tag{17}
\]

where \(N_{FE}^c\) is the number of FEs consumed by the best algorithm at the \(t\)th time step to obtain its best solution \((\hat{x}^*, t)\). \(N_{FE}^*\) is the number of FEs required by one of the other peer algorithms to achieve \(f(\hat{x}^*, t)\). Note that \(N_{FE}^*\) can be up to \(8 \times N_{FE}\) in our experiments no matter whether the corresponding peer algorithm obtains \(f(\hat{x}^*, t)\) or not. In practice, \(\rho_c \geq 1\) and the larger \(\rho_c\) is, the worse the corresponding peer algorithm is.

5) Finally, we develop another metric \(\rho_t\) to investigate the effectiveness brought by the knowledge transfer, i.e., leveraging historical data from the previous optimization practice. It is calculated as

\[
\rho_t = \frac{1}{T} \sum_{t=1}^{T} \frac{N_{FE}^t}{N_{c}^*} \tag{18}
\]

where \(N_{c}^t\) is the number of FEs required by RBO, which completely ignores the historical data but restarts itself from scratch after each change, to obtain its best solution \((x^*_c, t)\) at the \(t\)th time step. \(N_{FE}^t\) is the number of FEs required by one of the other peer algorithms that leverages historical information to a certain extent to achieve \(f(x^*_c, t)\). In practice, the corresponding knowledge transfer approach is regarded as effective when

\[
0 < \rho_t < 1 \quad \text{(the smaller} \quad \rho_t \quad \text{is, the better-knowledge transfer is)}; \quad \text{otherwise it is negative to the underlying baseline optimization routine.}
\]

2) Results: Let us interpret the results according to the methods introduced in Section V-A1.

1) From the comparison results on \(\bar{e}_t\) and \(\tilde{e}_t\) shown in Table I in the Appendix B of the supplemental document, we can see that our proposed DETO is always the best algorithm in all comparisons without any exception. Note that all the better results are of statistical significance according to the Wilcoxon signed-rank test.

2) In view of the overwhelmingly superior performance observed above, it is not surprising to see the stacked bar charts of \(A_{12}\) shown in Fig. 4 demonstrate that all the better comparison results achieved by DETO are always classified to be large.

3) Likewise, based on the previous two comparison results, it is not difficult to infer the outstanding rank achieved by DETO according to the Scott-Knott test as shown in Figs. 5 and 6. In addition, we also notice that RBO, CBO, and Box+GP are the worst-peer algorithms. As for RBO and CBO, their inferior performance is not surprising as ignoring any changes can hardly enable the BO to adapt to the new problem within a limited amount of FEs. Although Box+GP claimed to use a transfer learning approach to leverage the previous knowledge, its idea of shrinking search space is shown to be a mess when tackling the benchmark problems considered in our experiments. As the predecessor of DETO, the performance of TBO is competitive as it is ranked in the second place according to the Scott-Knott test. BHGP and RGPE also achieve a similar performance as TBO.

4) From the selected trajectories of \(L(x, t)\) shown in Fig. 7 while full results are provided in Figs. 2–9 in the Appendix B of the supplemental document, we can see that the performance DETO is close to the other nine peer algorithms at the first time step. This makes sense because there is no previous data available for DETO, whereas we observe a clear jump start since the second time step. This can be attributed to the effectiveness
of leveraging “knowledge” collected from the previous optimization practices. Although the other peer algorithms, except RBO and CBOP, claimed to have a strategy to transfer knowledge from the historic data, they are not as effective as DETO. This might be because these algorithms are not specifically designed for dynamic optimization of which the problem changes over time. Since only a limited amount of data are collected at each time step, they are not adequate to train a good model.

5) As the bar charts of \( \rho_c \) shown in Fig. 8, we can see that DETO is still the best algorithm. However, its superiority is not as overwhelming as those evaluated by the \( \tau_f \) and \( \tau_t \). The \( \rho_c \) obtained by TBO and RGPE are also close to 1.0, which means they can catch up with DETO when being allocated with reasonably more FEs. It is interesting to note that RBO is competitive where it even converges faster than some BO variants, especially BoX+GP, the worst algorithm in all comparisons.

