# Learning to Expand/Contract Pareto Sets in Dynamic Multi-objective Optimization with a Changing Number of Objectives

Ruan, Gan; Minku, Leandro; Menzel, Stefan; Sendhoff, Bernhard; Yao, Xin

*DOI:*
10.1109/TEVC.2024.3375751

# Learning to Expand/Contract Pareto Sets in Dynamic Multi-objective Optimization with a Changing Number of Objectives

Gan Ruan, Leandro L. Minku, *Senior Member, IEEE,* Stefan Menzel, Bernhard Sendhoff, *Fellow, IEEE,* and Xin Yao, *Fellow, IEEE*

***Abstract*—Dynamic multi-objective optimization problems (DMOPs) with a changing number of objectives may have Pareto-optimal set (PS) manifold expanding or contracting over time. Knowledge transfer has been used for solving DMOPs, since it can transfer useful information from solving one problem instance to solve another related problem instance. However, we show that the state-of-the-art transfer approach based on heuristic lacks diversity on problem with extremely strong bias and loses convergence on problems with multi-modality and variable correlation, after the number of objectives increases and decreases, respectively. Therefore, we propose a novel transfer strategy based on learning, called learning to expand and contract PS (denoted as LEC) for enhancing diversity and convergence after number of objective increases and decreases, respectively. It firstly learns potentially good directions for expansion and contraction separately via principal component analysis. Then, the most promising expansion and contraction directions are selected from their candidates according to whether they help diversity and convergence, respectively. Lastly, PS is learnt to be expanded and contracted based on these most promising directions. Comprehensive studies using 13 DMOP benchmarks with a changing number of objectives demonstrate that our proposed LEC is effective on improving solution quality, not only right after changes but also after optimization of different generations, compared to state-of-the-art algorithms.**

***Index Terms*—Evolutionary algorithms, Multi-objective optimization, Dynamic optimization, Changing objectives, Learning to optimize, Knowledge transfer**

## I. Introduction

In the field of dynamic multi-objective optimization, most existing work focus on solving dynamic multi-objective optimization problems (DMOPs) with changing position/shape of Pareto sets (PSs) and/or Pareto fronts (PFs) [1] [2]. However, DMOPs characterised by changes in the number of objectives (NObj) over time [3], also widely exist in project scheduling [4, 5], resource allocation [6, 7], etc. Such dynamics may cause the manifold where the Pareto-optimal solutions are located to expand or contract [3]. For instance, an increase in the NObj may result in the PS expanding along a given dimension of the solution space, incorporating new Pareto-optimal solutions that were not Pareto-optimal before the number of objectives increased. Similarly, a decrease in the NObj means that some solutions that were previously in the PS will not be Pareto-optimal anymore, potentially contracting the PS along a given dimension in the solution space. Considering this characteristic, knowledge transfer is a promising direction for solving DMOPs with a changing NObj [3, 8], since it can transfer useful information shared by PSs before and after changes to help the PS expansion or contraction.

Dynamic two archive evolutionary algorithm (DTAEA) [3] is a recent transfer approach for tackling changes in the NObj. However, DTAEA was computationally revealed in [8] to lose diversity when solving problems with complex features. These features are reflected in PF shapes (convex, discontinues and mixed shape of convex and concave) and fitness landscapes (nonseparability and deceptiveness). To address this limitation, knowledge transfer dynamic multi-objective evolutionary algorithm (KTDMOEA) [8] was proposed to enhance diversity via explicit heuristic rules for PS expansion or contraction. Computational studies demonstrated that KTDMOEA performed better than DTAEA on most problems.

However, we show that knowledge transfer via PS expansion in KTDMOEA is unable to increase quickly enough diversity on problems with extremely strong bias, since the heuristic used by KTDMOEA for setting the PS expansion direction fails to work for this kind of problems. We also show that, when decreasing the NObj, solutions transferred by KTDMOEA have poor convergence on problems with multi-modality and variable correlation. One reason is that the multi-modality and variable correlation cause solutions previously converged in the old environment to be far away from the true PF after decreasing the NObj. Another one is that solutions transferred along heuristically found contraction directions by KTDMOEA only aim at improving population diversity without any improvement on convergence.

Utilizing the right directions for expansion and contraction is essential for such kind of algorithm to perform well right after a change in the NObj. However, it is almost impossible to obtain correct expansion and contraction directions directly

This work was partially supported by the NSFC under Grant 62250710682, the Guangdong Provincial Key Laboratory under Grant 2020B121201001, the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X386, and the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186.

Gan Ruan, Leandro L.Minku and Xin Yao are with CERCIA, School of Computer Science, University of Birmingham, Edgbaston Birmingham B15 2TT, UK (e-mail: GXR847@student.bham.ac.uk, L.L.Minku@bham.ac.uk, xinyao@ln.edu.hk).

Stefan Menzel and Bernhard Sendhoff are with the Honda Research Institute Europe GmbH, 63073 Offenbach, Germany. (email: stefan.menzel@honda-ri.de, bernhard.sendhoff@honda-ri.de)

Xin Yao is also with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong, China. Part of this work was done while Gan Ruan and Xin Yao were with the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), Shenzhen 518055, China.

without doing any exploration of the new problem right after changes, since no prior knowledge about the new problem is available. Therefore, given the limitation of heuristic rules in finding PS expansion and contraction directions of KTDMOEA, we propose a novel approach that automatically learns candidate directions for expansion and contraction by mining the data feature of the PS before the change. Then, it selects the right expansion and contraction directions according to certain criteria from these candidate directions after exploration of the new problem beyond the evaluation of existing candidate solutions on the new set of objectives.

We aim to answer the following research questions (RQ):

RQ1 How to learn potentially good directions for PS expansion and contraction so as to enhance knowledge transfer right after changes?

RQ2 How does learned PS expansion and contraction help the optimization process after the changes?

To answer these RQs, we first propose to conduct Principal Component Analysis (PCA) on the PS of the previous problem to get eigenvectors. Then, when increasing the NObj, we learn candidate expansion directions from those directions perpendicular to these eigenvectors; when decreasing the NObj, we learn candidate contraction directions by regarding these eigenvectors as the candidate contraction directions. Last, we select the most promising expansion and contraction directions based on which candidate expansion helps diversity and which contraction helps convergence, so as to enhance the diversity and convergence of expanded and contracted PS, respectively. Experimental studies based on 13 benchmarks problems and 3 real-world problems for DMOPs with a changing NObj, demonstrate the effectiveness of our proposed approach in terms of solution quality both (1) right after changes, and (2) beyond the changes, after further optimization.

Our novel contributions are summarized as follows:

- Comprehensive experiments have been carried out on representative DMOPs with a changing NObj to understand the limitations of the state-of-the-art transfer algorithm KTDMOEA. Our analyses reveal that its knowledge transfer via PS expansion and contraction (1) still lacks diversity on the problem presenting extremely strong bias when increasing NObj, and (2) has poor convergence on problems with multi-modality and variable correlation when decreasing the NObj.
- A novel knowledge transfer-based strategy, called learning to expand and contract PS (denoted as LEC), is proposed, to enhance diversity and convergence for increasing and decreasing NObj, respectively. LEC expands and contracts PSs after learning promising expansion and contraction directions under the assistance of PCA.
- Systematic computational studies have been conducted to compare LEC with 4 algorithms on 13 benchmarks problems and 3 real-world problems for DMOPs with a changing NObj under different frequencies and types of changes in the NObj. Experimental results have shown that our algorithm is competitive in terms of solution quality right after changes and after further optimization against 4 compared algorithms.

The remainder of this paper is organized as follows. Section II presents related work on DMOPs with a changing NObj and evolutionary transfer optimization as well as the motivation of our proposal. LEC is elaborated in Section III. Section IV describes the experimental setup. The experimental results are in Section V. Section VI concludes this paper.

## II. Related Work and Motivation

This section first reviews related work on DMOPs with a changing NObj and evolutionary transfer optimization. Then, a computational investigation of the existing work KTDMOEA [8] is conducted to reveal its limitations on solving DMOPs with a changing NObj.

### A. DMOPs with a Changing NObj

*1) Problem Formulation:* In this paper, we focus on continuous DMOPs defined as follows:

$$\begin{cases} \min \ \mathbf{F}(\mathbf{x},t) = \big(f_1(\mathbf{x},t),\ldots,f_{m(t)}(\mathbf{x},t)\big)^T, \\ s.t. \ \ \mathbf{x}\in\Omega, \ t\in\Omega_t, \end{cases} \tag{1}$$

where $t$ is a discrete time defined as $t = \lfloor \frac{\tau}{\tau_t} \rfloor$, $\tau$ and $\tau_t$ refer to the iteration counter and the frequency of changes, respectively; $\mathbf{x} = (x_1, ..., x_n)$ is a candidate solution; $\Omega \subseteq R^n$ is the decision (variable) space; $\Omega_t \subseteq R$ is the time space; $\mathbf{F}(\mathbf{x}, t) : \Omega \times \Omega_t \to R^{m(t)}$ is the objective function vector that evaluates a candidate solution $\mathbf{x}$ at time $t$; $m(t)$ is the number of objectives at time $t$, where $m(t) \neq m(t+1)$ meaning that the NObj changes over time.

It is particularly important to obtain new good solutions right after changes, even though one may also be interested in good solutions after further optimization is carried out. The reason is that obtaining good solutions right after changes enables environmental changes with different frequency of change to be quickly responded to, especially under very high frequency of change, such that solutions can be quickly deployed before the next change.

*2) Related Work:* The possibly earliest work on DMOPs with a changing NObj is [9]. Even though this topic has been later mentioned in [10–12], few studies have been done on it until recently in [3], where a comprehensive investigation of the challenges of DMOPs with a changing NObj was conducted. Specifically, solutions tend to get crowded in certain areas, lacking diversity after increasing NObj, and solutions are not converged and nor diversified after decreasing NObj.

*a) DTAEA and KTDMOEA:* Facing the challenges, dynamic two archive evolutionary algorithm (DTAEA) [3] was proposed. DTAEA simultaneously maintains two complementary archives, convergence archive (CA) and diversity archive (DA), both right after changes and in the evolution process to focus on population convergence and diversity, respectively.Computational studies in [3] demonstrated that DTAEA was effective on DMOPs with a changing NObj presenting simple problem features. However, it has been experimentally shown in [8] that DTAEA lacks diversity right after changes when facing complex problem features in PF shape (discontinues, etc) and fitness landscape (nonseparability, etc).

