Solving Multiobjective Optimization Problems in Unknown Dynamic Environments: An Inverse Modeling Approach

Sen Bong Gee, Kay Chen Tan, *Fellow, IEEE*, and Cesare Alippi, *Fellow, IEEE*

***Abstract*—Evolutionary multiobjective optimization in dynamic environments is a challenging task, as it requires the optimization algorithm converging to a time-variant Pareto optimal front. This paper proposes a dynamic multiobjective optimization algorithm which utilizes an inverse model set to guide the search toward promising decision regions. In order to reduce the number of fitness evalutions for change detection purpose, a two-stage change detection test is proposed which uses the inverse model set to check potential changes in the objective function landscape. Both static and dynamic multiobjective benchmark optimization problems have been considered to evaluate the performance of the proposed algorithm. Experimental results show that the improvement in optimization performance is achievable when the proposed inverse model set is adopted.**

***Index Terms*—Change detection, decomposition, dynamic multiobjective optimization, evolutionary computation.**

I. Introduction

MULTIOBJECTIVE optimization in dynamic environments involves simultaneous optimization of several time-varying constraints and/or objective functions. This is a challenging optimization problem as the fitness landscape may change over time. Moreover, the time variance in the fitness landscape is mostly unknown or not observable which further complicates the optimization problem [1]–[3]. Recently, there is a rising interest in using evolutionary algorithms (EAs) to address this challenging issue [4]–[7] partly due to the success of EAs in solving static multiobjective optimization problems (MOPs) [8]–[14] and dynamic optimization problems [15]–[17]. However, most of the existing literature considers that the change profile of the fitness landscape is known and happens in between algorithm's evolutionary generations. Typically, in these algorithms, the population is diversified to avoid premature convergence after the change in fitness landscape is detected. In this direction, the random immigrant scheme approach adds randomly generated solutions into the population to enhance its solution diversity [18], [19], whereas hyper-mutation scheme one enhances the solution-space exploration by adapting the mutation rate [20]–[22]. Another popular approach is to learn the pattern of the time-varying optimal solution set in decision space and predicts the location of the consecutive optimal solution set after the change is recognized [4], [23]–[25]. Although these algorithms have partly solved the dynamic MOP, their effectiveness is still questionable if the change in the environment is, as mostly happens, uknown or hardly predictable.

To adapt to the change affecting the fitness landscape in environments with unknown dynamics, there are two major strategies that can be used: 1) active and 2) passive. In the active case, the change detection mechanism is implemented in the optimization algorithm and aims at detecting changes in fitness landscape. Once the change has been detected, the optimization algorithm takes certain action to search for Pareto optimal solutions. This approach is effective only if the change affecting the dynamic environment is detectable. In the passive case, there are no dedicated change detection mechanisms. An optimization algorithm adopting passive approach evolves based on available information independently from the fact that change happened or not. Aging scheme [26], [27] is a typical example of a passive approach. Instead of detecting changes in fitness landscape, aging scheme re-evaluates or discards individual solutions which survive more than a predefined number of generations. In [7], a multipopulation multiobjective EA framework is proposed to solve dynamic MOP in undetectable environments. An environment is undetectable when there is not enough available information for a detection algorithm to effectively detect a change. For example, there are some dynamic problems whose fitness landscape only changes on some random subareas in the search space [7]. The framework uses a hierarchical clustering technique to track multiple optima. It limits the survival of solutions based on the improvement of the solution over evolutionary generations without using any detection mechanism. On one hand, a passive approach is more robust to different types of changes and it can be used in undetectable dynamic environment. On the other hand, an active approach could provide additional

Manuscript received February 25, 2016; revised June 12, 2016; accepted August 11, 2016. This work was supported by the Singapore Ministry of Education Academic Research Fund Tier 1 under Project R-263-000-A12-112. This paper was recommended by Associate Editor Q. Zhang.

S. B. Gee and K. C. Tan are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: a0039834@u.nus.edu; eletankc@nus.edu.sg).

C. Alippi is with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy, and also with the Università della Svizzera Italiana, 6900 Lugano, Switzerland (e-mail: cesare.alippi@polimi.it).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. The supplementary PDF file provides the optimization performances of the proposed IM-MOEA/D on static multiobjective benchmark test suites. The total size of the file is 72.9 KB.

information of changes in the fitness landscape which could facilitate the decision making process. This paper focuses on the active approach and develops a detection mechanism for changes associated with a drift or an abrupt in the fitness function.

One of the most commonly used change detection strategies in dynamic multiobjective EAs is to regularly re-evaluate the fitness of certain solutions, (see [28], [29]). As the purpose of solution re-evalutions is to detect the change in fitness landscape, these solutions are usually called detectors [30]. Detectors monitor the change of objective values by comparing recently evaluated objective vectors with past evaluated ones. When the difference of these two objective vectors is larger than a certain threshold, it is concluded that there is a change in the fitness landscape. The threshold is influenced by the noise level and uncertainty in the fitness evaluations. Although this approach is simple and easy to be implemented, extra fitness function evaluations are required for detecting purpose. Moreover, an appropriate number of detectors has to be chosen, as excessive number would introduce an unneccessary computational cost whereas an insufficient number of detectors would fail in detecting the change or introduce a high level of false positives [31]. Change detection based on solution decision and objective distribution information is another attractive mechanism as it does not require additional function evaluations. However, the detection may not be accurate as there is no dedicated detectors [30], [32].

Most of the existing dynamic multiobjective optimization literature assumes that the change of fitness landscape happens *in between generations*. However, in the real world scenario, the change may happen *within a generation*. To the best of the authors' knowledge, there are relatively few studies in the literature focusing on this issue. Detecting change *within a generation* requires finer time scale to define the processing sequence. In this respect, this paper considers the solutions are evaluated in a serial manner and change may happen in between solutions' evaluation.

In the evolutionary dynamic multiobjective optimization literature [18], [25], it is commonly assumed that the change of the optimization problem is gradual, i.e., the fitness landscape changes smoothly over. Convergence speed of the dynamic EAs is a crucial issue due to the stringent time constraint of the problem. As the fitness landscape of the dynamic multiobjective problem is changing over time, diversity of the solution set is important to avoid premature convergence after the change happens. Therefore, the balance between population convergence and diversity is extremely important for the practical use of optimization algorithms [4]. Inspired by the recently proposed inverse modeling approach [33] which has empirically shown fast convergence speed, this paper presents a decomposition-based multiobjective EA, inverse modeling-based multiobjective EA (IM-MOEA/D), with inverse modeling, to solve the dynamic MOP. The proposed algorithm continuously monitors the fitness landscape for changes by observing the error between the target objective value and the current evaluated objective discrepancy.

This paper is organized as follows. Section II formalizes the dynamic MOP considered in this paper. The literature about change detection and surrogate model is briefly reviewed. Section III proposes the dynamic multiobjective EA and the change detection mechanism. Experiment results are presented and analyzed in Section IV. Finally, conclusions are drawn in Section V.

II. Background

In this section, some basic definitions used in the evolutionary multiobjective optimization community and change detection literature are introduced. The purpose is to provide certain background for the understanding of the sequel.

A. Dynamic Multiobjective Optimization

In this paper, we consider the dynamic MOP as a time-variant MOP. Without loss of generality, a minimization problem is considered here. The dynamic multiobjective optimization can be formalized as

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \;\; \mathbf{f}(\mathbf{x}, t) = \left[f_1(\mathbf{x}, t)\, f_2(\mathbf{x}, t) \;\; \ldots \;\; f_m(\mathbf{x}, t)\right]^T \\ &\text{subject to} \;\; \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

where f_i is the ith objective function out of m objectives; $t \in \mathbb{N}$ is the time; $\mathbf{f}(\mathbf{x}, t) \in \mathbb{R}^m$ is the objective vector; $\mathbf{x} \in \mathbb{R}^n$ is the decision vector; n is the number of decision variables and $\Omega \in \mathbb{R}^n$ is the feasible decision space. The time value is often associated with the generation number of an EA. The generation to time conversion is calculated as follows [28]:

$$t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_T} \right\rfloor \tag{2}$$

where τ is the generation number, n_t is number of distinct step per time unit and τ_T is the number of generations where the fitness landscape is fixed. To consider the case where change happens *within a generation*, the time value is associated with the fitness evaluation number of an EA. The fitness evaluation number to time conversion is calculated as follows:

$$t = \frac{1}{n_t}\left\lfloor \frac{\tau_{\text{eval}}}{\tau_{\text{eval},T}} \right\rfloor \tag{3}$$

where τ_{eval} is the fitness evaluation number, $\tau_{\text{eval},T}$ is the number of fitness evaluations where the fitness landscape is fixed. Only continuous optimization problems are considered in this paper. Generally, Ω can be described by the set of constraints

$$\begin{aligned} \Omega = \big\{\mathbf{x} \in \mathbb{R}^n \big| & g_j(\mathbf{x}, t) \leq 0 \text{ for } j = 1, \ldots, p; \\ & h_k(\mathbf{x}, t) = 0 \text{ for } k = 1, \ldots, q\big\} \end{aligned} \tag{4}$$

where $g_j(\mathbf{x}, t)$, $j = 1, \ldots, p$, is the jth inequality constraint and $h_k(\mathbf{x}, t)$, $k = 1, \ldots, q$, is the kth equality constraint. Even though the model permits both inequality and equality constraints to be time-dependent, here we focus on the relevant case where only the fitness function $\mathbf{f}(\mathbf{x}, t)$ is time-variant and the feasible decision space is fixed throughout the optimization process (time-variant feasible decision space would require additional mechanisms for validating the feasibility of the solutions and will be objective of future research).