6) The bar charts of \( \rho_t \) shown in Fig. 9 demonstrate the effectiveness of the knowledge transfer approaches used in DETO, TBO, and RGPE. In particular, DETO can be one time faster than RBO to achieve its comparable result. On the other hand, the performance of CBOP is worse than RBO. This implies that simply feeding all historical data for surrogate modeling can be detrimental to the data-driven optimization. In addition, the knowledge transfer approaches used in DIN, MBO, and BHGP can be negative when handling DOPs with large changes.

B. Ablation Study

In this section, we investigate the effectiveness of the four building blocks of DETO as introduced in Sections III-A–III-D, respectively.

1) Methods: To address RQ3 to RQ4, we design the following four variants of DETO in this ablation study.

1) To address the RQ3, we develop a variant, denoted as DETO-v1, that replaces the HMOGP with the conventional LMC in the surrogate modeling step.

2) To address the RQ4, we first investigate the sensitivity of \( k \) by considering three different settings, i.e., \( k \in \{3, 5, 7\} \), while \( k = 5 \) is the default setting in DETO. In addition, we develop the following three variants with different source selection mechanisms.

   a) DETO-v2: It selects the \( k \) most recent sources, i.e., \( \{D_t\}_{i=t-k}^{t-1} \), where \( t > k \), to build the HMOGP.

   b) DETO-v3: It selects \( k \) sources whose hyperparameters of the corresponding GPR models closest to \( h_t \). In other words, the selected source data are supposed to be most similar to the current time step.

   c) DETO-v4: It randomly selects \( k \) sources to serve the surrogate modeling purpose.

3) To address the RQ5, we develop another variant DETO-v5 that replaces the warm starting initialization mechanism by a random initialization.

4) To address the RQ6, we developed two variants DETO-v6 and DETO-v7 of which the hybrid DE with local search is replaced by a vanilla DE and a vanilla SGD, respectively.

2) Results: Let us interpret the results according to the methods introduced in Section V-B1.

1) From the comparison results of \( \bar{c}_f \) and \( \bar{c}_t \) shown in Table II in the Appendix C of the supplemental document along with the 100% large A12 effect size shown in Fig. 10, we can see that a significant performance degradation of DETO when replacing the HMOGP with the conventional LMC. This is counter-intuitive at the
first glance. Because the HMOGP can be treated as a low-rank approximation of the LMC with a significantly reduced order of hyperparameters while the LMC was supposed to be more expressive and have a more powerful modeling capability than the HMOGP. We attribute the inferior performance of DETO-v1 to the underfitting problem discussed in Section III-A. Let us use an illustrative example of a MPB problem with $n = 1$ shown in Fig. 11 to elaborate this issue. As shown in this figure, both the LMC and the HMOGP work well when $t = 2$, i.e., the surrogate model is built by two data sets. However, the LMC becomes clearly underfit when $t = 5$ with a surprisingly high-uncertainty estimation. It is even worse when $t = 10$. As discussed in Section III-A, the hyperparameters of the LMC surges to 1000 when $t = 10$ whereas the number of training data is way smaller. In contrast, the HMOGP performs consistently well even when $t = 10$ as its number of hyperparameters is merely 10, which is manageable.

2) As the comparison results shown in Fig. 12, we can see that $k = 5$ is the best choice. Moreover, it is interesting to note that using a smaller number of sources ($k = 3$) is even better than considering many sources ($k = 7$). This may suggest that HMOGP can even become overfitting when taking into account a large volume of data. As the comparison results of $\bar{\epsilon}_f$ and $\bar{\epsilon}_t$ shown in Table III in the Appendix C of the supplemental document along with the $A_{12}$ effect size shown in Fig. 11, we confirm that our proposed algorithm benefits from the adaptive source data selection mechanism. It is counter-intuitive to see that DETO-v3, which picks up the most similar $k$ sources, is the worst variant. It is even outperformed by DETO-v4 which randomly selects $k$ sources for surrogate modeling. This can be explained as the data collected during the initialization of each time step is not sufficient to build a reasonably good surrogate model. Thus, choosing the similar source tasks can be misleading accordingly. It is interesting to note that the performance of DETO is close to that of DETO-v2 when $\bar{h} = 1.0$ and $\bar{s} = 1.0$, i.e., the dynamic problems with a small change. This can be explained as the landscape of such problem is close to its recent time steps, it makes sense to leverage the data collected from therein.