To overcome DTAEA's limitation, KTDMOEA [8] was proposed to enhance diversity via PS expansion and contraction. The main idea is first to heuristically find certain directions for PS expansion and contraction, and then generate transferred solutions along the directions. KTDMOEA outperformed DTAEA on most problems with complex problems features mentioned before. However, as we will show in Section II-C, KTDMOEA still lacks diversity on the problem with extreme bias right after increasing the NObj, and loses convergence on problems with multi-modality and variable correlation right after decreasing the NObj. Our analysis in Section II-C shows that the heuristic rules for PS expansion and contraction do not work well for these problems.

*b) Solution Generation in PS Expansion of KTDMOEA:* This paragraph describes the solution generation method based on a population and a set of directions from PS expansion of KTDMOEA [8] after finding the expansion directions. Given a population $Pop$ with size $N$ and a set of directions $\mathbf{D}$ with size $N_D$, to generate a new population based on $Pop$ and $\mathbf{D}$, some solutions are firstly selected from $Pop$ as base solutions (denoted as $\mathbf{P}_{base}$) with the number of $N_{base}$. Then, for each solution $\mathbf{x}$ in $\mathbf{P}_{base}$ and each direction $\mathbf{D}_i$ in $\mathbf{D}$, generate one new solution $\mathbf{x}_{new}$ using the following equation:

$$\mathbf{x}_{new} = \mathbf{x} + ss * rand() * \mathbf{D}_i, (i = 1, ..., N_D), \quad (2)$$

where $ss$ is the step size whose calculation is presented in the next paragraph and $rand()$ samples a number uniformly at random from (0, 1].

The calculation of $ss$ should follow the criterion that solutions (i.e., $\mathbf{x} + ss * \mathbf{D}_i$) generated from any solution $\mathbf{x}$ and one direction $\mathbf{D}_i$ are within the bound of each decision variable and they should reach the boundary of the search space. Bearing this criterion in mind, the calculation of $ss$ was designed in [8]. Given a solution $\mathbf{x}$ and one direction $\mathbf{D}_i$, suppose $para^k$ is the value that makes the $k$-th variable of the generated solution reach the boundary of this variable. Therefore, each $para^k$ can be calculated according to whether the $k$-th element of $\mathbf{D}_i$ is positive or negative, via the following equation:

$$para^k = \begin{cases} \frac{upper^k - x^k}{D_i^k}, & D_i^k > 0 \\ \frac{lower^k - x^k}{D_i^k}, & D_i^k < 0 \end{cases} \quad (3)$$

where $upper^k$ and $lower^k$ are the upper bound and lower bound of the $k$-th dimension of the decision space; $x^k$ is the $k$-th decision value of $\mathbf{x}$; $D_i^k$ is the value of the direction $\mathbf{D}_i$ at the $k$-th dimension. In order to ensure each generated solution is located within the region, $ss = \min_{k=1,...,n} para^k$, where $n$ is the dimension of the decision space.

### B. Evolutionary Transfer Optimization

People have recently exploited the use of knowledge transfer in evolutionary computation to tackle multitasking optimization problems [13–16] and dynamic multi-objective optimization problems [17]. Specifically, knowledge transfer is able to learn useful knowledge from related problem instances to solve the targeted problem instance [13, 14]. However, evolutionary multi-tasking optimization (EMT) [13–16] differs from our scenario here. The reason is that EMT considers solving multiple tasks simultaneously, while our work considers solving different problem instances sequentially as the environment, in particular the NObj, changes over time. At any given time, we solve only one problem instance, not multiple ones.

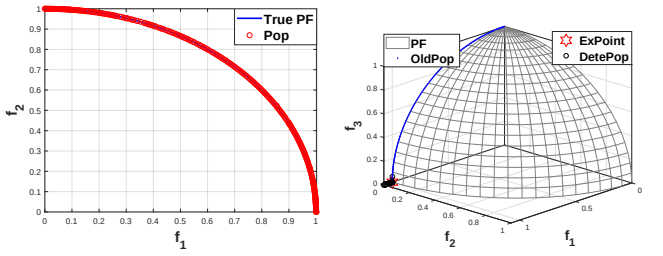
For dynamic multi-objective optimization, knowledge transfer was shown to help predicting good solutions for the next change based on previously optimized solutions and many outstanding algorithms have beeen proposed [18–21]. However, these knowledge transfer-based DMOEAs have not considered and cannot tackle DMOPs with a changing NObj, since they were designed to track the changing position or shape of PSs or PFs rather than expand or contract PS/PF. An increase/decrease in the NObj means that new solutions are added/removed from the PS rather than resulting in the formation of a trend in the changing position of the PS. Therefore, algorithms able to expand/contract the PS [8] may be more suitable, which can be supported by experimental results of KTDMOEA [8]. In general, it is always very challenging to decide when and how to transfer in DMOPs [22, 23], since bad answers to these questions would result in worse solutions by transfer than without transfer.

### C. A Computational Investigation of KTDMOEA

Even though KTDMOEA has been computationally demonstrated in [8] to be effective on DMOPs with a changing NObj, it has limitations on some problems. Specifically, we show in this section that it lacks diversity on the problem with extremely strong bias right after increasing the NObj, and loses convergence on problems with multi-modality and variable correlation right after decreasing the NObj. To analyze the limitations of KTDMOEA, F4 with extremely strong bias and F3 with multi-modality and variable correlation, from the DTLZ suite [3], are selected as examples to conduct an experimental investigation of how the quality of solutions transferred by KTDMOEA is after the NObj changes. At the same time, another problem F2 which has the same PF shape as F3-F4 and no problem feature in the fitness landscape was also selected from the DTLZ suite [3] as a base comparison.

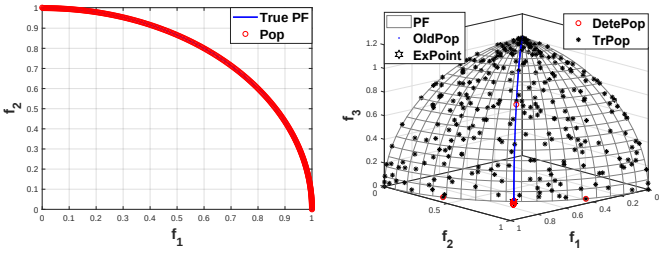
Specifically, when increasing NObj, problems F2 and F4 are set as bi-objective problem and then given 1000 generations by KTDMOEA to make population approximate true PF well before increasing the NObj from 2 to 3. Similarly, when decreasing the NObj, problems F2 and F3 are set as a tri-objective problem and then given 1000 generations by KTDMOEA to make population approximate true PF well before decreasing the NObj from 3 to 2. In these two examples, other parameters are the same as in KTDMOEA paper [8].

*1) Analyses of Increasing NObj:* Figs. 1 and 2 show the distribution of approximated population given 1000 generations on bi-objective problems and solutions transferred by PS expansion of KTDMOEA at the first generation right after increasing NObj from 2 to 3 on F4 and F2, respectively. As shown in Figs. 1(b) and 2(b), the extreme point ('ExPoint') is firstly selected from the old population ('OldPop'). A detective population ('DetePop') is then randomly generated around the extreme point in the solution space. Solutions dominated by




(a) Approximation on bi-F4 (b) After increasing one NObj

Fig. 1. (a) Distribution of approximated population on bi-F4; (b) Distribution of the old population ('OldPop'), i.e., reevaluating 'pop' obtained in (a) on the tri-F4, the extreme point ('ExPoint'), the detective population ('DetePop'), all used to find the expansion directions. Transferred solutions by KTDMOEA on F4 are still on the curve of 'OldPop', thus not being plotted to avoid solutions duplication, and failing to spread through the PF.




(a) Approximation on bi-F2 (b) After increasing one NObj

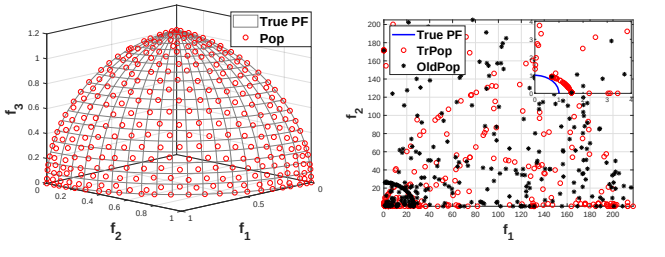
Fig. 2. (a) Distribution of approximated population on bi-F2; (b) Distribution of the old population ('OldPop'), i.e., reevaluating 'pop' obtained in (a) on the tri-F2, the extreme point ('ExPoint'), the detective population ('DetePop'), all used to find the expansion directions in the objective space, and transferred population('TrPop') via PS expansion in KTDMOEA.

the old population and solutions located in the same subspace as any solution of the old population will be deleted from the detective population. The extreme point and the remaining solutions in the detective population are connected to form the expansion directions by regarding the extreme point as the start of these directions.

However, when observed in the objective space (Fig. 1(b)), there will be no solution in the detective population after removing solutions dominated by the old population and located in the same subspace as any solution of the old population, resulting in no expansion directions to be found. As a comparison, there are still two solutions in the detective population for F2 without the feature of strong bias (Fig. 2(b)), forming expansion directions together with the extreme point.

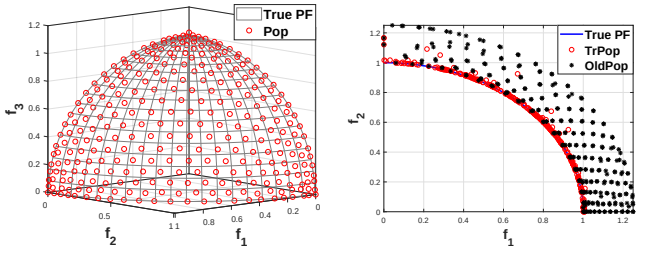
The reason is that the feature of strong bias in the fitness landscape of F4 hinders solutions with diversity to be generated in the detective population. More specifically, for most possible values of each variable ($x_i$) within the range [0,1], the intermediate function (i.e., $y = x_i^{100}$) containing the feature of strong bias which forms the objective function of F4 maps them to the value of nearly zero. Therefore, most randomly generated solutions around the extreme point would have the same objective value as that of the extreme point, appearing duplicated in the extreme point, as shown in 1(b).