B. Change Detection

By extending the definition given in [30] to the dynamic multiobjective optimization case, a change point t_{cp} is defined as

$$\mathbf{f}\big(\mathbf{x}, t_{cp}\big) \neq \mathbf{f}\big(\mathbf{x}, t_{cp}+1\big) \tag{5}$$

where $\mathbf{f}(\mathbf{x}, t_{cp})$ and $\mathbf{f}(\mathbf{x}, t_{cp}+1)$ are the underlying objective vectors given decision vector $\mathbf{x}$, at time t_{cp} and $t_{cp}+1$, respectively. The time is associated with the run-time of the EA. During the evolutionary search process, some change points are more important than others. In fact, for a minimization problem, detecting a change that causes the whole fitness landscape shift upward [or equivalently, $f_i(\mathbf{x}, t_{cp}+1) \geq f_i(\mathbf{x}, t_{cp}), \forall i \in \{1, \ldots, m\}$] is more important than detecting a change that causes the whole fitness landscape shift downward [or equivalently, $f_i(\mathbf{x}, t_{cp}+1) \leq f_i(\mathbf{x}, t_{cp}), \forall i \in \{1, \ldots, m\}$]. As the goal of an EA is to guide the population toward fitter solutions, evaluated offsprings after a downward change are likely to replace the parent population. Therefore, the negative effects of the downward change is less pronounced than the upward change in fitness landscape. These have been exploited for change detection in [30].

A change detection algorithm is expected to detect changes in fitness landscape when condition (5) is fulfilled. There are two common approaches which are used in the evolutionary computation community for detecting changes in fitness landscape: 1) a fixed detector approach and 2) a behavior-based approach [30], [32]. Fixed detector approach regularly evaluates a number of randomly generated solutions to monitor the change of the detectors' objective values. As most of the existing literature assumes that the change of fitness landscape happens *in between generations*, these detectors are usually re-evaluated at every generation. A behavior-based detection approach inspects the discrepancy between the objective space solution statistics with the algorithm's inherited behavior. This approach can be found in the evolutionary single-objective optimization literature [32]. For example, a change point can be detected by inspecting the trend of the average in offspring's fitness value. If there is a sudden increase of average fitness value in the minimization problem, then there is a change in fitness landscape. Behavior-based detection approaches are attractive as there is no need for additional fitness evaluations to be used to detection purpose.

Change detection latency is an important issue. For the fixed detector approach, the detection mechanism is activated only when the detectors are evaluated. To improve the responsiveness of the detection mechanism, the evaluations of detectors are required to be distributed uniformly across different time instances. Suppose one generation of the EA requires $t_{\text{generation}}$ time to evaluate all the individual solutions with the detectors, the algorithm has to evaluate $n_{\text{detector}}/t_{\text{generation}}$ detectors per unit time to ensure that the detection is activated every unit time where n_{detector} is the total number of detectors used in the algorithm. However, this in turn reduces the coverage of the detectors in the decision space during the activation of the detection. Therefore, it is clear that there is a tradeoff between the coverage of the detectors and the responsiveness of the detection mechanism given a fixed number of detectors.

C. Surrogate and Inverse Modeling

Fitness landscape approximation techniques have been proposed and applied to assist evolutionary search for almost two decades [34]. These fitness approximation models also named surrogates or meta-models are often used to reduce the computational time in optimization problems [35]. Recently, an IM-MOEA has been proposed to reduce the computational cost of the evolutionary optimization procedure [33]. The main idea of the algorithm is to construct an inverse model which maps all found nondominated solutions from the objective space to the decision space. This approach is different from the fitness landscape approximation approach, which, instead models the mapping from decision space to objective space. IM-MOEA uses a Gaussian process-based inverse model on domination-based multiobjective optimization framework [36] to solve the optimization problem. By using the inverse model, the search process can be improved as the model helps to direct the search toward those promising areas which are more probable to contain good solutions. This is a desirable feature for an EA especially when the problem is computationally expensive, dynamic, or the search space is large.

III. Proposed Algorithm

A. Basic Idea

IM-MOEA builds a set of probabilistic models to estimate the probability distribution of the decision vector given the desired objective vector. The information of current parent population is used to build the inverse model. The m-input-n-output probabilistic inverse model is decomposed into $m \times n$ univariate models. As the algorithm makes an implicit assumption that all decision variables are mutually independent, random grouping strategy is introduced into the algorithm to alleviate the negative effect of the assumption violation [33]. Different from IM-MOEA, the proposed IM-MOEA/D is based on decomposition-based MOEA (MOEA/D [37], [38]) rather than domination-based MOEA non-dominated sorting genetic algorithm (NSGA-II [36]). By using weight vector of MOEA/D, the task of objective subspace allocation for different inverse models is simplified.

It is possible that the objective function to be optimized is injective (many-to-one function mapping) or the inverse function of the objective function may not exist. To relax the strict assumption that the objective function is globally bijective (one-to-one function mapping), we assume that the objective function is invertible locally thus the corresponding inverse model can be built to approximate the fitness landscape reasonably. In the proposed IM-MOEA/D, the inverse model is used for detection purpose and to direct the search toward promising decision subspace regions. Building a global inverse model is difficult as training a too complex model is computationally expensive whereas choosing a too simple model would degrade the modeling performance. To circumvent the problem, multiple inverse models are trained to capture the fitness landscape information in different regions. Furthermore, linear

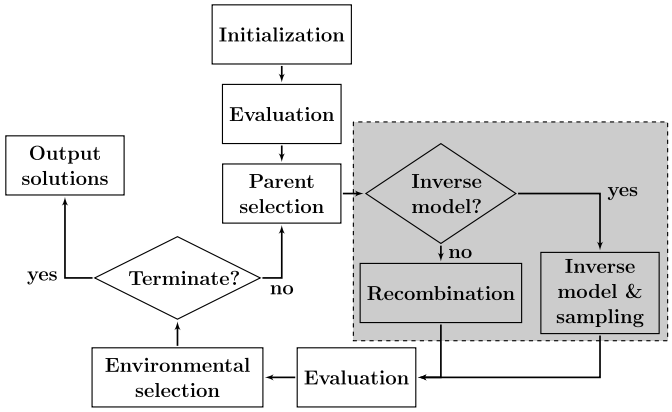

Fig. 1. Flowchart of the proposed algorithm.

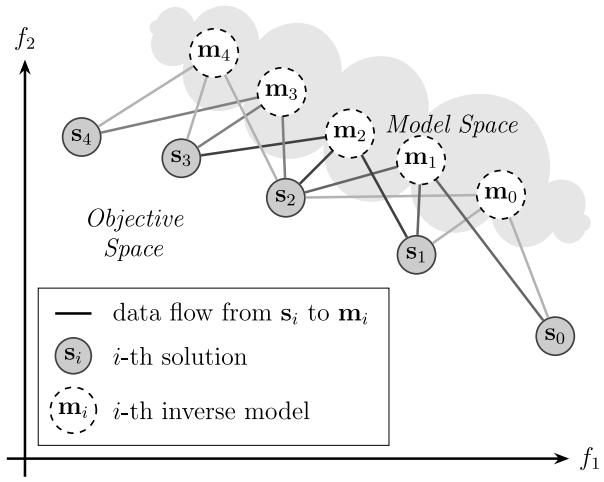

Fig. 2. Objective subspace allocation for training inverse models. In this example, $\mathbf{s}_0$ to $\mathbf{s}_4$ denote five individual solutions in the population. Each solution belongs to 2–3 neighborhoods. Line denotes the data flow from a particular solution to a specific inverse model whereas lightness of line denotes a specific neighborhood relation. Solutions in the same neighborhood are used to train the inverse model. For example, solutions $\mathbf{s}_0$, $\mathbf{s}_1$, and $\mathbf{s}_2$ are in the same neighborhood and their objective and decision vectors are used to train the inverse model $\mathbf{m}_0$.

inverse models are employed in the proposed algorithm due to their relatively low computational cost as compared to other more complicated nonlinear ones.

The overall flowchart of the proposed algorithm is presented in Fig. 1. From the figure, the algorithm follows the EA paradigm and the inverse modeling mechanism is introduced for the offspring population generation as shown in the shaded box. In each generation, an offspring solution is generated using either the crossover operator or the inverse model with sampling mechanism. A decision procedure is designed to allocate offspring solution generation between the former two methods (as shown in Fig. 1). In the following sections, the decision procedure, inverse model, and sampling mechanism will be described in detail. After that, the proposed change detection scheme which utilizes the inverse model is presented.