3) From the comparison results of $\bar{\epsilon}_f$ and $\bar{\epsilon}_t$ shown in Table IV in the Appendix C of the supplemental document along with the $A_{12}$ effect size shown in Fig. 14, we can see the effectiveness of the warm starting initialization mechanism introduced in Section III-C. However, it is also worth noting that the overwhelmingly better performance is diminishing when $n = 10$. This can be explained as the ineffectiveness of the surrogate modeling in a high-dimensional scenario with a strictly limited amount of training data. In this case, the
local optima estimated from a less reliable model can be misleading.

4) From the statistical comparison results shown in Table V in the Appendix C of the supplemental document along with the $A_{12}$ effect size shown in Fig. 15, we can see that our proposed hybrid DE is the most effective optimizer for the acquisition function. Due to the nonconvex property, neither vanilla DE nor SGD works well as both of them have a risk of being stuck at local optima.

C. Real-World Dynamic Optimization

As an additional investigation, we validate the performance of DETO and the other nine peer algorithms on a real-world DOP of which the unknown objective function evolves over time. We consider here an automated machine learning task for handwritten digit recognition of the famous MNIST dataset [51]. As an illustrative example shown in Fig. 16, the digits are periodically rotated with time. Therefore, the hyperparameters associated with the baseline classifier, a multilayer perceptron (MLP) in particular, need to be adjusted accordingly. The experimental settings can be found in the Appendix D of the supplemental document.

From the comparison results shown in Fig. 17, it is clear to see that our proposed DETO consistently outperforms the other peer algorithms on both $\bar{\epsilon}_f$ and $\bar{\epsilon}_t$ metrics. It is interesting to note that RBO is the second best algorithm on the $\bar{\epsilon}_f$ metric whereas it becomes the second worst one when considering the $\bar{\epsilon}_t$ metric. This observation implies that the transfer learning approaches used in other peer algorithms might lead to negative effect at the early stage of the hyperparameter optimization at each time step. Whereas a simple restarting from scratch is more robust at the early stage while its performance might be stagnated afterward.

D. Further Studies

From the experimental results discussed in the previous sections, we have witnessed the outstanding performance of our proposed DETO against nine peer algorithms. In this section, we deliberately design the following four experiments to further study the boundary of DETO.

1) As introduced in Section IV-E, the computational budget in our experiments is set as an extremely limited number of FEs. However, in some real-world application scenarios, the computational time, e.g., measured as the CPU wall clock time, is instead used as the capping threshold for an optimization algorithm. To mimic such scenario, we set the computational budget at each time step to be a fixed CPU wall clock time and repeat the experiments designed for RQ1. From the statistical comparison results on $\bar{\epsilon}_f$ and $\bar{\epsilon}_t$ shown in Table VI in Appendix E of the supplemental document and the $A_{12}$ effect size shown in Fig. 18, we can see that DETO is still the best algorithm comparing to the other peer algorithms. Moreover, it is surprising to note that RBO, which is not competitive in the previous experiments, becomes much more competitive. This can be explained as RBO does not need to take any historical data into account when training the GPR model. However, as shown in Fig. 19, RBO costs nearly twice more FEs than DETO.

2) Another issue related to the computational budget is the larger amount of FEs allocated to the first time step as in Section IV-E. Here, we design another experiment where $N_{FE}$ is constantly set to $9n$ for all time steps.
with $2n$ FEs for the initialization purpose. The statistical comparison results on $\tilde{\epsilon}_f$ and $\bar{\epsilon}_f$ shown in Table VII in Appendix E of the supplemental document and the $A_{12}$ effect size shown in Fig. 20, we can see an observable performance degradation experienced by DETO, although it is still the most competitive algorithm. This can be attributed to the cold start at the first time step. Since there is no prior knowledge to leverage, the very limited FEs can lead to an insufficient exploration for the optimization at the first time step. Even worse, such deficit will be propagated to the follow-up time steps. On the other hand, it is interesting to note that RBO becomes competitive under this setting. This can be explained as RBO does not need to leverage any historical information but just restart every time. Therefore, it is not influenced by the poor optimization results in the previous time steps.