*2) Analyses of Decreasing NObj:* Figs. 3 and 4 show the distribution of approximated population given 1000 generations on tri-objective problems and solutions transferred by PS contraction of KTDMOEA at the first generation right after decreasing NObj from 3 to 2 on F3 and F2, respectively. We can find (1) from Fig. 3(a) that the population is converged on




(a) Approximation on tri-F3 (b) After decreasing one NObj

Fig. 3. Distribution of approximated population on tri-F3 and solutions transferred by PS contraction of KTDMOEA at the first generation right after decreasing NObj from 3 to 2 on F3.




(a) Approximation on tri-F2 (b) After decreasing one NObj

Fig. 4. Distribution of approximated population on tri-F2 and solutions transferred by PS contraction of KTDMOEA at the first generation right after decreasing NObj from 3 to 2 on F2.

F3 when NObj = 3; and (2) from Fig. 3(b) that the population is far away from the true PF after decreasing the NObj from 3 to 2. This phenomenon does not occur on F2 (Fig. 4). The comparison between F2 and F3 shows that it is the problem feature difference of F2 and F3 (multi-modality and variable correlation) that causes the phenomenon difference.

To simplify presentation, in Fig. 3(b) and Fig. 4(b), the non-dominated set of the old population (black diamonds) is denoted as **NDSO**; red circles near the $f_2$ axis are denoted as **SpTrS**; red circles between two black diamonds are denoted as **EvTrS**. The heuristic rule used in KTDMOEA's PS contraction only aims at improving population diversity in terms of spread and even distribution. For spread, KTDMOEA first (1) selects one extreme point and its nearest solution from **NDSO**, and (2) generates spread solutions (i.e., **SpTrS**) along the inverse direction from the extreme point and its nearest solution. For even distribution, evenly distributed solutions (i.e., **EvTrS**) are generated by selecting any two solutions from **NDSO**.

We can find from Fig. 3(b) that transferred solutions (i.e., red circles) do not have better convergence than the old population (i.e., black diamonds). Therefore, the heuristic rule in the PS contraction of KTDMOEA is unable to increase population convergence when decreased NObj degenerates population convergence on problems with multi-modality and variable correlation.

## III. Learning to Expand and Contract PSs (LEC)

Given the limitations of heuristic rules in KTDMOEA when expanding and contracting PS regarding unable to (1) determine expansion directions for problems with extremely strong bias and (2) improve population convergence for problems with variable correlation and multi-modality, it is essential to learn how to expand and contract the PS. Therefore, in this section, we present our proposed knowledge transfer-based

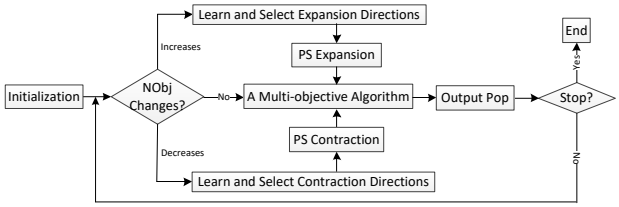
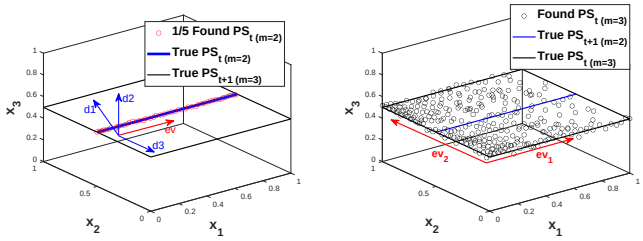



Fig. 5. Flow chart of LEC.




(a) Learning expansion directions (b) Learning contraction directions

Fig. 6. Method of learning the candidate expansion and contraction directions for increasing and decreasing NObj, respectively: (a), **ev** is the eigenvector of the found $PS_{m=2}$ (converged) and **d**1 to **d**3 are three directions that are perpendicular to $PS_{m=2}$, (i.e., **ev** * $\mathbf{d}_i$ = 0, where $i = 1, 2, 3$); **d**1 to **d**3 are the three candidate expansion directions (b), **ev1** and **ev2** are two eigenvectors of found $PS_{m=3}$, which are also the candidate contraction directions.

strategy, i.e., learning to expand and contract PSs (LEC), answering RQ1. This strategy contains three procedures that need to be carried out sequentially: (1) learning candidate expansion or contraction directions under the assistance of PCA (Section III-A), (2) selecting the most promising directions among these candidate expansion or contraction directions according to which candidate expansion direction improves diversity and which contraction direction improves convergence (Section III-B), and (3) PS expansion or contraction (Section III-C).

Note that our proposed LEC is conducted right after changes. After LEC is conducted, the population can be further optimized by any existing algorithm. Fig. 5 illustrates the flowchart of a DMOEA using our proposed LEC strategy.

### A. Learning Candidate Expansion and Contraction Directions

*1) Learning Candidate Expansion Directions:* This section aims at learning candidate expansion directions from PS before the change under the assistance of PCA, as illustrated in Fig. 6(a). Note this figure is just drawn to illustrate the process of learning the expansion directions; the specific solutions in real problems may be different. As shown in the figure, suppose the red circle points are the found $PS_{m=2}$ before the change, and **ev** is the eigenvector obtained by conducting PCA on $PS_{m=2}$. Then, $d1$ to $d3$ are three of many candidate expansion directions, which are perpendicular to $PS_{m=2}$.

We assume that $PS_t$ is a subset of $PS_{t+1}$ when the NObj increases, by following Theorem 1 of [3]. Hence, $PS_{m=2}$ manifold is a subset of $PS_{m=3}$. Assume $PS_{m=3}$ manifold is a plane, then $PS_{m=2}$ manifold could be a segment line in this plane, where the $PS$ manifold is the location of Pareto-optimal solutions in the solution space. However, we acknowledge that in the most generic case the relationship between PS and PF may change when the NObj changes, such that $PS_t$ is not always a subset of $PS_{t+1}$ when the NObj increases.

It is well-known in geometry that two intersecting lines in space determine a plane. Therefore, there are infinite lines in the PS manifold of tri-objective problem that have intersections with the eigenvector of the $PS_{m=2}$, since $PS_{m=3}$ manifold is the expansion of $PS_{m=2}$ manifold along certain directions. However, there is only one unit vector and its inverse vector on $PS_{m=3}$ that are perpendicular to the eigenvector of the $PS_{m=2}$. Therefore, only the directions perpendicular to the $PS_{m=2}$ are selected as the candidate expansion directions, reducing the number of candidate expansion directions.

**Algorithm 1: Learning Candidate Expansion Directions**

**Input:** PS at time step $t$ before the change ($\mathbf{PS}_t$); re-evaluated solutions of $\mathbf{PS}_t$ on the new problem after the change (i.e., time step $t+1$) **OldP**; population size $N$; the NObj for the old problem before the change $m(t)$; dimension of the problem $n$;

**Output:** Set of candidate expansion directions $\mathbf{D}_p$ or NULL.

1 Conduct principal component analyses on $\mathbf{PS}_t$ to get $m(t)$-1 eigenvectors **ev**$s$;
2 Initialize a set of $N$ vectors $\mathbf{D}_p = (\mathbf{D}_p^1, ..., \mathbf{D}_p^j, ..., \mathbf{D}_p^N)$ with all elements ($d_j^1$ to $d_j^n$) in $\mathbf{D}_p^j$ as null value;
3 Form $N$ system of $n$-variable homogeneous linear equations: $\mathbf{D}_p^j * \mathbf{ev}_i = 0$ $(i = 1, ..., m(t)-1)$;
4 Sample $N$ $(n - m(t) + 1)$-d vectors as **Vec** using Latin hypercube sampling with all elements within [0,1] ;
5 **for** *j = 1 to N* **do**
6 Assign $\mathbf{Vec}_j$ to the first $n - m(t) + 1$ elements of $\mathbf{D}_p^j$, i.e., $d_j^k = \mathbf{Vec}_j^k$ $(k = 1, ..., n - m(t) + 1)$ ;
7 Solve the $(m(t) - 1)$-variable equation set in line 3 and get value of $d_j^{n-m(t)+2}$ to $d_j^n$ in $\mathbf{D}_p^j$ ;
8 Convert $\mathbf{D}_p^j$ to a unit vector;
**end**
**Return** $\mathbf{D}_p$.

LEC will create a set of size $N$ containing evenly sampled perpendicular vectors to the $PS$ of the previous environment as the candidate expansion directions. The framework of learning candidate expansion directions is in Algorithm 1. Given the Pareto optimal solution set at the time step t ($\mathbf{PS}_t$) before the change, the first step of learning the candidate expansion directions is conducting PCA on $\mathbf{PS}_t$, to get the eigenvectors (**ev**$s$), as shown in line 1 of Algorithm 1. The number of the eigenvectors is $m(t) - 1$, as the $PS$ of a continuous multiobjective optimization problem with $m(t) - 1$ is a piecewise continuous $m(t) - 1$-D manifold [3] [24]. In line 2, suppose $\mathbf{D}_p = (\mathbf{D}_p^1, ..., \mathbf{D}_p^j, ..., \mathbf{D}_p^N)$ is a set of $N$ vectors, which are perpendicular to these eigenvectors (i.e., $\mathbf{ev}_i$), where each vector in $\mathbf{D}_p$ is initialized with null values, i.e., $\mathbf{D}_p^j = (d_j^1, ..., d_j^n) = (null, ..., null)$, where $d_j^1$ to $d_j^n$ are the variables we want to obtain from steps 3 to 7.

Afterwards, a system of homogeneous linear equations is constructed in line 3 by the fact that the product of two orthogonal vectors (i.e., **ev** and candidate expansion directions $\mathbf{D}_p^j$) is zero. Note the number of homogeneous linear equations is $m(t) - 1$, since there are only $m(t) - 1$ eigenvectors (i.e.,

**Algorithm 2: Learning Candidate Contraction Directions**

**Input:** PS at time step $t$ before the change ($\mathbf{PS}_t$)
**Output:** Set of the learnt candidate contraction directions $\mathbf{C}_{con}$ or NULL.
1 Conduct principal component analyses on $\mathbf{PS}_t$ to get $m(t)$-1 eigenvectors as $\mathbf{ev}_i$, where $i = 1, ..., m(t) - 1$;
2 Put these eigenvectors, i.e., $\mathbf{ev}_1$, ..., $\mathbf{ev}_{m(t)-1}$ in $\mathbf{C}_{con}$;
**Return** $\mathbf{C}_{con}$.