B. Objective Subspace Allocation and Decision Procedure

Weight vector is an important component in MOEA/D for scalarizing a given MOP into a set of subproblems. The element of a weight vector indicates the relative importance of the corresponding objective function with respect to other objective functions in a given MOP. Tchebycheff approach used in MOEA/D scalarizes a MOP into a set of subproblems using following function:

$$g\left(\mathbf{x}\middle|\lambda, \mathbf{z}^*\right) = \max_{1\le i\le m}\left\{\lambda_i^j\middle|f_i(\mathbf{x}) - z_i^*\right\} \tag{6}$$

where $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$ is the jth individual's weight vector, $\mathbf{z}^* = (z_1, \ldots, z_m)^T$ is an approximate ideal vector and $f_i(\mathbf{x})$, $i = 1, \ldots, m$, is the ith objective for static MOP. A weight vector's element (λ_i^j, $j = 1, \ldots, N$) is usually highly correlated to its corresponding evaluated objective value ($f_i(\mathbf{x}_j)$, $j = 1, \ldots, N$), except during the very early phase of the evolutionary search. This also implies that any two individuals whose weight vectors are adjacent to each other are usually neighbors in the objective space. This property is used in the proposed IM-MOEA/D to allocate different inverse models to different regions in the objective space. Allocating a set of adjacent weight vectors to an inverse model has the same effect as assigning the corresponding objective subspace region to the model. Objective subspace allocation for training different inverse models is an important task as it directly affects the performance of the detection and optimization of the proposed algorithm. To avoid introducing complicated subspace allocation mechanisms which would introduce additional computation resources, the proposed algorithm utilizes the neighborhood relation of MOEA/D for individual solution assignment. In particular, each individual solution with its weight vector's neighbor solutions are used to train an inverse model. Therefore, the number of inverse models is equal to the number of individuals in the population. To further illustrate the concept, a simple example is given in Fig. 2.

MOEA/D takes certain number of generations from the solution initialization to the existence of high correlation between weight vector's element to the corresponding objective value. The actual number of generations is application dependent. A simple decision procedure is devised to decide whether to use the trained inverse model for generating an offspring solution through the Spearman's correlation [39] as shown in Algorithm 1. It is often used to measure the correlation between two variables. An inverse model is only used if the Spearman's correlation of the corresponding weight vector set and the evaluated objective vector set is higher than the predefined confidence interval. Conversely, it indicates that the corresponding fitness vectors may not be adjacent and recombination operator is used for generating a new offspring solution. In this paper, differential evolution mutation operator is used as the recombination operator.

C. Inverse Model

Each individual solution with its weight vector neighborhood solutions are used to form a linear inverse model. Suppose $\mathbf{x}_i = [x_{1i}, x_{2i}, \ldots, x_{ni}] \in \mathbb{R}^n$ and

Algorithm 1 Decision Procedure For Using Inverse Model

Require:

m: number of objective functions
α: decision threshold for Spearman's correlation
$B(i)$: neighbour individual indices of ith individual
$\{\mathbf{f}_k\}$: objective vector set for $k \in B(i)$ where $\mathbf{f}_k$ denotes kth individual's evaluated objective vector
$\{\mathbf{w}_k\}$: weight vector set for $k \in B(i)$ where $\mathbf{w}_k$ denotes kth individual's weight vector

Ensure: Boolean output

1: **for** $j = 1$ to m **do**
2: $\quad \mathbf{f}_{*,j} \coloneqq j$th objective values from $\{\mathbf{f}_k\}$
3: $\quad \mathbf{w}_{*,j} \coloneqq j$th weight element from $\{\mathbf{w}_k\}$
4: $\quad r_j \coloneqq \text{spearman}(\mathbf{f}_{*,j}, \mathbf{w}_{*,j})$
5: **end for**
6: **if** $|r_j| > \alpha$ for $j = 1, \ldots, m$ **then**
7: $\quad$ **return** True
8: **else**
9: $\quad$ **return** False
10: **end if**

$\mathbf{f}_i = [f_{1i}, f_{2i}, \ldots, f_{mi}] \in \mathbb{R}^m$ are the decision and objective vector of the ith individual in the population, respectively. The neighborhood function of the ith individual is denoted as $B(i) = \{b_{1i}, b_{2i}, \ldots, b_{Ti}\}$ which contains the indices of the T closest weight vectors to the ith individual's weight vector, λ^i. This neighborhood function is available in the MOEA/D. The linear regression inverse model of the proposed algorithm can be written as

$$\mathbf{X} = \mathbf{BF} \tag{7}$$

where $\mathbf{X} = [\mathbf{x}_{b_{1i}}, \mathbf{x}_{b_{2i}}, \ldots, \mathbf{x}_{b_{Ti}}]$ and $\mathbf{F} = [\mathbf{f}_{b_{1i}}, \mathbf{f}_{b_{2i}}, \ldots, \mathbf{f}_{b_{Ti}}]$ denote the decision and objective matrices; $\mathbf{B}$ is the matrix that captures the relationship of the inverse model. The $\mathbf{B}$ matrix is computed according to the classic least square procedure as follows:

$$\begin{aligned} \mathbf{XF}^T &= \mathbf{BFF}^T \\ \mathbf{B} &= \left(\mathbf{XF}^T\right)\left(\mathbf{FF}^T\right)^{-1} \end{aligned} \tag{8}$$

where $\{\cdot\}^T$ and $\{\cdot\}^{-1}$ denote matrix transpose and inverse, respectively. Matrix $\mathbf{B}$ contains the coefficients of the linear inverse model given in (7).

D. Sampling Method

Availability of an inverse model enables the generation of the offspring decision vector by specifying the desired objective vector, $\mathbf{f}_{\text{desired}}$. The estimated decision vector is computed as follows:

$$\mathbf{x}_{\text{estimated}} = \mathbf{Bf}_{\text{desired}}. \tag{9}$$

The $\mathbf{f}_{\text{desired}}$ is generated by sampling the objective space. The sample point should be close to the objective subspace used to train the inverse model. As the sample point moves away from the training data, the modeling error tends to increase and the accuracy of the estimated decision vector suffers. Besides, point sampling should move toward directions enhancing the converging speed of the algorithm. As such, the proposed algorithm samples the objective subspace according to the Gaussian distribution and imposes the condition that the sampled point has to dominate its parent solution. The mean and standard deviation of the Gaussian distribution is determined by the parent solution and the estimated ideal point in MOEA/D. For ith individual with previously evaluated objective vector, $\mathbf{f}_i$, the Gaussian distribution is

$$\mathbf{f}_{\text{desired}} \sim \mathcal{N}(\mathbf{f}_i, \Sigma_{\text{diff}}) \tag{10}$$

where Σ_{diff} is a diagonal matrix which contains the absolute difference between $\mathbf{f}_i$ and estimated ideal vector, $\mathbf{z}^*$. The pseudocode of the sampling mechanism is shown in Algorithm 2. In line 3 of the pseudocode, "$\mathbf{f}_{\text{desired}} \not\prec \mathbf{f}_i$" means objective vector $\mathbf{f}_{\text{desired}}$ does not dominate objective vector $\mathbf{f}_i$.

Algorithm 2 Sampling Mechanism

Require:

$\mathbf{f}_i$: i-th individual's evaluated fitness vector
$\mathbf{z}^*$: estimated ideal vector
$\mathbf{B}$: i-th model's inverse model matrix
δ_{sampling}: Sampling parameter to ensure $|\Sigma_{\text{diff}}| > 0$

Ensure: Offspring decision variable, $\mathbf{x}_{\text{offspring}}$

1: $\Sigma_{\text{diff}} = \text{diag}(|\mathbf{f}_i - \mathbf{z}^* + \delta_{\text{sampling}}|)$
2: $\mathbf{f}_{\text{desired}} \sim \mathcal{N}(\mathbf{f}_i, \Sigma_{\text{diff}})$
3: **while** $\mathbf{f}_{\text{desired}} \not\prec \mathbf{f}_i$ **do**
4: $\quad \mathbf{f}_{\text{desired}} \sim \mathcal{N}(\mathbf{f}_i, \Sigma_{\text{diff}})$
5: **end while**
6: $\mathbf{x}_{\text{offspring}} = \mathbf{Bf}_{\text{desired}}$
7: **return** $\mathbf{x}_{\text{offspring}}$

E. Two-Stage Change Detection Test

In a dynamic environment with unknown change, change detection mechanisms are crucial to avoid keeping obsolete solutions. Empirical studies have shown that fixed detector approaches outperform behavior-based ones especially when change detection is difficult [30]. However, a fixed detector approach significantly increases the number of function evaluations since a high number of detectors is desirable. To reduce the number of function evaluations and improve the coverage of the detection region, a two-stage change detection approach is proposed. In the first stage, a sequential change point detection test is used to monitor potential changes in fitness landscape. The first-stage detection mechanism has larger coverage of the detectable region than the fixed detectors' approach. Each offspring solution generated by the inverse model acts as a weak detector in the decision space to probe potential changes in the fitness landscape. The second stage has to confirm the suspected change by the fixed detector approach. The proposed method reduces the required number of fitness evaluations for the detection purpose as compared to the fixed detector approach because it is not required to re-evaluate a fixed number of solutions in correspondence with every generation. In addition, a subset of the whole population acts as weak detectors during the evolutionary search and this improves the coverage of the detectable decision space as

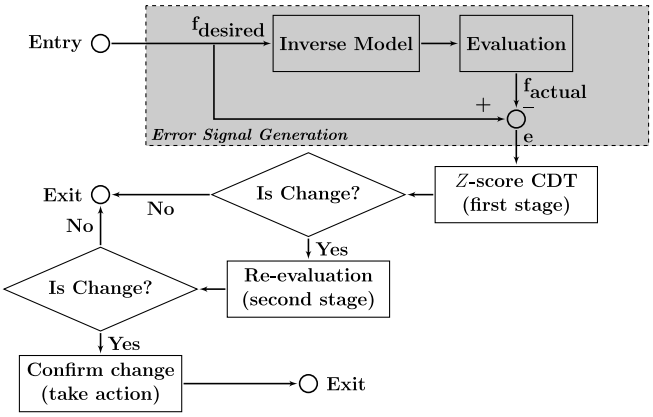

Fig. 3. Flowchart of the two-stage CDT.

compared to the fixed detector approach. The flow chart of the proposed two-stage change detection mechanism is shown in Fig. 3.