3) As discussed in [52], the DOP test instances generated by the MPB are relatively simple due to their blunt peak summits. Here, we consider the generalized MPB (GMPB), used in the IEEE CEC 2022 Competition on Dynamic Optimization, to constitute the synthetic benchmark problems. From the statistical comparison results on $\tilde{\epsilon}_f$ and $\bar{\epsilon}_f$ shown in Table VIII in Appendix E of the supplemental document and the $A_{12}$ effect size shown in Fig. 21(a), it is clear to see that all algorithms become struggling including DETO. To understand the underlying reason, we design a 2-D problem instance with an irregular and rough landscape similar to those generated by GMPB. From the contour lines shown in Fig. 22(b), it is clear to see the vanilla GPR model cannot fit such landscape. This can be attributed to the RBF kernel function defined in (4), of which the length scale $\ell$ is a constant for each input dimension. It makes the GPR model less expressive. To remedy this issue, here we use the automatic relevance determination (ARD) kernel [12] to replace the RBF kernel as follows:

$$k(\mathbf{x}, \mathbf{x'}) = \prod_{i=1}^{n} \gamma_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{x'}\|^2}{\ell_i}\right)^2$$

where each input dimension $i \in \{1, \ldots, n\}$ is associated with a dedicated $\gamma_i$ and $\ell_i$. By using this ARD kernel in the GPR models of all peer algorithms, we can see that DETO becomes competitive again comparing to the other nine peer algorithms as the statistical comparison results on $\tilde{\epsilon}_f$ and $\bar{\epsilon}_f$ shown in Table IX in Appendix E of the supplemental document and the $A_{12}$ effect size shown in Fig. 21(b). As shown in Fig. 22(c), we can see that the GPR model becomes more capable to fit the irregular landscape after using the ARD kernel.

4) Finally, the benchmark problems considered in the previous experiments only have five peaks in total. To investigate the impact of involving more peaks in the landscapes of the DOP test instances, we conduct another set of experiments by setting $m \in \{20, 50, 100\}$, respectively. From the statistical comparison results on $\tilde{\epsilon}_f$ and $\bar{\epsilon}_f$ shown in Tables X to XII in Appendix E of the supplemental document and the $A_{12}$ effect size shown in Fig. 23, we find that all algorithms experienced a significant performance degradation with the increase of $m$. As the contour plots shown in Fig. 24(b), we can see that the GPR model is completely underfitting when having 100 local peaks, which confuse the GPR model as noises in surrogate modeling. To mitigate this issue, we apply the simple strategy proposed in [53] that uses the $k$-means algorithm to divide $\mathcal{D}$ into different clusters. Then, we build a local model for each cluster. As the statistical comparison results on $\tilde{\epsilon}_f$ and $\bar{\epsilon}_f$ shown in Tables XIII to XV in Appendix E of the
supplemental document, we can see that the performance of DETO have been improved by using this simple strategy in surrogate modeling. This is also reflected by the improved surrogate modeling performance shown in Fig. 24(c). Nevertheless, it is still very challenging when tackling problems with too many peaks, i.e., \( m = 100 \).

VI. CONCLUSION

This article presents an efficient, data-driven transfer optimization framework that enhances data-driven evolutionary optimization in addressing expensive black-box optimization problems in dynamic environments. The cornerstone of our proposed DETO is a HMOGP surrogate model. This model effectively transfers knowledge across datasets gathered from distinct time steps, optimizing cost-efficiency. Moreover, an adaptive source task selection mechanism, coupled with a bespoke warm-start initialization strategy, further augments this knowledge transfer. Looking ahead, there are several potential extensions to our proposed DETO framework.