$\mathbf{ev}_i$s, $i = 1, ..., m(t) - 1$). Now, there are only $m(t) - 1$ homogeneous linear equations while $n$ variables (i.e., $d_i^1$ to $d_i^n$) to get. It is known that there are infinite solutions for this set of linear equations. To support the creation of directions that evenly cover the space, we create a set **Vec** containing $N$ evenly sampled vectors with dimension $n - m(t) + 1$ within a unit hypercube. To make solutions of the $m(t) - 1$ homogeneous linear equations with $n$ variables finite, the values of $n-m(t)+1$ variables must be assigned beforehand, which are from **Vec**.

The 'for' loop from lines 5 to 8 calculates values of $d_j^1$ to $d_j^n$ for $\mathbf{D}_j$ to construct $N$ vectors ($\mathbf{D}_p^1$ to $\mathbf{D}_p^N$) and converts them to unit vectors as candidate expansion directions. Specifically, line 6 puts the values of the $j$-th sampled vector ($\mathbf{Vec}_j$) into the first $n-m(t)+1$ positions of the $j$-th vector of $\mathbf{D}_p$ (i.e., $\mathbf{D}_p^j$). Then, line 7 calculates values of $d_j^{n-m(t)+2}$ to $d_j^n$ after putting $d_j^1$ to $d_j^{n-m(t)+1}$ in the system of $m(t)-1$ homogeneous linear equations. Now, all values of $d_j^1$ to $d_j^n$ in $\mathbf{D}_p^j$ are obtained. Then line 8 converts $\mathbf{D}_p^j$ to a unit vector. Once all $N$ vectors in $D_p$ are created, they are returned as the candidate expansion directions.

*2) Learning Candidate Contraction Directions:* This section aims at learning candidate contraction directions from the PS that was available right before the change based on PCA, as illustrated in Fig. 6(b). Note this figure is just drawn to demonstrate the process of learning PS contraction directions, the specific cases in real problems may be different. As shown in Fig. 6(b), black points are the found $PS_{m=3}$ before the change; $\mathbf{ev}_1$ and $\mathbf{ev}_2$ are two eigenvectors of $PS_{m=3}$, which are regarded as the candidate contraction directions.

The method for learning candidate contraction directions is described in Algorithm 2. Given a set of Pareto optimal solutions at the time step t ($\mathbf{PS}_t$) before the change, the first step of learning the candidate contraction directions is conducting PCA on $\mathbf{PS}_t$, to get the eigenvectors ($\mathbf{ev}s$), as shown in line 1 of Algorithm 2. Then, in line 2, regard these eigenvectors as the candidate contraction directions and put them in $\mathbf{C}_{con}$ to return.

### *B. Selecting Most Promising Expansion and Contraction Directions Among the Generated Candidate Directions*

*1) Selection of Expansion Directions to Improve Diversity:* The selection criterion of the most promising expansion directions is to see which candidate direction does not result in dominated solutions by any solution in reevaluated $PS$ of the previous environment on the new problem. Since increasing the NObj may lead to the expansion of PF [3], it is intuitive that population diversity increases after the expansion of PF. Therefore, solutions generated along the most promising expansion directions will improve population diversity.

**Algorithm 3: Most Promising Expansion Directions Selection**

**Input:** Candidate expansion directions $\mathbf{D}_p$;
re-evaluated solutions of $\mathbf{PS}_t$ on the new problem after the change (i.e., time step $t + 1$)
**OldP**; population size $N$;
**Output:** Set of the most promising expansion directions $\mathbf{D}_{exp}$ or NULL.
1 Randomly sample a solution **x** from **OldP**;
2 **for** *i = 1 to* $N$ **do**
3 Generate a solution **y** based on **x** and $\mathbf{D}_p^i$ using Equation (2);
4 Evaluate **y** and compare it to all solutions in **OldP**;
5 **if** *no solution in* ***OldP*** *dominating* ***y*** **then**
Put $\mathbf{D}_p^i$ into $\mathbf{D}_{exp}$;
**end**
**end**
**Return** $\mathbf{D}_{exp}$.

The process of selecting most promising expansion directions from their candidates is in Algorithm 3. Given the set of candidate expansion directions $\mathbf{D}_p$ and re-evaluated solutions of $\mathbf{PS}_t$ on the new problem after the change (i.e., time step $t + 1$) **OldP**, a solution **x** is firstly randomly sampled from **OldP** in line 1. For each candidate direction in $\mathbf{D}_p$, generate a solution **y** based on **x** and $\mathbf{D}_p^i$ using Equation (2). Then, evaluate **y** and compare it to all solutions in **OldP**; if **y** is not dominated by any solution in **OldP**, then add this candidate direction to the set of expansion directions $\mathbf{D}_{exp}$.

*2) Selection of Contraction Directions to Improve Convergence:* The selection criterion of the most promising contraction directions is to see which candidate direction results in solutions with better convergence. Since decreasing the NObj may lead to the contraction of PF/PS [3], solutions used to be optimal may not be optimal after the change. Therefore, solutions moving from positions of past optimal solutions to the positions of new optimal solutions along the most promising contraction direction will improve population convergence to help them reach the new, contracted PS/PF.

The process of selecting the most promising expansion directions from their candidates is in Algorithm 4. Given the set of candidate expansion directions $\mathbf{C}_{exp}$ and re-evaluated solutions of $\mathbf{PS}_t$ on the new problem after the change (i.e., time step $t + 1$) **OldP**, a solution **x** is firstly randomly sampled from **OldP** in line 1 of Algorithm 4. Since PS of a continuous $m(t)$- objective MOP is an ($m(t)$ - 1)-dimensional piecewise continuous manifold in the decision space, there are ($m(t)$ - 1) candidate expansion directions in $\mathbf{C}_{exp}$. In order not to consume too much function evaluations, we make total generated solutions to find the most promising contraction directions not exceed the population size $N$. Therefore,

**Algorithm 4: Most Promising Contraction Directions Selection**

**Input:** Candidate contraction directions $\mathbf{C}_{con}$; re-evaluated solutions of $\mathbf{PS}_t$ on the new problem after the change (i.e., time step $t+1$) **OldP**; population size $N$; the NObj for the old problem before the change $m(t)$
**Output:** Set of the most promising contraction directions $\mathbf{D}_{con}$ or NULL.

1 Randomly sample a solution **x** from **OldP**;
2 **for** *i = 1 to* $m(t)-1$ **do**
3 Generate $\lfloor N/(m(t)-1) \rfloor$ solutions $\mathbf{y}_j$ $(j = 1, ..., \lfloor N/(m(t)-1) \rfloor)$, based on **x** and $\mathbf{C}^i_{con}$, using Equation (2);
4 Put these generated solutions ($\mathbf{y}_j$) into **Y** ;
5 Evaluate these generated solutions and compare them to **x**;
6 **if** $\exists \mathbf{y}_j \in$ ***Y*** *dominates* ***x*** **then**
Put $\mathbf{C}^i_{con}$ into $\mathbf{D}_{con}$.
**end**
**end**
**Return** $\mathbf{D}_{con}$.

for each candidate contraction direction in $\mathbf{C}_{con}$, generate $\lfloor N/(m(t)-1) \rfloor$ solutions $\mathbf{y}_j$ based on **x** and $\mathbf{C}^i_{con}$ using Equation (2). Later on, in line 4, these generated solutions (i.e., $\mathbf{y}_j$) are put into the set **Y**. These generated solutions are evaluated and compared to **x**; if there exist one solution $\mathbf{y}_j \in \mathbf{Y}$ dominating **x**, then put $\mathbf{ev}_i$ into $\mathbf{D}_{con}$.

### C. PS Expansion and Contraction

After determining the most promising expansion and contraction directions, the next step is to expand and contract PSs for enhancing diversity and convergence when increasing and decreasing NObj, respectively. Note that the enhanced diversity and convergence by our proposed PS expansion and contraction are in the objective space rather than the decision space. In particular, whenever we use the term diversity in this paper, we are referring to population diversity in the objective space. If no expansion or contraction directions are found based on the procedures from Sections III-A and III-B2, the population from the old environment is directly copied as the initial population of the new environment.

*1) Diversity-Enhancing PS Expansion:* The process of expanding PS to enhance diversity after determining the most promising expansion directions is the same as that of KTD-MOEA [8]. The main idea is to evenly select some solutions from the $PS$ before the change as the base solutions, and then generate a new population based on these base solutions and the learnt expansion directions. Specifically, after determining the expansion directions $\mathbf{D}_{exp}$, we generate $\theta$ (a parameter set by the user) solutions based on each evenly selected solution from **OldP** along each direction from $\mathbf{D}_{exp}$, following Equation (2). If $\mathbf{P}_{exp}$ is not full, just evenly select solutions from $\mathbf{PS}_t$ in the objective space. Note that in our proposed LEC, the learning-based PS expansion is conducted right after increasing the NObj. The expanded PS $\mathbf{P}_{exp}$ is regarded as the initial population for the optimization of the new problem.

*2) Convergence Enhancing PS Contraction:* The process after determining the most promising contraction directions is to contract the PS along these learnt contraction directions for enhancing convergence. The main idea is to randomly select some solutions from the $PS$ before the change as the base solutions, and then generate a new population based on these base solutions and the learnt contraction directions. In particular, we randomly select $\lambda$ solutions from $PS_t$ and put them in the set $P_{base}$. The size of the learnt contraction directions $\mathbf{D}_{con}$ is $N_{con}$. Then, to make the number of generated solutions from each selected solution along each contraction direction equal and generated solutions not exceed the population size $N$, we generate $N_g = \lfloor \frac{N}{N_{con}*\lambda} \rfloor$ solutions based on each solution in $\mathbf{P}_{base}$ along each contraction direction in $\mathbf{D}_{con}$, following Equation (2) in Section II-A2b, and put them into $\mathbf{P}_g$. Afterwards, we combine evaluated $\mathbf{P}_g$ and $\mathbf{PS}_t$, and select solutions with the number of $N$ as the contracted PS $\mathbf{P}_{con}$ using nondominated sorting and decomposition-based density estimation from [8]. Note that in our proposed LEC, the learning-based PS contraction is conducted right after decreasing the NObj. The contracted PS $\mathbf{P}_{con}$ is regarded as the initial population for the optimization of the new problem.