In the first stage, the inverse model set is used to assist the task of change detection. The linear inverse model set captures the objective-to-decision mapping information of the explored regions. This information is utilized to determine whether there is a change in the fitness landscape of the explored regions. As the algorithm takes explicit action to respond to a change in the fitness landscape, an online change detection mechanism [40] is more suitable to the problem. The error of the actual objective vector and the desired objective vector is used to detect the change in the fitness landscape

$$\mathbf{e} = \mathbf{f}_{\text{desired}} - \mathbf{f}_{\text{actual}} \tag{11}$$

where $\mathbf{f}_{\text{desired}}$ is the desired objective vector in (9) and $\mathbf{f}_{\text{actual}}$ is the evaluated objective vector given the input $\mathbf{x}_{\text{estimated}}$. There are at least two important sources which contribute to this error. First, the proposed algorithm uses the linear model to approximate the regional objective-to-decision mapping information. This error is related to the nonlinearity of the fitness landscape. Second, the input vector, $\mathbf{x}_{\text{estimated}}$, is estimated using previously evaluated objective vectors. If there is a change in the fitness landscape, the change will contribute to the error as the amplitude of this error is related to the change magnitude of the explored fitness landscape region. To ease the detection task, the second source of error is assumed to dominate over the first contribution. With this assumption, detecting changes by monitoring the abnormality of the error signal is possible. By transforming the task of detecting changes in fitness landscape to identifying abnormality in the signal, sequential change-point detection tests can be applied to solve the problem [41], [42].

In this paper, a fixed sliding window method [40], [43] has been used for detecting the change in the first stage. n_{window} samples (window size) are taken from the end of sequential signal values and a decision rule is applied on hypothesis

$$H_0 : \mathbf{f}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t_{\text{last}}), \mathbf{f}(\mathbf{x}, t) \in \delta_i$$
$$H_1 : \mathbf{f}(\mathbf{x}, t) \neq \mathbf{f}(\mathbf{x}, t_{\text{last}}), \mathbf{f}(\mathbf{x}, t) \in \delta_i$$

Algorithm 3 Z-Score-Based Change Detection

Require:
$\{x_k\}$: input signal values for $k = 1, 2, \ldots, \tau_{\text{eval}}$ where τ_{eval} is the current number of evaluations
n_{window}: size of the sliding window
θ_d: detection threshold

Ensure: Boolean output

1: Estimated global and window statistic values, which are mean and standard deviation ($\hat{\mu}_{\text{global}}$, $\hat{\sigma}_{\text{global}}$, $\hat{\mu}_{\text{window}}$, $\hat{\sigma}_{\text{window}}$), are required to detect the change. To online compute the estimated global mean, Welford's method is used [45].
2: $\hat{\mu}_{\text{global}}$, $\hat{\sigma}_{\text{global}}$ = Welford($x_{\tau_{\text{eval}}}$)
3: **if** $\tau_{\text{eval}} < n_{\text{window}}$ **then**
4: **return** False
5: **else**
6: $\hat{\mu}_{\text{window}}$ = mean($[x_{\tau_{\text{eval}}}, x_{\tau_{\text{eval}}-1}, \ldots, x_{\tau_{\text{eval}}-n_{\text{window}}+1}]$)
7: $\hat{\sigma}_{\text{window}}$ = stdev($[x_{\tau_{\text{eval}}}, x_{\tau_{\text{eval}}-1}, \ldots, x_{\tau_{\text{eval}}-n_{\text{window}}+1}]$)
8: SE = $(\hat{\sigma}_{\text{global}} - \hat{\sigma}_{\text{window}})/\sqrt{n_{\text{window}}}$
9: $z = (\hat{\mu}_{\text{window}} - \hat{\mu}_{\text{global}})/$ SE
10: **if** $z > \theta_d$ **then**
11: **return** True
12: **else**
13: **return** False
14: **end if**
15: **end if**

where t_{last} is the time of the previously evaluated objective function and δ_i is the objective subspace covered by the ith individual with its neighborhood. Any sequential change detection algorithm can be employed for detecting the change in the fitness landscape. To accomplish the task, a simple change detector based on the Z-score [40], [44] is used to detect the change in the fitness landscape. The change detector monitors the absolute value of the error signal and makes a decision about whether there is a change in the fitness landscape or not. The pseudo-code of the detection algorithm used in this paper is given in Algorithm 3. Notice that the algorithm detects the fitness landscape changes when the Z-score is higher than the predefined detection threshold (as shown in line 10, Algorithm 3). The input signal to the Algorithm 3 is the error signal computed using (11) (as shown in Fig. 3).

Once the first-stage detection is triggered, a second-stage detection is executed to confirm the change by randomly re-evaluating a fixed number of individuals in the population. If the recently evaluated function values are significantly different from the previously evaluated ones, change in the fitness landscape is confirmed. The fitness values of the rest of the population are evaluated as well. Since the first-stage detection can be triggered any time during the evolutionary search, there is no assumption that the change in fitness landscape only happened *in between* generations. The pseudo-code for the two-stage change detection test (CDT) is shown in Algorithm 4. There, $n_{\text{min-int}}$ denotes the minimum number of fitness evaluations between two fitness landscape changes. This parameter is used to control the number of activations

Algorithm 4 Two-Stage CDT

Require:

$\{x_k\}$: input signal values for $k = 1, 2, \ldots, \tau_{\text{eval}}$ where τ_{eval} is the current number of evaluations

n_{window}: size of the sliding window

θ_d: detection threshold

n_{fixed}: number of fixed detectors used

$\tau_{\text{eval,min-int}}$: minimum number of fitness evaluations between two changes

$\tau_{\text{eval,triggered}}$: a global variable which records the latest evaluation number when the change detection is triggered

Ensure: Boolean output

if $\tau_{\text{eval,triggered}}$ is not initialized **then**
  $\tau_{\text{eval,triggered}} = 0$
end if
for each new observation, $x_{\tau_{\text{eval}}}$ **do**
  Feed $x_{\tau_{\text{eval}}}$ into Algorithm 3 for the first-stage change detection test and get the output, θ.
  $\tau_{\text{eval,int}} = \tau_{\text{eval}} - \tau_{\text{eval,triggered}}$
  if $\tau_{\text{eval,int}} \geq \tau_{\text{eval,min-int}}$ and θ is true **then**
    Re-evaluate n_{fixed} randomly selected solutions to confirm the change.
    if change is confirmed **then**
      The first-stage change detection is reset and set $\tau_{\text{eval,triggered}} = \tau_{\text{eval}}$.
      return True
    else
      return False
    end if
  end if
  return False
end for

for second-stage CDT which can be triggered in a generation. Different from the first-stage CDT, the second-stage CDT requires additional fitness evaluations. The number of second-stage CDTs per generation is limited to $\lceil N/n_{\text{min-int}} \rceil$ where N is the population size.

F. Overall Algorithm

The pseudo-code of the overall procedure is shown in Algorithm 5. In each generation, the proposed algorithm makes a decision about whether to use an inverse model for each offspring solution generation. If an inverse model is used for generating an offspring solution, the generated solution acts as a weak change detector for the first-stage CDT. The error signal [using (11)] is computed to feed the first-stage CDT. Once the first-stage CDT is triggered, the second-stage CDT will be performed to confirm the change. A comparison test is designed for the second-stage CDT. If any of the recently evaluated objective values are different from the stored objective values, it is confirmed that there is a change in the fitness landscape.

For simplicity, this paper only considers noiseless fitness landscape. For the case of noisy fitness landscape, modification

Algorithm 5 Dynamic IM-MOEA/D

Require:

A dynamic multiobjective problem
A stopping criterion
N: Population size
T: Number of the weight vectors in the neighbourhood
n_r: Maximal number of solutions can be replaced
δ: Probability that neighborhood is used over population
α: Cutoff for using an inverse model
n_{window}: Size of the sliding window
θ_d: detection threshold

Ensure:

Current approximated POF at time t, $\{\mathbf{f}^1, \ldots, \mathbf{f}^N\}$
Current approximated POS at time t, $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$

Step 1 ➢ Initialization:

1) Generate evenly spread weight vectors. Find the T closest weight vectors (in terms of Euclidean distance) for each vector. Set $B(i) = i_1, \ldots, i_T$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i.
2) Generate an initial population, $\mathbf{x}^1, \ldots, \mathbf{x}^N$, by uniformly random sampling the decision space. Evaluate each solution and set $\mathbf{f}^i = \mathbf{f}(\mathbf{x}^i, t)$.
3) Initialize $\mathbf{z}$ by setting $z_k = \min_{j=1,\ldots,N} f_k^j$ where $k = 1, \ldots, m$ and f_k^j is the k element of $\mathbf{f}^j$.