1) A significant volume of work exists on transfer learning [54]. Although any widespread transfer learning approach can be integrated into our DETO framework, there is a need for additional focus to prevent underfitting problems caused by the extremely constrained computational budget in DOPs, as discussed in Section V-B.

2) As per our discussion in Section V-D, there is value in developing more sophisticated surrogate models for data-driven evolutionary optimization. They should be capable of handling uneven and rough landscapes and problems with many local optima, known as multimodal problems.

3) Meta-learning have been an emerging approach to learn new concepts and skills fast with a few training examples by a learning to learn paradigm [55]. It is promising to unleash the potential of meta-learning to tackle expensive DOPs under our proposed DETO framework. This will be helpful to handle the cold start problem discussed in Section V-D.

4) Finally, existing transfer optimization approaches [56] are not easily scalable for many tasks. Real-world systems, however, are typically designed to handle numerous (possibly infinite) problems over their lifetimes. An intriguing direction for future work involves developing a life-long optimization paradigm. Such a paradigm would continuously acquire knowledge from ongoing optimization practices and autonomously select useful information for new, unseen tasks in an open-ended environment.

REFERENCES

[1] S. Yang and X. Yao, *Evolutionary Computation for Dynamic Optimization Problems*. Heidelberg, Germany: Springer, 2013.
[2] T. Chen, K. Li, R. Balasoon, and X. Yao, “FEMOSAA: Feature-guided and knee-driven multi-objective optimization for self-adaptive software,” *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 2, pp. 1–50, 2018.
[3] D. B. Fogel and Z. Michalewicz, *How to Solve It: Modern Heuristics*. Heidelberg, Germany: Springer-Verlag, 2004.
[4] Y. Jin and J. Branke, “Evolutionary optimization in uncertain environments—A survey,” *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.
[5] T. T. Nguyen, S. Yang, and J. Branke, “Evolutionary dynamic optimization: A survey of the state of the art,” *Swarm Evol. Comput.*, vol. 6, pp. 1–24, Oct. 2012.
[6] M. Mavrovouniotis, C. Li, and S. Yang, “A survey of swarm intelligence for dynamic optimization: Algorithms and applications,” *Swarm Evol. Comput.*, vol. 33, pp. 1–17, Apr. 2017.
[7] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part A,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 609–629, Aug. 2021.
[8] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A survey of evolutionary continuous dynamic optimization over two decades—Part B,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 630–650, Aug. 2021.
[9] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, “Data-driven evolutionary optimization: An overview and case studies,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 442–458, Jun. 2019.
[10] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *J. Global Optim.*, vol. 13, no. 4, pp. 455–492, 1998.
[11] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.
[12] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[13] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. New York, NY, USA: Springer-Verlag, 1999.
[14] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*. New York, NY, USA: Springer-Verlag, 2018.
Renzhi Chen received the B.Sc. degree in computer science from the National University of Defence Technology, Changsha, Hunan, China, in 2012, and the Ph.D. degree in computer science from the University of Birmingham, Birmingham, U.K., in 2018. His current research interests include evolutionary algorithms for solving dynamic multiobjective optimization problems.

Xin Yao (Fellow, IEEE) received the B.Sc. degree from the University of Science and Technology of China (USTC), Langfang, Hefei, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technologies, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990. He is a Chair Professor of Computer Science with the Southern University of Science and Technology, Shenzhen, China, and a part-time Professor of Computer Science with the University of Birmingham, Birmingham, U.K. His current research interests include evolutionary computation, machine learning, and their real world applications, especially to software engineering. Dr. Yao was a recipient of the prestigious Royal Society Wolfson Research Merit Award in 2012; the IEEE Computational Intelligence Society (CIS) Evolutionary Computation Pioneer Award in 2013; and the IEEE Frank Rosenblatt Award in 2020. His work won the 2001 IEEE Donald G. Fink Prize Paper Award, the 2010, 2016, and 2017 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Awards, the 2011 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award, and many other best paper awards. He was the President of IEEE CIS from 2014 to 2015 and the Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008. He was a Distinguished Lecturer of IEEE CIS.