### D. Time Complexity Analysis

The time complexity of LEC is discussed in this subsection for one generation. When increasing or decreasing the NObj, the PS expansion or contraction is evoked. In the case of increasing NObj, the main complexity of PS expansion is the cost of PCA (line 1 of Algorithm 1) and of solving systems of homogeneous linear equations with $m(t)-1$ variables (line 7 within the 'for' loop of Algorithm 1) for learning candidate expansion directions. The former consumes $O(m(t)^2)$ and the latter costs $O(Nm(t)^2)$, where $m(t)$ is the number of objectives at time step $t$. As for the case of decreasing the NObj, the PCA in line 1 of Algorithm 2 cost $O(m(t)^2)$; while the nondominated sorting and density estimation (Section III-C2)) consume $O(N^2)$ and $O(N|W(t)|)$, respectively, where $W(t)$ is the number of weight vectors at time step $t$. As LEC utilizes the CA update mechanism from DTAEA to update the population, the population update in LEC costs $O(N^2)$ comparisons. In summary, assuming that $N > W(t)$ and $N > m(t)$, complexity of LEC in one generation is max($O(Nm(t)^2)$, $O(N^2)$).

## IV. Experimental Setup

This section shows the experimental design to verify whether LEC successfully answers RQ1, to answer RQ2, and to reveal whether existing DMOEAs for DMOPs are able to deal with a changing NObj despite not being designed to do so (these are important baselines).

### A. Benchmark Problems

Two suites of multi-objective optimization test problems DTLZ [25] and WFG [26] are modified to be DMOPs with

a changing NObj. Four DMOPs with a changing NObj from DTLZ1-DTLZ4 are renamed as F1-F4 following [3]. These two suites are used to verify LEC's ability to deal with problems with different problem features. Descriptions of such features can be found in Section I of our Supplementary File.

The sequence of changes for these benchmark problems is: the NObj firstly increases from 2 to 7 one by one and then decreases from 7 to 2 one by one as [3], which is shown below:

$$m(t) = \begin{cases} 3, & t\text{=}1 \\ m(t-1)+1, & t \in [2,5] \\ m(t-1)-1, & t \in [6,10] \end{cases} \quad (4)$$

### B. Compared Algorithms

Four algorithms are selected for the comparison, so as to verify the performance of our proposal against the state-of-the-arts. Two popular and state-of-the-art DMOEAs including DNSGA-II [27] and MOEA/D-KF [28] specifically designed for DMOPs with changing shapes and/or positions of PS and/or PF are selected as the baselines. These two algorithms are selected as the baselines due to their popularity and good performance on solving DMOPs with changing shapes and/or positions of PS and/or PF. To verify whether LEC answers the research questions mentioned in Section I, as two of recently developed algorithms targeted for handling changes in the NObj, DTAEA [3], and KTDMOEA [8] are also chosen to be compared. Note that the optimization algorithm used within our LEC is the same as the one used in KTDMOEA due to its competitive results obtained in [8] on these problem instances. Section II of the Supplementary File presents detailed descriptions of these algorithms.

### C. Parameter Settings

The parameters of the algorithms are set as follows:

- Population size: $popsize$ = 300, the same as that of DTAEA and KTDMOEA, $\theta$ in KTDMOEA [8] is set as 2. The impact of $\theta$ on KTDMOEA's performance was analyzed in KTDMOEA paper [8].
- Number of solutions $\lambda$ selected from the $\mathbf{PS}_t$ in learning to contract PS is set as $popsize/20$ based on experience.
- Several different frequencies of change ($\tau_t$) are investigated to understand their effect on the algorithm: $\tau_t$ is set as 50 and 100 and 200; note a small value of $\tau_t$ means high frequency of change.
- When a change occurs, we may need a solution to be adopted as quickly as possible. So, one may wish to adopt the transferred solutions right after the change happens. However, while this solution is being adopted in practice, it is possible to continue running the optimization for a number of generations (i.e., $\tau_t$s), so that the population can converge before the next change happens, enabling good knowledge transfer for the next change. Therefore, we could output the optimized solutions before the end of running the optimization to see which algorithm is able to quickly output optimized solutions with better quality. Following this line, we set different generations of outputting optimized solution (denoted as $gap$) for different $\tau_t$s: 5, 25 and 50 for $\tau_t$ = 50; 5, 25, 50 and 100 for $\tau_t$ = 100; 5, 25, 50, 100 and 200 for $\tau_t$ = 200;
- All algorithms run 31 times independently, also the same as in DTAEA's work [3].
- 1000 generations are given to each algorithm before the first change so that the population before the change can converge.
- The crossover probability is $p_c$ = 1.0 and its distribution index is $\eta_c$ = 20. The mutation probability was $p_m$ = 1/n (where n denotes the number of decision variables) and its distribution $\eta_m$ = 20. These parameters were chosen because of their good performance on solving continuous problems, which have been analyzed in [29] and [25].
- The neighbourhood size and the number $n_r$ of solutions allowed to replace in MOEA/D were set to 20 and 2, respectively, as in the original paper [30].

### D. Performance Metrics

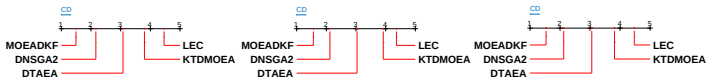
- Mean Hypervolume (MHV) [3] is the mean Hypervolume (HV) [31] of obtained solution sets by an algorithm at a set of discrete time steps. HV comprehensively measures the convergence and diversity of solution sets; the larger the better.
- Mean Generational Distance (MGD) [32] [2] is the mean Generational Distance (GD) [33] of obtained solution sets by an algorithm at a set of discrete time steps. GD evaluates the convergence of obtained solution sets; the smaller the better.

## V. Experimental Results and Analyses

This section presents the experimental results to answer RQ1–2 and verify whether the existing DMOEAs for solving DMOPs with fixed NObj are able to tackle DMOPs with a changing NObj. We also investigate the impact of different NObj changing sequences and algorithm parameters.

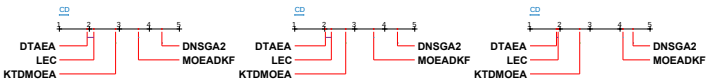
The metrics MHV and MGD are used to measure the quality of the solutions at the first generation after the change and in the last generation before the next change by the five algorithms. MHV (MGD) is the mean HV (GD) of obtained solution sets by an algorithm at a set of discrete time steps. To check if there is significant difference between the proposed LEC and other algorithms across all problem instances in general, Friedman and Nemenyi statistical tests [34] with a significance level 0.05 are adopted across all benchmark problems regarding the MHV and MGD of the five compared algorithms. The larger (smaller) the Friedman ranking of MHV (MGD), the better the algorithm. The Wilcoxon rank sum test at the 5% significance level is utilized to check if there is significant difference between the LEC and other algorithms on each individual problem and frequency of change.

For Friedman and Nemenyi statistical tests, the MHV and HGD that each algorithm gets on one problem at each independent run is regarded as an observation of the test, for all frequencies of change. Therefore, there are 403 (13 problems, 31 independent runs) observations for each algorithm for these tests. For Wilcoxon rank sum test, it is implemented on each benchmark problem regarding each metric of five compared




(a) $\tau_t$ = 50 (b) $\tau_t$ = 100 (c) $\tau_t$ = 200

Fig. 7. Friedman ranking among MHV of obtained solutions at the first generation by 5 algorithms when $\tau_t$ = 50, 100 and 200, respectively.




(a) $\tau_t$ = 50 (b) $\tau_t$ = 100 (c) $\tau_t$ = 200

Fig. 8. Friedman ranking among MGD of obtained solutions at the first generation by 5 algorithms when $\tau_t$ = 50, 100 and 200, respectively.

algorithms at each parameter setting. Therefore, there are 31 observations obtained from 31 independent runs for each algorithm on each problem with one parameter setting.

Due to the space limitations, only the results of the Friedman and Nemenyi statistical tests are presented here. Mean and standard deviation values of MHV and MGD of obtained solutions in the first generation after changes and the last generation before the next change averaged across 10 environmental changes are presented in the Supplementary File.

### A. Initial Effectiveness of Learnt PS Expansion / Contraction

To verify whether (1) the proposed LEC is able to increase solution quality regarding convergence and diversity right after the change compared to KTDMOEA and DTAEA, and (2) LEC obtains better solution quality over other baselines after the change, the quality of the solutions obtained by all algorithms in the first generation after changes is compared.

*1) Overall Statistic Results:* Figs. 7 and 8 present the Nemenyi post-tests results among MHV and MGD of obtained solutions at the first generation after changes by 5 algorithms, when $\tau_t$ = 50, 100 and 200, respectively. Table I shows the number of win/loss/ties in terms of MHV and MGD for LEC compared to the other algorithms based on Wilcoxon rank sum tests at the first generation, for the set of discrete time steps across all 10 changes ('Overall'), 5 changes of increasing NObj from 2 to 7 one by one ('InNObj') and 5 changes of decreasing NObj from 7 to 2 one by one ('DeNObj'), respectively.

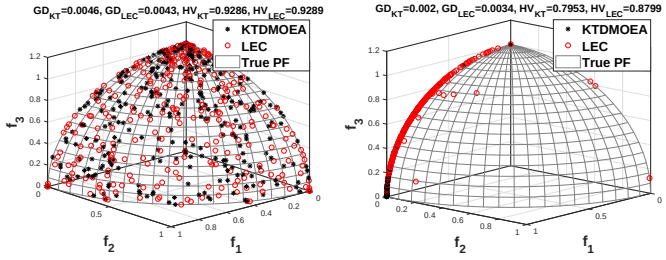
Overall, it can be seen from Fig. 7 that when comparing all algorithms, LEC significantly outperformed all others at all frequencies of changes in terms of MHV. In addition, it can be observed from Fig. 8 that LEC was the best, together with DTAEA, regarding MGD at all frequencies of changes. This implies that the proposed LEC indeed obtained the best quality of transferred solutions right after changes among all compared algorithms, under all frequencies of changes. This observation is also confirmed by the 'Overall' row of Table I. This implies that the proposed LEC indeed improves the quality of transferred solutions regarding convergence and diversity directly after changes.

It can be seen from the 'InNObj' row of Table I that LEC got significantly better MHV than and similar MGD to KTDMOEA when increasing NObj from 2 to 7. This implies that LEC indeed increases the diversity of KTDMOEA when increasing NObj right after changes. In addition, it can be found from the 'DeNObj' row of Table I that LEC got better MGD than all other algorithms when decreasing NObj from 7 to 2, which implies that LEC indeed increases the convergence when decreasing NObj right after changes. The comparison results of LEC and KTDMOEA regarding MGD on cases of 'InNObj' and 'DeNObj' explain the result that LEC got overall better MGD than KTDMOEA.