Step 2 ➢ Update: Set For $i = 1, \ldots, N$, do

1) **Mating selection/update pool:** Set $P = B(i)$ if uniformly distributed random number, $rand() < \delta$. Otherwise, set $P = \{1 \ldots, N\}$.
2) **Decision procedure:** Apply Algorithm 1 to decide whether an inverse model is used to generate an offspring solution.
3) **Reproduction:** Based on the decision from the previous step, generate the solution either using a heuristic crossover operator or an inverse model. If the inverse model is used, compute the $\mathbf{B}$ matrix (8) and use Algorithm 2 to generate $\mathbf{f}_{\text{desired}}$ and offspring solution with decision vector $\mathbf{y}$. Apply mutation operator and repair the solution $\mathbf{y}$ if it is not in the feasible decision space.
4) **Update of z:** Evaluate $\mathbf{y}$ to get $\mathbf{f}_{\text{actual}} = \mathbf{f}(\mathbf{y}, t)$. If $f_j(\mathbf{y}) < z_j$ for any $j \in \{1, \ldots, m\}$, set $z_j = f_j(\mathbf{y}, t)$.
5) **Update of solutions:** Apply decomposition-based selection [38]. Set $c = 0$ and then do the following:
   a) If $c = n_r$, exist the inner loop. Otherwise, randomly pick an index $j \in P$.
   b) If $g(\mathbf{y}|\lambda^j, \mathbf{z}) <= g(\mathbf{x}|\lambda^j, \mathbf{z})$, set $\mathbf{x}^j = \mathbf{y}$, $\mathbf{f}^j = \mathbf{f}(\mathbf{y}, t)$ and increment c by one.
   c) Remove j from P and go to a).

Step 3 ➢ Change detection: If the inverse model is used, compute $|\mathbf{f}_{\text{desired}} = \mathbf{f}_{\text{actual}}|$ and feed the value into Algorithm 4. Re-evaluate the rest of the population and go to **Step 1.3** to reset $\mathbf{z}$ if the returned value is true.

Step 4 ➢ Stopping criterion: If the stopping criterion is satisfied, stop the process, output $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ and $\{\mathbf{f}^1, \ldots, \mathbf{f}^N\}$. Otherwise, go to **Step 2**.

is needed for the second-stage CDT and the evolutionary selection to account for the uncertainty in the fitness values [46]. For the evolutionary selection, one possibility is to modify the subproblem in the MOEA/D to account for the uncertainty [47]. For the second-stage CDT, one example is to set a specific threshold for the acceptable amount of

objective function deviation for the detectors [28]. Once the objective deviation is greater than a predefined threshold, the second-stage CDT confirms there is a change in the fitness landscape.

Although the fitness landscape is desired to be locally invertible, it is not a strict requirement for the proposed algorithm to work properly. A good local linear inverse model would guide the search toward promising decision regions and have higher change detection accuracy. When the fitness landscape is not locally invertible, the inverse model directs the search direction toward less promising decision regions. Besides, the error between the desired objective vector and the evaluated objective vector would be increased and cause a spike of error signal. As the proposed sampling method and recombination operator introduce randomness in the search process, the inverse model unlikely causes the solution population trapping into some local optima. The spike of error signal likely triggers the second-stage CDT and fixed detection approach is performed. This means poor local linear model leads to an increase number of triggering second-stage CDT. The proposed detection algorithm takes advantage of the possibility to reduce fitness evaluations using local linear inverse model. If the inverse model is not useful, the two-stage CDT still behaves like a fixed detection approach.

IV. Experiment Results

In this section, comparative studies are performed to investigate the optimization performance of the proposed algorithm in dynamic environments. Furthermore, a section is devoted to study the fitness landscape change detection performance. The optimization performance of the proposed algorithm in static environment can be found in the supplementary material. Zitzler–Deb–Thiele test suite [48], Walking Fish Group test suite [49], Congress on Evolutionary Computation 2009 MOEA Competition test suite (CEC-09) [50], Farian–Deb–Amato test suite (FDA) [28], and Gee–Tan–Abbass test suite (GTA) [51] benchmarks have been considered. All results are obtained by performing 30 independent runs for each test problem under a specific setting. NSGA-II [36] and MOEA/D [38] serve as benchmark algorithms in the comparative studies. Differential Evolution mutation operator is used as genetic operator as reported in [38]. Inverted generational distance (IGD) [38] and generational distance (GD) [36] are used as performance metrics in this paper. For each experiment setting, all MOEAs have the same number of fitness evaluations to ensure the fair comparative studies. The total number of fitness evaluations equals to the product of the population size and the generation numbers. Welch's *t*-test [52] is conducted to test the significance performance difference between the IM-MOEA/D and other algorithms. The null hypothesis is that there is no significant performance difference between the IM-MOEA/D and other algorithm given that the two algorithm's performances have unequal variances.

A. Parameter Settings

The parameter settings of the algorithms used in the experiment are shown in Table I. The proposed inverse model

TABLE I
Parameter Settings

Parameters	**Values**
Population size (N)	100
Neighbourhood size (T) & neighbourhood prob. (δ)	20, 0.1
Maximum number of replacement (n_r)	2
Scaling factor (F) & crossover rate (Cr)	0.5, 1.0
Distribution index in mutation (η_m) & mutation rate (p_m)	20, $1/n$
Cutoff for using inverse model (α) & detection threshold (θ_d)	0.7, 1.0
No. fixed detectors (n_{fixed}) & no. min. interval ($n_{\text{min-int}}$)	1, 100
Size of sliding windows (n_{window})	100
Samping parameter (δ_{sampling})	0.01

set introduces α which serves as a threshold for deciding whether to use the inverse model in offspring solution generation. As the Spearman's rank correlation coefficient ranges from -1 to 1, the recommended range for the α is from 0.5 to 1.0. The higher the α threshold, the less likely that the inverse model set is used. The number of fixed detectors, n_{fixed}, controls the number of sampling points to decide whether there is a change in fitness landscape. This parameter is not specific to the proposed algorithm and it has been commonly used in existing dynamic MOEAs. In this paper, the minimum value for n_{fixed} is used to demonstrate the minimum amount of fitness re-evaluation for detection purpose. To avoid more than one second-stage CDT triggered in a generation, $n_{\text{min-int}}$ can be set to the population size. Size of sliding windows is a common parameter for sequential CDT [40]. It is recommended to use the same value as the population size. Sampling parameter, δ_{sampling}, is used to avoid infinite loop when $\mathbf{f}_i$ is equal to $\mathbf{z}^*$ happened.

B. Dynamic Multiobjective Optimization

To assess the optimization performance of the proposed algorithm on dynamic MOP, FDA and GTA test suites are used in the simulation experiments. For each test problem, total generation number, n_{gen}, is set to 500. In this paper, n_t and τ_T [used in (2)] are set to 5 and 10, respectively. In terms of fitness evaluation number, the corresponding $\tau_{\text{eval},T}$ value is 1000. To summarize the overall performance of an algorithm on optimizing a specific test problem, average of performance metric values and average of ranks are used. Averages performance metric values of a given algorithm are computed as follows:

$$\text{IGD}_{\text{mean}} = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \text{IGD}_i \tag{12}$$

$$\text{GD}_{\text{mean}} = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \text{GD}_i \tag{13}$$

where IGD_i and GD_i are the IGD and GD values of the algorithm at *i*th generation. Similarly, the averages ranks of a given algorithm are computed as follows:

$$\text{rank}_{\text{mean}}(\text{IGD}) = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \text{rank}(\text{IGD}_i) \tag{14}$$

TABLE II
PERFORMANCE COMPARISON USING DYNAMIC MULTIOBJECTIVE OPTIMIZATION TEST INSTANCES

Algorithm	Type	Metric	FDA1	FDA3	GTA1a	GTA2a	GTA3a	GTA4a
NSGA-II	Ave. rank	IGD	1.3828 ± 0.7829	1.2966 ± 0.7598	0.8397 ± 0.4677	0.9860 ± 0.3317	0.8397 ± 0.7897	0.8016 ± 0.7771
		GD	1.5070 ± 0.7685	1.8637 ± 0.6741	0.5651 ± 0.5848	0.6733 ± 0.7270	**0.5210 ± 0.7709**	1.0000 ± 0.9042
	Stat.	IGD	0.5349 ± 0.5568 †	1.0051 ± 0.5414 †	0.0973 ± 0.0898 †	0.1547 ± 0.1437 †	**0.5332 ± 0.4896**	**0.7241 ± 0.5593**
		GD	1.2161 ± 1.7431 †	2.4859 ± 2.0835 †	0.1294 ± 0.2185 †	0.2213 ± 0.3803 †	**0.7634 ± 0.8913**	1.0208 ± 0.9208
MOEA/D	Ave. rank	IGD	1.4248 ± 0.5258	1.5311 ± 0.4990	2.1643 ± 0.4995	2.3607 ± 0.5786	1.7214 ± 0.4701	1.7555 ± 0.4699
		GD	1.3046 ± 0.5693	1.1102 ± 0.3257	1.8938 ± 0.3619	1.7956 ± 0.4502	1.5351 ± 0.6612	1.3888 ± 0.6738
	Stat.	IGD	0.4234 ± 0.3809 †	0.9334 ± 0.3307 †	0.2104 ± 0.1243 †	0.4254 ± 0.2710 †	0.7288 ± 0.6087 †	0.9302 ± 0.7628 †
		GD	0.8292 ± 0.9891 †	1.6205 ± 0.9630 †	0.2225 ± 0.1859 †	0.4637 ± 0.4211 †	0.9189 ± 0.9910	1.1866 ± 1.1246 †
IM-MOEA/D	Ave. rank	IGD	**0.2064 ± 0.4426**	**0.1784 ± 0.3828**	**0.2184 ± 0.4547**	**0.0641 ± 0.2450**	**0.4429 ± 0.5576**	**0.4529 ± 0.5439**
		GD	**0.2084 ± 0.4111**	**0.1142 ± 0.4016**	**0.5411 ± 0.5731**	**0.5311 ± 0.5560**	0.9439 ± 0.6706	**0.6112 ± 0.6527**
	Stat.	IGD	**0.2912 ± 0.2528**	**0.6820 ± 0.2684**	**0.0684 ± 0.0777**	**0.0884 ± 0.1102**	0.5702 ± 0.5796	0.7380 ± 0.6579
		GD	**0.6288 ± 0.8817**	**1.2323 ± 0.8977**	**0.1114 ± 0.1607**	**0.1529 ± 0.2427**	0.8270 ± 0.9066	**1.0168 ± 1.0141**
Algorithm	Type	Metric	GTA5a	GTA6a	GTA7a	GTA8a	GTA1m	GTA2m
NSGA-II	Ave. rank	IGD	0.9800 ± 0.3344	**0.5651 ± 0.7499**	1.1463 ± 0.4439	0.9679 ± 0.9059	**0.5691 ± 0.6765**	0.5992 ± 0.6485
		GD	1.2665 ± 0.6542	0.8697 ± 0.8476	1.2906 ± 0.6919	1.2425 ± 0.8128	**0.3768 ± 0.5857**	**0.3948 ± 0.5754**
	Stat.	IGD	0.2133 ± 0.1256 †	**0.9176 ± 0.6098** †	0.4917 ± 0.2155 †	1.1900 ± 0.6967	**0.0572 ± 0.0429** †	**0.0809 ± 0.0641**
		GD	0.3813 ± 0.3413 †	**1.3629 ± 1.0388**	0.7984 ± 0.3550 †	1.9778 ± 1.1118 †	**0.0686 ± 0.0880** †	**0.1035 ± 0.1482** †
MOEA/D	Ave. rank	IGD	2.3427 ± 0.5634	1.7916 ± 0.4255	2.1222 ± 0.7825	1.5631 ± 0.5533	1.9699 ± 0.6797	1.9078 ± 0.3221
		GD	1.4489 ± 0.7211	1.4529 ± 0.6513	1.4870 ± 0.6083	1.3427 ± 0.6795	1.7054 ± 0.6100	1.7896 ± 0.5426
	Stat.	IGD	0.5159 ± 0.3399 †	1.1560 ± 0.6884 †	0.7504 ± 0.4380 †	1.3617 ± 0.8196 †	0.0932 ± 0.0586 †	0.1288 ± 0.0754 †
		GD	0.6054 ± 0.5323 †	1.5600 ± 1.1239 †	1.0310 ± 0.6208 †	2.0187 ± 1.2386 †	0.1123 ± 0.0940 †	0.1692 ± 0.1466 †
IM-MOEA/D	Ave. rank	IGD	**0.0661 ± 0.2485**	0.6453 ± 0.5666	**0.0721 ± 0.2664**	**0.4810 ± 0.5673**	0.6473 ± 0.6071	**0.5030 ± 0.5235**
		GD	**0.2846 ± 0.5174**	**0.6774 ± 0.7288**	**0.2224 ± 0.4744**	**0.4148 ± 0.5991**	0.9178 ± 0.6261	0.8156 ± 0.5923
	Stat.	IGD	**0.1486 ± 0.1020**	0.9768 ± 0.6618	**0.4068 ± 0.2104**	**1.1768 ± 0.7653**	0.0704 ± 0.0499	0.0844 ± 0.0599
		GD	**0.2480 ± 0.2400**	1.3814 ± 1.0680	**0.6294 ± 0.3225**	**1.8418 ± 1.1898**	0.0934 ± 0.0925	0.1233 ± 0.1404
Algorithm	Type	Metric	GTA3m	GTA4m	GTA5m	GTA6m	GTA7m	GTA8m
NSGA-II	Ave. rank	IGD	0.9198 ± 0.6406	1.4950 ± 0.6829	0.9619 ± 0.5488	1.0762 ± 0.6746	0.6653 ± 0.7009	1.0180 ± 0.8142
		GD	1.0401 ± 0.7998	1.8958 ± 0.3367	1.2565 ± 0.5891	1.6754 ± 0.5895	1.0381 ± 0.9001	1.5812 ± 0.7669
	Stat.	IGD	0.3270 ± 0.2681 †	0.4654 ± 0.3041 †	0.1497 ± 0.0681 †	0.6429 ± 0.3260 †	0.4000 ± 0.2296	0.7515 ± 0.4397 †
		GD	0.6112 ± 0.5212 †	1.0091 ± 0.5726 †	0.2805 ± 0.1652 †	1.3721 ± 0.5805 †	0.6698 ± 0.2780 †	1.5383 ± 0.7055 †
MOEA/D	Ave. rank	IGD	1.7074 ± 0.6320	1.2806 ± 0.7088	1.8998 ± 0.4317	1.6974 ± 0.6028	1.7956 ± 0.4968	1.5411 ± 0.5801
		GD	1.1984 ± 0.6780	0.7074 ± 0.5359	1.4930 ± 0.7003	1.0441 ± 0.6023	1.4649 ± 0.6269	1.0721 ± 0.6029
	Stat.	IGD	0.4071 ± 0.3442 †	0.5214 ± 0.4080 †	0.2147 ± 0.1149 †	0.7458 ± 0.3833 †	0.4460 ± 0.2585 †	0.8171 ± 0.4899 †
		GD	0.6094 ± 0.4862 †	0.7694 ± 0.5667 †	0.3122 ± 0.1695 †	1.2231 ± 0.5908 †	0.6344 ± 0.3300 †	1.3437 ± 0.7518 †
IM-MOEA/D	Ave. rank	IGD	**0.3988 ± 0.6293**	**0.2505 ± 0.4602**	**0.1784 ± 0.3828**	**0.2565 ± 0.4677**	**0.6072 ± 0.7288**	**0.5070 ± 0.7942**
		GD	**0.7615 ± 0.8960**	**0.3968 ± 0.5895**	**0.2505 ± 0.5403**	**0.2806 ± 0.5604**	**0.5010 ± 0.5815**	**0.3487 ± 0.5359**
	Stat.	IGD	**0.2734 ± 0.2683**	**0.3405 ± 0.3293**	**0.1271 ± 0.0642**	**0.5589 ± 0.3571**	**0.3985 ± 0.2365**	**0.6888 ± 0.4469**
		GD	**0.5004 ± 0.4469**	**0.6017 ± 0.5489**	**0.2282 ± 0.1489**	**1.0346 ± 0.6052**	**0.5807 ± 0.3087**	**1.2372 ± 0.7353**

$$\text{rank}_{\text{mean}}(\text{GD}) = \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \text{rank}(\text{GD}_i) \tag{15}$$

where rank$(\cdot)$ is an operator which ranks the performance metric of the algorithm among all algorithms' performance metric value at a specific generation. The output of the operator ranges from 0 to $k-1$, where k is the total number of algorithms used for the comparative study. Table II has recorded the above-mentioned performance metrics of the three algorithms used in the experiment. The values in the row "Ave. rank" are computed using either (14) or (15) whereas the values in row "Stat." are calculated using either (12) or (13). We observe that the proposed algorithm performs well in most of the benchmarks. This suggests that the inverse model set used in the proposed algorithm could be useful in improving the algorithm's convergence speed.

C. Change Detection

As the main purpose of the proposed two-stage CDT is to reduce the fitness evaluations used for detection purpose, number of fitness evaluations spent for monitoring change of fitness landscape is used as one of the performance metrics. Average of root mean square error (RMSE) between the actual objective vectors and the nominal objective vectors is also used to assess the proposed two-stage CDT's performance. A nominal objective vector of an individual is the objective vector of the past evaluated solution. If there is a change in the fitness landscape, the nominal objective vector is different from the actual objective vector. The metric is computed as follows:

$$\text{RMSE}_{\text{mean}} = \frac{1}{n_{\text{gen}} N} \sum_{\tau=1}^{n_{\text{gen}}} \sum_{i=1}^{N} \left| \mathbf{f}_{\tau}^{i} - \mathbf{f}\left(\mathbf{x}^{i}, \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_T} \right\rfloor\right) \right| \tag{16}$$

where $\mathbf{f}_{\tau}^{i}$ is the ith individual's nominal objective vector at generation τ; $\mathbf{f}(\mathbf{x}^{i}, (1/n_t)\lfloor(\tau/\tau_T)\rfloor)$ is the evaluated $\mathbf{x}_i$ decision vector at generation τ. A nominal objective vector could be an obsolete objective vector if the CDT fails to detect a change in the fitness landscape. A perfect CDT will result in a zero average of RMSE. This implies that the nominal objective vector of any decision vector is equal to its actual objective vector.