TABLE I

NUMBER OF WIN/LOSS/TIES IN TERMS OF MHV AND MGD FOR LEC AGAINST OTHER ALGORITHMS AT THE FIRST GENERATION BASED ON WILCOXON RANK SUM TESTS, FOR THE SET OF DISCRETE TIME STEPS ACROSS ALL 10 CHANGES ('OVERALL'), 5 CHANGES OF INCREASING NOBJ FROM 2 TO 7 ONE BY ONE ('INNOBJ') AND 5 CHANGES OF DECREASING NOBJ FROM 7 TO 2 ONE BY ONE ('DENOBJ'), RESPECTIVELY.

| Win/Loss/Tie | Metrics | DNSGA2 | MOEAD-KF | DTAEA | KTDMOEA |
|---|---|---|---|---|---|
| Overall | MHV | 34/2/3 | 36/3/0 | 36/2/1 | 31/8/0 |
| | MGD | 39/0/0 | 32/7/0 | 18/21/0 | 24/8/7 |
| InNObj | MHV | 35/4/0 | 36/3/0 | 35/1/3 | 5/0/34 |
| | MGD | 36/0/3 | 27/12/0 | 11/21/7 | 3/3/32 |
| DeNObj | MHV | 34/3/2 | 36/2/1 | 28/8/3 | 31/8/0 |
| | MGD | 37/2/0 | 31/8/0 | 26/13/0 | 27/9/3 |




(a) F2 (b) F4

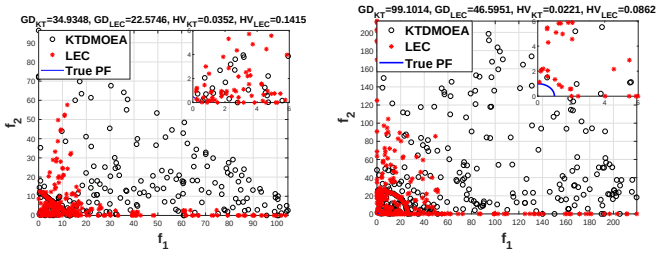
Fig. 9. Transferred solutions by KTDMOEA and LEC on two problem F2 and F4 right after increasing the NObj from 2 to 3. The GD and HV values of both approaches are also put in the title of each subfigure.

For readers who want to examine the details, results of mean and standard deviation values for MHV and MGD are presented in Tables 1-6 of the Supplementary File.

*2) How and Why Does LEC Get Better Solution Quality Right After Changes?:* In this section, examples are presented to testify how LEC got overall better solutions on most problems than KTDMOEA. The reasons behind are then elaborated. Since learning the expansion and contraction directions is one of the essential and key steps to expand and contract PSs, the specific process of learning the directions will be particularly detailed.

*a) When Increasing NObj:* Fig. 9 presents transferred solutions by KTDMOEA and LEC on problems F2 and F4 right after increasing the NObj from 2 to 3. The GD and HV values of both approaches are also listed above each subfigure. It is clear that KTDMOEA and LEC got equally good solutions on F2 while LEC got better solutions than KTDMOEA on F4. This result demonstrates that our proposed strategy of learning to expand PS indeed overcomes the limitation of KTDMOEA in solving problems with strong bias in the fitness landscape.

Note that transferred solutions by LEC were still not so well spread throughout the true PF of F4. The reason is that even though LEC is able to determine the most promising expansion directions, the strong bias of F4 hinders most transferred solutions generated along the directions spread on the true PF, since only a small portion of $x$ values within the range of [0,1] would make $y = x^{100}$ in F4 cover most values of [0,1]




(a) F1 (b) F3

Fig. 10. Transferred solutions by KTDMOEA and LEC on two problem F1 and F3 right after decreasing the NObj from 3 to 2. The GD and HV values of transferred solutions by both approaches are also put in the title of each subfigure.

with all other values of $x$ resulting in $y$ equal to 0.

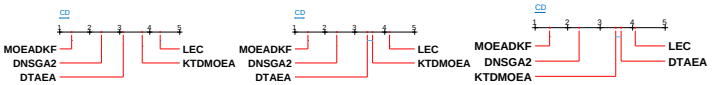
For problems with either strong bias or no bias in the fitness landscape, as long as solutions are converged before the change, LEC was able to learn the correct expansion directions. As described in Section II-C, for problems with strong bias like F4, KTDMOEA is unable to find the expansion directions. However, in our proposed LEC, the expansion direction is learnt from the candidate directions that are perpendicular to the previous PS. Therefore, as long as solutions are converged before the change, as shown in Fig. 6(a), the eigenvectors can be obtained via PCA for learning the candidate expansion directions. After learning these candidate directions, the most promising expansion directions can be naturally selected according to which direction helps diversity.

*b) When Decreasing NObj:* Fig. 10 presents transferred solutions by KTDMOEA and LEC on problems F1 and F3 right after decreasing the NObj from 3 to 2. The GD and HV values of both approaches are also shown above each subfigure. It is clear that LEC got better solutions than KTDMOEA on both F1 and F3 regarding both convergence and diversity. This result demonstrates that our proposed learning to contract PS indeed overcomes the limitation of KTDMOEA in solving problems with multimodality and variable correlation and improves population convergence.

When decreasing the NObj, the manifold of PS after the change is a subset of that before the change [3]. Therefore, there are certain directions along which solutions move from the manifold of PS after the change to that before the change. In this case, these directions locate in or across the PS manifold before the change. Considering these facts, our proposed method of learning contraction directions using PCA exactly captures these potential contraction directions, which are $\mathbf{ev}_1$ and $\mathbf{ev}_2$, as shown in Fig. 6(b). Then, based on which candidate direction results in solutions with better convergence, the most promising contraction directions can be chosen from the candidate directions. Therefore, the contracted PS in LEC can have better convergence than that of KTDMOEA, which does not have any mechanism for improving convergence.

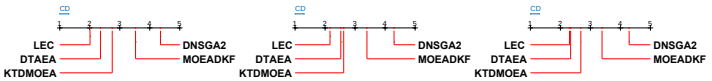
### *B. How Does Learnt PS Expansion and Contraction Help Optimization Process after the Changes?*

To verify whether LEC can find solutions with better convergence and diversity after $gap$ generations of running the optimization against all other algorithms, the solution quality of all compared algorithms after $gap$ generations' optimization for different frequencies of changes is compared.




(a) $gap$ = 5 (b) $gap$ = 25 (c) $gap$ = 50

Fig. 11. Friedman ranking among MHV of optimized solutions after $gap$ generations by 5 algorithms when $\tau_t$=50.




(a) $gap$ = 5 (b) $gap$ = 25 (c) $gap$ = 50

Fig. 12. Friedman ranking among MGD of optimized solutions after $gap$ generations by 5 algorithms when $\tau_t$=50.

*1) Overall Statistic Results:* Figures 11-12, 13-14 and 15-16 present the Nemenyi post-tests results among HV and GD of obtained solutions after $gap$ generations' optimization by 5 algorithms, when $\tau_t$ = 50, 100 and 200, respectively. Table II shows the number of win/loss/ties in terms of MHV and MGD for LEC against other algorithms based on Wilcoxon rank sum tests after optimization of different generations under each different frequency of changes (i.e., $\tau_t$=50, 100 and 200), for the set of discrete time steps across all 10 changes ('Overall'), 5 changes of increasing NObj from 2 to 7 ('InNObj') and 5 changes of decreasing NObj from 7 to 2 'DeNObj', respectively.

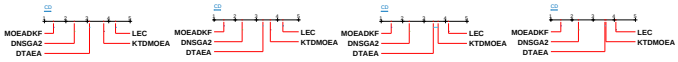
Overall, the statistical test results show that LEC performed significantly better than or similar to the other approaches at all settings of $gap$ under different frequencies of change, regarding MHV and MGD. Additionally, it can be seen from the 'InNObj' row of Table II that LEC got slightly better MHV and slightly worse MGD than KTDMOEA when increasing NObj from 2 to 7. This implies that LEC also increases the diversity of KTDMOEA after optimization when increasing NObj. In addition, it can be found from the 'DeNObj' row of Table II that LEC got significantly better MHV and MGD than all other algorithms when decreasing NObj from 7 to 2, which implies that LEC increases both convergence and diversity after optimization when decreasing NObj.

From Figures 11-16, it is also clear that LEC got significantly best results among all compared algorithms regarding MHV and MGD values when $gap$ is smaller for all frequencies of changes, whereas some ties occur when $gap$ is larger. In addition, LEC was superior to KTDMOEA regarding MGD in almost all cases, which shows that our proposed approach is also able to maintain its superiority over KTDMOEA regarding convergence in the optimization process. Note that for all frequencies of changes, when $gap$ is smaller, solution quality obtained by LEC was significantly better than DTAEA in most cases. This implies that our proposed LEC is more suitable for dynamic optimization, as better solutions are urged to be found given limited budgets in real-word dynamic scenarios.

*2) Why do the Learnt PS Expansion and Contraction Help Optimization Process after the Changes?:* As shown in Section V-A, LEC obtained competitive solution quality compared to other state-of-the-arts in the first generation after changes. This means that LEC was able to quickly find solutions with good convergence and diversity at all frequencies of change. LEC also managed to obtain good solutions at the initial stage

TABLE II
NUMBER OF WIN/LOSS/TIES IN TERMS OF MHV AND MGD FOR LEC AGAINST OTHER ALGORITHMS BASED ON WILCOXON RANK SUM TESTS ACROSS ALL DIFFERENT VALUES OF $gap$, FOR THE SET OF DISCRETE TIME STEPS ACROSS ALL 10 CHANGES ('OVERALL'), 5 CHANGES OF INCREASING NOBJ FROM 2 TO 7 ('INNOBJ') AND 5 CHANGES OF DECREASING NOBJ FROM 7 TO 2 'DENOBJ', RESPECTIVELY.

| Win/Loss/Tie | $\tau_t$ | Metrics | DNSGA2 | MOEAD-KF | DTAEA | KTDMOEA |
|---|---|---|---|---|---|---|
| Overall | 50 | MHV | 33/6/0 | 36/3/0 | 30/4/5 | 30/2/7 |
| | | MGD | 36/3/0 | 25/14/0 | 25/11/3 | 21/7/11 |
| | 100 | MHV | 44/8/0 | 48/4/0 | 41/10/1 | 45/5/3 |
| | | MGD | 48/4/0 | 32/20/0 | 31/16/5 | 26/11/15 |
| | 200 | MHV | 55/10/0 | 57/5/3 | 42/5/18 | 42/1/22 |
| | | MGD | 60/0/5 | 42/23/0 | 44/15/6 | 34/10/21 |
| InNObj | 50 | MHV | 34/5/0 | 36/3/0 | 26/7/6 | 3/3/33 |
| | | MGD | 38/1/0 | 23/15/1 | 23/6/10 | 0/3/33 |
| | 100 | MHV | 44/7/1 | 47/5/0 | 35/12/5 | 4/0/48 |
| | | MGD | 52/0/0 | 31/19/2 | 25/13/14 | 1/7/44 |
| | 200 | MHV | 55/5/5 | 56/5/4 | 39/13/13 | 6/0/59 |
| | | MGD | 65/0/0 | 39/23/3 | 30/20/15 | 0/8/57 |
| DeNObj | 50 | MHV | 32/5/1 | 36/3/0 | 23/12/4 | 30/4/5 |
| | | MGD | 36/3/0 | 26/11/2 | 27/11/1 | 22/5/12 |
| | 100 | MHV | 43/8/1 | 48/4/0 | 32/16/4 | 38/6/8 |
| | | MGD | 48/4/0 | 34/18/0 | 41/9/2 | 34/10/8 |
| | 200 | MHV | 55/10/0 | 60/0/5 | 43/2/20 | 43/2/20 |
| | | MGD | 60/3/2 | 42/21/2 | 59/1/5 | 46/13/6 |




(a) $gap$=5 (b) $gap$=25 (c) $gap$=50 (d) $gap$=100

Fig. 13. Friedman ranking among MHV of optimized solutions after $gap$ generations by 5 algorithms when $\tau_t$=100.

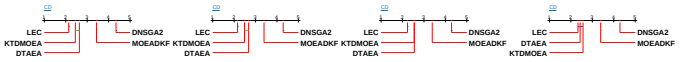
of the evolution (i.e., when $gap$ is small), which is supported by the Friedman test results in Section V-B1. This means that LEC is robust to different gaps of outputting optimized solutions under different frequencies of change.

Because the transferred solutions are better distributed and converged in the new environment, LEC is able to find better solutions across different frequencies of change. This is also the reason why LEC is able to quickly respond to the changes in the NObj, since finding good solution under high frequency of change means fast response to changes. When $gap$ is large, there is less need for obtaining a good initial population right after the changes, because more time can be spend searching for good solutions through the optimisation process. There are some specific problem instances where LEC did not perform best when the frequency of changes is high. The specific results and analyses are presented in Section III.B.3) of the Supplementary File.

### C. Performance Comparison on Other Changes in the NObj

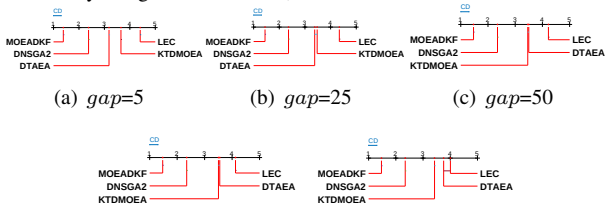
In the previous experiments, the NObj only increases or decreases one by one. This section aims to verify the performance of the proposed algorithm in the scenario where the NObj increases or decreases by more than one. The new changing sequence where the NObj increases or decreases by two each time is designed as follows: NObj increasing from 2 to 4 and then from 4 to 6; then NObj decreases from 6 to 4 and then from 4 to 2, i.e., '2-4-6-4-2'. Experimental settings including test problems and compared algorithms are set as the same to those in Section IV-C. Here, both the frequency of change and $gap$ are set as 200. Besides, only MHV is used to measure the quality of optimized solutions at the last generation to save space. The used statistical tests are the same as those in paragraphs above Section V-A.

Fig. 17 presents the Nemenyi post-tests results among MHV values of solutions at the first generation after changes and




(a) $gap$=5 (b) $gap$=25 (c) $gap$=50 (d) $gap$=100

Fig. 14. Friedman ranking among MGD of optimized solutions after $gap$ generations by 5 algorithms when $\tau_t$=100.




(a) $gap$=5 (b) $gap$=25 (c) $gap$=50

(d) $gap$=100 (e) $gap$=200

Fig. 15. Friedman ranking among MHV of optimized solutions after $gap$ generations by 5 algorithms when $\tau_t$=200.

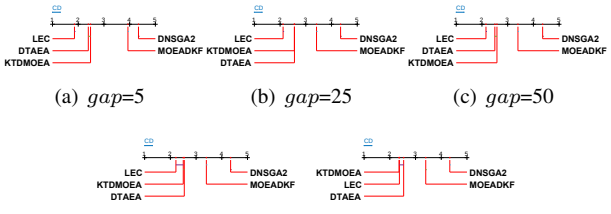
optimized solutions at the last generation by all compared algorithms. Overall, LEC performed significantly best among all compared algorithms at the first generation right after changes in the scenario where NObj changes by more than one. More details can be found in Tables 141 and 142 in the Supplementary File.

To explore the reason why LEC got significantly worse results than DTAEA after optimization when the NObj changes two-by-two, the HV trajectory of sampled generations over evolution on problem F3 is drawn in Fig. 18 to see in which stage LEC performed worse. We can see that, when decreasing the NObj, LEC got worse HV results than DTAEA. The contracted solutions by LEC were also worse than the reconstructed solutions by DTAEA. The reason might be that even though LEC finds most promising contraction directions, it is still almost impossible for some solutions to reach the true PS after decreasing the NObj. For example, suppose the true PS of a problem with four objectives is a 3-dimension cube and the true PS of a problem with two objectives is a 1-dimension line segment in the cube. When decreasing the NObj from 4 to 2, solutions filled in the cube need to move along different directions from their original position to reach the line segment.

### D. Impact of LEC Strategy Parameter

In the process of generating solutions along the learnt contraction directions when decreasing NObj, there is a parameter $\lambda$ to set which is the number of selected solutions to generate an initialized population. Different values of $\lambda$ will be set to verify whether different parameter settings affect the performance of LEC against other compared algorithms. Experimental settings including test problems and compared algorithms are set as the same to those in Section IV-C. Here, both the frequency of change and gap are set as 200. Besides, only MHV is used to measure the quality of optimized solutions at the last generation to save space. The changing sequence is to decrease NObj from 7 to 2 one by one. There are three LECs (denoted as LEC-10, KTDMOEA-20 and KTDMOEA-30), which have the value of $\lambda$ as $popsize/10$, $popsize/20$ and $popsize/30$, respectively.

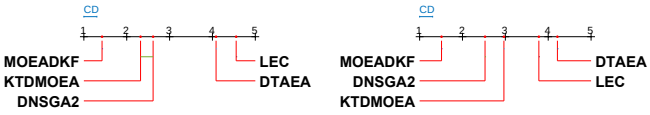
In order to verify whether different parameter settings affect the performance of LEC, the Friedman and Nemenyi tests on 4 state-of-the-arts and 3 LECs were conducted to




(a) $gap$=5 (b) $gap$=25 (c) $gap$=50

(d) $gap$=100 (e) $gap$=200

Fig. 16. Friedman ranking among MGD of optimized solutions after $gap$




(a) At the first generation after changes (b) At the last generation after opt

Fig. 17. Friedman ranking among MHV of obtained solutions in the first generation and solutions after $\tau_t$ generations' optimization by 5 algorithms when $\tau_t$=200.

indicate the significant differences among them. The setting of the Friedman and Nemenyi tests are the same as those in paragraphs above Section V-A. The results are in Fig. 19. The three LECs got the best or equally best MHV values among all algorithms. There is no significant difference among the three LECs with different setting of the parameter $\lambda$. These results show that the performance of the proposed LEC approach was not sensitive to these setting of the parameter $\lambda$. The performance of LEC against the existing algorithms was not affected by the setting of different parameter values.

### *E. Sensitivity Analysis of LEC to Evolutionary Algorithm Parameters*

This section aims to analyze the sensitivity of LEC to evolutionary algorithm parameters, i.e., probability of crossover and mutation, and population size. To save space, we let our proposed LEC run on six selected representative problems with different problem features under different settings of probability of crossover and mutation, and population size. Note that when analyzing the sensitivity of one parameter, the other parameters are fixed to the ones used in Section IV-C.

Table III presents the MHV of LEC with different settings of crossover and mutation probability on six problems. LEC with the setting of $Pc = 1$ performed the best when comparing the last column and the first three columns where the setting of the mutation probability is the same. When comparing the last three columns where the setting of the crossover probability is the same, we can find that the setting of $Pm = 0.05$ performed best on five problems. However, LEC with this setting performed the worst on F4 with strong bias. The reason is that much diversity is required when solving problems with strong bias. Therefore, the setting of $Pm = 0.1$ is the most compromising one on all investigated problems.

Table IV presents the MHV of LEC with different settings of population size on six problems. Note that LEC with $popsize = 500$ costs the most function evaluations, since all algorithms run the same number of generations. It is intuitive that an algorithm with more function evaluations would obtain better solution quality. It can be found from Table IV that LEC with the setting of $popsize = 500$ got the best MHV value on 5 out of 6 problems. However, the MHV value improvement

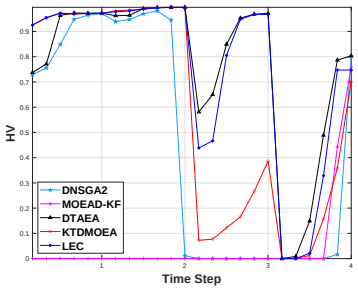



Fig. 18. HV trajectory of sampled generations (1, 5, 25, 100 and 200) after each change over evolution on one problem F3.

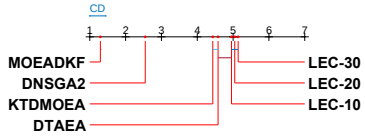



Fig. 19. Friedman ranking among MHV of optimized solutions at the last generation by 6 state-of-the-arts and 3 KTDMOEAs with different values of parameters theta (1, 2 and 4) in the changing sequence of Equation 4.

of LEC with $popsize = 500$ was not much, compared to LEC with $popsize = 300$. Therefore, the setting of population size as 300 in our paper makes a compromise between solution quality and number of function evaluations.

### *F. Running Time Analysis*

To investigate how efficient our proposal is compared to other algorithms, we recorded the running time (in seconds) of all compared algorithms on all benchmark problems with the changing sequence of Equation (4) when $\tau_t = 50$ under the same hardware configuration [1]. Also, we used Friedman and Nemenyi statistical tests [34] with a significance level 0.05 across all benchmark problems to compare the running time of the five compared algorithms. The running time obtained by all algorithms on one problem under one run is regarded as an observation of the test.