TABLE III
OPTIMIZATION AND DETECTION PERFORMANCE

	FDA1	FDA3	GTA1a	GTA2a	GTA3a	GTA4a
Ignored (IGD)	5.636 ± 5.026	6.263 ± 5.284	0.623 ± 0.387	0.786 ± 0.565	1.652 ± 1.104	2.083 ± 1.263
Ignored (GD)	6.671 ± 5.571	10.769 ± 8.473	1.404 ± 0.621	2.225 ± 0.952	3.201 ± 1.592	3.192 ± 1.663
Fixed (IGD)	**0.292 ± 0.292**	**0.684 ± 0.314**	**0.069 ± 0.082**	**0.090 ± 0.116**	0.600 ± 0.692	**0.739 ± 0.736**
Fixed (GD)	**0.628 ± 0.908**	**1.240 ± 0.938**	**0.110 ± 0.162**	**0.158 ± 0.257**	**0.861 ± 1.035**	**1.016 ± 1.102**
Proposed (IGD)	0.427 ± 1.131	0.706 ± 0.604	0.078 ± 0.096	0.102 ± 0.144	**0.536 ± 1.004**	1.034 ± 1.815
Proposed (GD)	1.124 ± 2.536	1.599 ± 1.928	0.129 ± 0.175	0.190 ± 0.285	1.185 ± 1.811	1.742 ± 2.477
Ignored (RMSE)	4.957 ± 0.459	8.122 ± 0.130	0.985 ± 0.024	1.550 ± 0.028	2.244 ± 0.088	2.312 ± 0.071
Proposed (RMSE)	0.223 ± 0.251	0.445 ± 0.219	0.016 ± 0.007	0.034 ± 0.012	0.514 ± 0.363	0.846 ± 0.654
Re-evaluation no.	7113.33 ± 466.00	7700.00 ± 601.11	8013.33 ± 511.69	7473.33 ± 517.00	5010.00 ± 462.13	3250.00 ± 832.17
Check no.	143.77 ± 11.58	175.47 ± 17.04	184.87 ± 20.50	161.07 ± 18.35	90.73 ± 11.07	48.40 ± 13.58
	GTA5a	GTA6a	GTA7a	GTA8a	GTA1m	GTA2m
Ignored (IGD)	0.861 ± 0.619	2.342 ± 0.939	1.767 ± 2.750	3.589 ± 2.387	0.432 ± 0.281	0.595 ± 0.406
Ignored (GD)	2.379 ± 0.954	4.155 ± 1.232	3.291 ± 2.927	4.813 ± 2.466	0.626 ± 0.298	1.037 ± 0.499
Fixed (IGD)	**0.149 ± 0.112**	**1.018 ± 0.759**	**0.407 ± 0.215**	**1.197 ± 0.869**	**0.071 ± 0.059**	**0.083 ± 0.070**
Fixed (GD)	**0.244 ± 0.244**	**1.431 ± 1.178**	**0.629 ± 0.327**	**1.873 ± 1.297**	**0.093 ± 0.099**	**0.121 ± 0.147**
Proposed (IGD)	0.180 ± 0.243	1.118 ± 1.500	1.178 ± 2.884	2.767 ± 3.032	0.074 ± 0.058	0.081 ± 0.070
Proposed (GD)	0.298 ± 0.365	2.196 ± 2.192	1.473 ± 3.001	4.146 ± 3.285	0.104 ± 0.101	0.137 ± 0.162
Ignored (RMSE)	1.620 ± 0.075	2.875 ± 0.076	2.106 ± 0.854	3.264 ± 0.493	0.490 ± 0.046	0.783 ± 0.090
Proposed (RMSE)	0.045 ± 0.054	0.845 ± 0.363	0.647 ± 1.433	2.411 ± 1.278	0.011 ± 0.006	0.021 ± 0.009
Re-evaluation no.	7103.33 ± 638.48	3440.00 ± 350.81	5716.67 ± 2286.93	1450.00 ± 971.25	7826.67 ± 654.69	7616.67 ± 578.55
Check no.	133.27 ± 19.81	54.60 ± 7.53	121.70 ± 50.17	20.42 ± 13.15	181.90 ± 20.70	168.07 ± 22.39
	GTA3m	GTA4m	GTA5m	GTA6m	GTA7m	GTA8m
Ignored (IGD)	0.833 ± 0.498	0.911 ± 0.530	0.640 ± 0.432	1.235 ± 0.486	2.055 ± 0.997	2.963 ± 1.117
Ignored (GD)	1.420 ± 0.749	1.384 ± 0.717	1.547 ± 0.732	2.553 ± 0.959	3.773 ± 1.708	4.405 ± 1.618
Fixed (IGD)	0.268 ± 0.276	0.343 ± 0.351	**0.127 ± 0.078**	0.563 ± 0.387	**0.403 ± 0.242**	**0.680 ± 0.466**
Fixed (GD)	**0.491 ± 0.457**	**0.604 ± 0.569**	**0.226 ± 0.163**	**1.042 ± 0.636**	**0.586 ± 0.315**	**1.228 ± 0.745**
Proposed (IGD)	**0.230 ± 0.222**	**0.298 ± 0.278**	0.134 ± 0.083	**0.498 ± 0.358**	2.088 ± 0.985	3.016 ± 1.146
Proposed (GD)	0.559 ± 0.523	0.690 ± 0.635	0.255 ± 0.174	1.170 ± 0.715	3.770 ± 1.716	4.423 ± 1.622
Ignored (RMSE)	1.071 ± 0.087	1.109 ± 0.021	1.150 ± 0.128	1.898 ± 0.059	2.665 ± 0.089	3.112 ± 0.039
Proposed (RMSE)	0.145 ± 0.020	0.195 ± 0.030	0.033 ± 0.013	0.266 ± 0.034	2.663 ± 0.080	3.125 ± 0.044
Re-evaluation no.	6453.33 ± 476.61	5536.67 ± 535.71	7300.00 ± 611.01	5920.00 ± 339.02	0.00 ± 0.00	0.00 ± 0.00
Check no.	128.37 ± 13.83	93.43 ± 16.81	151.30 ± 20.68	103.17 ± 9.88	0.00 ± 0.00	0.00 ± 0.00

Low value of the average RMSE is desired as this implies that the CDT is able to detect the change of fitness landscape when the change is significant. High value of average RMSE implies that the CDT fails to detect the change in fitness landscape even if the change is significant. A change detection experiment is designed to assess the performance of the proposed two-stage CDT. Fixed detector approach is a perfect CDT if the objective evaluation is not noisy and there is no uncertainty in function evaluation. However, fixed detector approach is expensive as the algorithm is expected to evaluate a fixed number of detectors in order to monitor possible changes of fitness landscape. Ignoring the change of fitness landscape is another approach which does not monitor the change of fitness landscape. This approach does not spend any fitness evaluation for detection purpose but it fails to detect any change in the fitness landscape. These two extreme approaches are used in the experiment to compare the optimization performance deterioration due to the absence of CDT mechanism. Similar to previous experiments, test functions from FDA and GTA test suites are employed in the experiment. The experiment results are shown in Table III.

In the table, IGD, GD, and average RMSE values of the ignoring changes approach (denoted as "Ignored"), fixed detector approach (denoted as "Fixed"), and the proposed two-stage CDT approach (denoted as "Proposed") are shown in "mean±standard deviation" format. The number of fitness re-evaluations after change is detected is also recorded in the table (denoted as "Re-evaluation"). The fitness evaluations spent for monitoring changes in fitness landscape is denoted as "Check no." The lowest GD and IGD values for each approach (under a specific benchmark problem) are shown in bold font.

For all the test problems, there are 100 changes of fitness landscape. Therefore, this results 10^4 fitness evalutions spent for re-evaluation the individual solutions as the population number is 100. From the table, it is clear that the proposed algorithm does not detect all changes in the fitness landscape as the number of the function re-evaluations is lower than 10^4. As mentioned in previous experiment, the generation number of all the algorithms in the experiment studies is set to 500. Suppose there is only one fixed detector which is used to monitor the fitness landscape changes. For fixed detector approach, the total number of fitness evaluations which are used to monitor fitness landscape is equal to 500. By inspecting the Check no. row in the table, it is clear that the proposed algorithm spent significantly lower number of fitness evaluations for the detection. For most of the test problems, the optimization performance of the algorithm with proposed two-stage CDT lies between fixed detector approach and ignoring change approach. Except for GTA7m and GTA8m test problems, the

proposed two-stage CDT has significantly lower RMSE values than ignoring change approach's RMSE values. From the experiment results, it is observed that the two-stage CDT is able to detect changes and reduce the RMSE for most of the problems (except GTA7m and GTA8m test instances). Pareto optimal front of GTA7m and GTA8m is a single point at the initial few generation. This results in low correlation between the weight vector's element and its corresponding evaluated objective value when the algorithm output reaches the optimal point. For these two benchmarks, the inverse model is not triggered due to the low Spearman's correlation (decision procedure). Therefore, the detection test is not triggered throughout the optimization process. To circumvent this problem, lower confidence interval threshold (parameter α in Algorithm 1) should be used for dynamic MOP with degenerate Pareto optimal front geometrics [49].

V. Conclusion

This paper has proposed an MOEA to solve dynamic MOP by using a set of linear inverse models. The inverse model set is used to direct the optimization search toward promising decision space and model objective-to-decision mapping information for fitness landscape change detection purpose. By using the proposed method, the number of fitness evalutions for detection can be reduced.

References

[1] J. A. D. Atkin, E. K. Burke, J. S. Greenwood, and D. Reeson, "On-line decision support for take-off runway scheduling with uncertain taxi times at London Heathrow airport," *J. Sched.*, vol. 11, no. 5, pp. 323–346, 2008.

[2] A. Isaacs, V. R. Puttige, T. Ray, W. F. Smith, and S. G. Anavatti, "Development of a memetic algorithm for dynamic multi-objective optimization and its applications for online neural network modeling of UAVs," in *Proc. IEEE Int. Joint Conf. Neural Netw. World Congr. Comput. Intell. (IJCNN)*, Jun. 2008, pp. 548–554.

[3] S. H. Ngo, X. Jiang, V. T. Le, and S. Horiguchi, "Ant-based survivable routing in dynamic WDM networks with shared backup paths," *J. Supercomput.*, vol. 36, no. 3, pp. 297–307, 2006.

[4] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 40–53, Jan. 2014.

[5] R. Azzouz, S. Bechikh, and L. B. Said, "A multiple reference point-based evolutionary algorithm for dynamic multi-objective optimization with undetectable changes," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Beijing, China, 2014, pp. 3168–3175.

[6] R. Azzouz, S. Bechikh, and L. B. Said, "Multi-objective optimization with dynamic constraints and objectives: New challenges for evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf.*, Madrid, Spain, 2015, pp. 615–622.

[7] C. Li and S. Yang, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 556–577, Aug. 2012.

[8] S. B. Gee, K. C. Tan, V. A. Shim, and N. R. Pal, "Online diversity assessment in evolutionary multiobjective optimization: A geometrical perspective," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 542–559, Aug. 2015.

[9] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, Apr. 2015.

[10] A. Zhou, J. Sun, and Q. Zhang, "An estimation of distribution algorithm with cheap and expensive local search methods," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 807–822, Dec. 2015.

[11] B. Chen, W. Zeng, Y. Lin, and D. Zhang, "A new local search-based multiobjective optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 50–73, Feb. 2015.

[12] W. Hu and G. G. Yen, "Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 1–18, Feb. 2015.

[13] X. Cai, Z. Yang, Z. Fan, and Q. Zhang, "Decomposition-based-sorting and angle-based-selection for evolutionary multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, to be published.

[14] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, Oct. 2015.

[15] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 881–897, Jun. 2013.

[16] S. Das, A. Mandal, and R. Mukherjee, "An adaptive differential evolution algorithm for global optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 966–978, Jun. 2014.

[17] T. T. Nguyen and X. Yao, "Continuous dynamic constrained optimization—The challenges," *IEEE Trans. Evol. Comput.*, vol. 16, no. 6, pp. 769–786, Dec. 2012.

[18] K. Deb, N. U. B. Rao, and S. Karthik, "Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling," in *Evolutionary Multi-Criterion Optimization*. Heidelberg, Germany: Springer, 2007, pp. 803–817.

[19] R. Tinós and S. Yang, "A self-organizing random immigrants genetic algorithm for dynamic optimization problems," *Genet. Program. Evol. Mach.*, vol. 8, no. 3, pp. 255–286, 2007.

[20] B. Zheng, "A new dynamic multi-objective optimization evolutionary algorithm," in *Proc. IEEE ICNC*, Haikou, China, 2007, pp. 565–570.

[21] R. W. Morrison, *Designing Evolutionary Algorithms for Dynamic Environments*. New York, NY, USA: Springer, 2004.

[22] N. Mori, H. Kita, and Y. Nishikawa, "Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm," in *Parallel Problem Solving from Nature—PPSN V*. Heidelberg, Germany: Springer, 1998, pp. 149–158.

[23] A. Muruganantham, Y. Zhao, S. B. Gee, X. Qiu, and K. C. Tan, "Dynamic multiobjective optimization using evolutionary algorithm with Kalman filter," in *Proc. 17th Asia Pac. Symp. Intell. Evol. Syst. (IES)*, vol. 24. 2013, pp. 66–75.

[24] W. T. Koo, C. K. Goh, and K. C. Tan, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment," *Memetic Comput.*, vol. 2, no. 2, pp. 87–110, 2010.

[25] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization," in *Evolutionary Multi-Criterion Optimization* (LNCS 4403), S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds. Heidelberg, Germany: Springer, 2007, pp. 832–846.

[26] A. Ghosh, S. Tsutsui, and H. Tanaka, "Individual aging in genetic algorithms," in *Proc. Aust. New Zealand Conf. Intell. Inf. Syst.*, Nov. 1996, pp. 276–279.

[27] G. S. Hornby, "ALPS: The age-layered population structure for reducing the problem of premature convergence," in *Proc. 8th Annu. Conf. Genet. Evol. Comput.*, Seattle, WA, USA, 2006, pp. 815–822.

[28] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: Test cases, approximations, and applications," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 425–442, Oct. 2004.

[29] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, Feb. 2009.

[30] H. Richter, "Detecting change in dynamic fitness landscapes," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2009, pp. 1613–1620.

[31] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for noisy and dynamic environments," *Genet. Program. Evol. Mach.*, vol. 7, no. 4, pp. 329–354, 2006.

[32] 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.

[33] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 838–856, Dec. 2015.

[34] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput.*, vol. 9, no. 1, pp. 3–12, 2005.

[35] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 61–70, 2011.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[37] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[38] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.
[39] C. Spearman, "The proof and measurement of association between two things," *Amer. J. Psychol.*, vol. 15, no. 1, pp. 72–101, 1904.
[40] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*, vol. 104. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.
[41] C. Alippi, G. Boracchi, and M. Roveri, "A hierarchical, nonparametric, sequential change-detection test," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2011, pp. 2889–2896.
[42] S. Muthukrishnan, E. van den Berg, and Y. Wu, "Sequential change detection on data streams," in *Proc. 7th IEEE Int. Conf. Data Min. Workshops (ICDM)*, 2007, pp. 550–551.
[43] E. Ahmed, A. Clark, and G. Mohay, "A novel sliding window based change detection algorithm for asymmetric traffic," in *Proc. IFIP Int. Conf. Netw. Parallel Comput. (NPC)*, 2008, pp. 168–175.
[44] M. F. Triola, *Elementary Statistics*. Reading, MA, USA: Addison-Wesley, 2006.
[45] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.
[46] R. F. Coelho, "Probabilistic dominance in multiobjective reliability-based optimization: Theory and implementation," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 214–224, Apr. 2015.
[47] Q. Chen, C. Fu, and Q. Zhang, "On performance of decomposition-based MOEAs in noisy environment," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Sendai, Japan, 2015, pp. 3412–3417.
[48] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[49] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.
[50] Q. Zhang *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition," School EEE, Univ. Essex, Colchester, U.K. and Nanyang Technol. Univ., Singapore, Tech. Rep. CES-487, 2008.
[51] S. B. Gee, K. C. Tan, and H. A. Abbass, "A benchmark test suite for dynamic evolutionary multiobjective optimization," *IEEE Trans. Cybern.*, to be published.
[52] B. L. Welch, "The generalization of 'Student's' problem when several different population variances are involved," *Biometrika*, vol. 34, nos. 1–2, pp. 28–35, 1947.

Sen Bong Gee received the B.Eng. (Hons.) degree in electrical and computer engineering and the Ph.D. degree from the National University of Singapore, Singapore, in 2011 and 2016, respectively.

His current research interests include on evolutionary computation, artificial intelligence, and machine learning.

Kay Chen Tan (SM'08–F'14) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He has published over 100 journal papers, over 100 papers in conference proceedings, co-authored five books. His current research interests include computational and artificial intelligence, with applications to multiobjective optimization, scheduling, automation, data mining, and games.

Dr. Tan was a recipient of the 2012 IEEE Computational Intelligence Society Outstanding Early Career Award for his contributions to evolutionary computation in multiobjective optimization. He is currently an Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He was an Editor-in-Chief of the *IEEE Computational Intelligence Magazine* from 2010 to 2013. He serves as an Associate Editor/Editorial Board Member of over 15 international journals, such as the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, *Evolutionary Computation* (MIT Press), the *European Journal of Operational Research*, the *Journal of Scheduling*, and the *International Journal of Systems Science*.

Cesare Alippi (S'92–M'97–SM'99–F'06) received the degree (*cum laude*) in electronic engineering and the Ph.D. degree from the Politecnico di Milano, Milan, Italy, in 1990 and in 1995, respectively.

He is currently a Professor of information processing systems with the Politecnico di Milano and of cyber-physical and embedded systems with the Universita' della Svizzera Italiana, Lugano, Switzerland. He has been a Visiting Researcher with University College London, London, U.K., the Massachusetts Institute of Technology, Cambridge, MA, USA, ESPCI ParisTech, Paris, France, Institute of Automation, Chinese Academy of Sciences, Beijing, China, A*STAR, Singapore. He holds five patents, has published one monograph book, six edited books, and about 200 papers in international journals and conference proceedings. His current research interests include adaptation and learning in nonstationary environments and intelligence for embedded and cyber-physical systems.

Prof. Alippi was a recipient of the Gabor Award from the International Neural Networks Society and the IEEE Computational Intelligence Society Outstanding Transaction on Neural Networks and Learning Systems Paper Award in 2016, the IBM Faculty Award in 2013, the IEEE Instrumentation and Measurement Society Young Engineer Award in 2004, and the Knight of the Order of Merit of the Italian Republic in 2011. He is a Board of Governors Member of the International Neural Network Society, Board of Directors Member of the European Neural Network Society, the Vice-President education of the IEEE Computational Intelligence Society, and was an Associate Editor of the *IEEE Computational Intelligence Magazine*, the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENTS, and the IEEE TRANSACTIONS ON NEURAL NETWORKS, and a member and the Chair of many IEEE committees.