Fig. 20 presents the Nemenyi post-tests results among running time of all compared algorithms. The specific mean values of running time (in seconds) obtained by all compared algorithms are presented in Table 28 of the Supplementary File. It is clear from Fig. 20 that our proposed LEC achieved competitive efficiency compared to algorithms tailored for changing NObj (KTDMOEA and DTAEA). In addition, our intuition is right that algorithms with one archive (LEC and KTMOEA) cost significantly less running time than that with two archives (DTAEA). Meantime, algorithms without specific strategies to cope with changing NObj (DNSGA2 and MOEAD-KF) ran faster.

### *G. How Does the HV Change as the NObj Changes?*

To investigate how the HV changes when the NObj changes over time, we plot the HV trajectory of sampled generations after each change over evolution on problems F1, F4, WFG3, WFG4, WFG6 and WFG9 for all $\tau_t s$. To save space, only the figures for problems F1 and F4 when $\tau_t$ = 50 are presented in Fig. 21. The figures for all other cases are presented in Figs. 1-3 of the Supplementary File. It is clear from Fig. 21 that LEC

[1]The implementation environment is as follows: 2.20-GHz Intel Core i7, 8-GB DDR4 2666MHz.

TABLE III
MHV VALUE OF LEC WITH DIFFERENT SETTINGS OF CROSSOVER PROBABILITY (PC) AND MUTATION PROBABILITY (PM) UNDER 31 RUNS. THE BEST AND SECOND BEST FOR EACH PROBLEM ARE SHOWN WITH DARK BACKGROUND AND BOLD FACE, RESPECTIVELY.

| Pc/Pm | 0.2/0.1 | 0.5/0.1 | 0.8/0.1 | 1/0.05 | 1/0.2 | 1/0.1 |
|---|---|---|---|---|---|---|
| F1 | 9.442E-01 | 9.905E-01 | 9.944E-01 | 9.960E-01 | 7.599E-01 | **9.949E-01** |
| F2 | 9.577E-01 | 9.581E-01 | 9.582E-01 | 9.585E-01 | 9.579E-01 | **9.584E-01** |
| F4 | 9.452E-01 | 9.390E-01 | 9.080E-01 | 8.747E-01 | 9.546E-01 | **9.491E-01** |
| WFG2 | 9.622E-01 | 9.641E-01 | 9.645E-01 | 9.679E-01 | 9.668E-01 | **9.678E-01** |
| WFG5 | 7.794E-01 | 7.886E-01 | 7.930E-01 | 7.951E-01 | 7.895E-01 | **7.938E-01** |
| WFG8 | 8.286E-01 | 8.312E-01 | 8.320E-01 | 8.325E-01 | 8.302E-01 | **8.322E-01** |

TABLE IV
MHV VALUE OF LEC WITH DIFFERENT SETTINGS OF POPULATION SIZE UNDER 31 RUNS. THE BEST FOR EACH PROBLEM IS SHOWN WITH DARK BACKGROUND.

| Popsize | 100 | 500 | 300 |
|---|---|---|---|
| F1 | 8.663E-01 | 9.958E-01 | 9.954E-01 |
| F2 | 9.441E-01 | 9.590E-01 | 9.584E-01 |
| F4 | 7.360E-01 | 8.941E-01 | 9.480E-01 |
| WFG2 | 9.574E-01 | 9.684E-01 | 9.678E-01 |
| WFG5 | 7.201E-01 | 7.982E-01 | 7.936E-01 |
| WFG8 | 7.659E-01 | 8.372E-01 | 8.322E-01 |

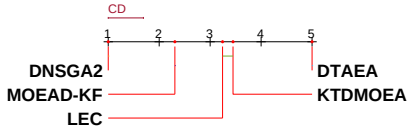
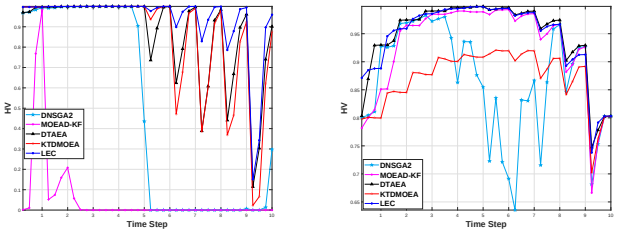



Fig. 20. Friedman ranking among running time obtained by all compared algorithm in changing sequence of Equation (4). Here, larger ranks mean larger running time.




(a) F1 (b) F4

Fig. 21. HV trajectory of sampled generations (1, 5, 25 and 50) after each change over evolution on problems F1 and F4, when $\tau_t = 50$.

performed better than KTDMOEA on F4 when increasing the NObj. In addition, LEC got better HV results when decreasing the NObj on these two problems. This observation is similar for most other problems in the Supplementary File. These results show that our proposed LEC indeed overcomes the limitations of KTDMOEA.

### *H. Results on Real-world Problems*

In this section, we utilize three widely used multi-objective optimization real-world problems to simulate the scenario of changing the NObj to test LEC's performance. These three problems are water problem [35], Car-Side Impact Problem [36] and Crash Worthiness in Design of Vehicles [37]. The water problem has five objectives and the others have three objectives. The NObj of these problems is originally set as 2 and then increases to their own maximal NObj one by one and then decreases to 2 one by one. The frequency of changes is set as 25. All the experimental settings for all compared algorithms are the same to those in Section IV-C. The same significant tests were also used. The MHV values of each compared algorithm on one problem at each run were regarded as one observation for the Friedman ranking and Nemenyi test. The Friedman ranking to compare the MHV of the obtained solutions in the first generation and at the last generation after optimization on the three real-world problems are presented in Fig. 22. Specific MHV values of obtained solutions for all compared algorithms on these problems can be found in Table 29 of the Supplementary File. It is clear from these figures that LEC achieved competitive performance regarding solution quality on the real-world problems.

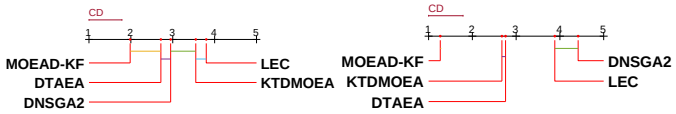
## VI. CONCLUSION AND FUTURE WORK

This paper showed that the state-of-the-art transfer approach (KTDMOEA) still has some limitations when solving DMOPs with a changing NObj. Specifically, when increasing NObj, transferred solutions by KTDMOEA lacks diversity on the problem with extremely strong bias, since the heuristic PS expansion of KTDMOEA is unable to find expansion directions on this kind of problem. In addition, KTDMOEA loses convergence on problems with multi-modality and variable correlation when decreasing NObj, because solutions generated along the found contraction directions only aim at enhancing diversity rather than convergence.

To overcome the limitations of KTDMOEA, we proposed a novel knowledge transfer strategy, LEC, so as to enhance diversity and convergence for increasing and decreasing the NObj, respectively. LEC conducts PCA on the PS of the problem before the change to get eigenvectors and uses the concept of dominance to select promising expansion and contraction directions. Experimental investigations have demonstrated the effectiveness of the proposed LEC strategy in improving the diversity and convergence right after changes and after optimization is run for a number of generations. Such improvements were observed for all frequencies of change.

As expected, no algorithm would be the best on all possible problems. According to the details in the Supplementary File, there are some problems on which LEC did not outperform existing algorithms. Moreover, we found that LEC still lacks convergence when the NObj is decreased by more than one. Future work could improve the algorithm for these kind of problems. Testing and improving our proposed algorithm on problems with large and random changes is also a possible direction of our future work. For that, more advanced learning methods could be proposed to learn as many expansion or contraction directions as possible right after the changes, whereas approaches with stronger diversity enhancement and converging capability could be proposed for further evolving the population after the changes. Future work could also relax the assumption made by LEC that the PS with less NObj is a subset of that with more NObj.

LEC has been evaluated and compared extensively to other methods on a number of problems. While such experimental studies help to evaluate and understand how, why and when LEC works, they do leave room for its further evaluation on other problems. In practice, if a real-world problem shares the same or similar characteristics as the problems used in this paper, LEC is expected to work well. Otherwise the performance of LEC needs to be further tested. In short, there are three




(a) At the first generation after changes (b) At the last generation after opt

Fig. 22. Friedman ranking among MHV of obtained solutions in the first generation and solutions after $\tau_t$ generations' optimization by 5 algorithms. Here, larger ranks mean higher MHV.

main areas in practice that may pose threats to the validity of our work. First, the underlying problem structure in practice may be very different from those considered in the problems investigated in our paper. Second, the dynamics, including the frequency and magnitude of changes, in practice may be very different from those considered in our study. Third, the problem scale, including both numbers of decision variables and objectives, may be much larger than that considered in our paper. Last but not least, the computational cost of LEC might be an issue for some real-world applications that need near real-time responses. These are all future directions of our research.

## REFERENCES

[1] M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: test cases, approximations, and applications, IEEE Transactions on Evolutionary Computation 8 (5) (2004) 425–442.

[2] G. Ruan, G. Yu, J. Zheng, J. Zou, S. Yang, The effect of diversity maintenance on prediction in dynamic multi-objective optimization, Applied Soft Computing 58 (2017) 631–647.

[3] R. Chen, K. Li, X. Yao, Dynamic multiobjectives optimization with a changing number of objectives, IEEE Transactions on Evolutionary Computation 22 (1) (2017) 157–171.

[4] X. Shen, L. L. Minku, R. Bahsoon, X. Yao, Dynamic software project scheduling through a proactive-rescheduling method, IEEE Transactions on Software Engineering 42 (7) (2015) 658–686.

[5] X.-N. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Information Sciences 298 (2015) 198–224.

[6] J. Xiao, L. J. Osterweil, Q. Wang, M. Li, Dynamic resource scheduling in disruption-prone software development environments, in: International Conference on Fundamental Approaches to Software Engineering, Springer, 2010, pp. 107–122.

[7] J. Zhuo, C. Chakrabarti, An efficient dynamic task scheduling algorithm for battery powered dvs systems, in: 2005 Asia and South Pacific Design Automation Conference, 2005, pp. 846–849.

[8] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff, X. Yao, Knowledge transfer for dynamic multi-objective optimization with a changing number of objectives,Doi:10.48550/arXiv.2306.10668 (2023).

[9] S.-U. Guan, Q. Chen, W. Mo, Evolving dynamic multi-objective optimization problems with objective replacement, Artificial Intelligence Review 23 (3) (2005) 267–293.

[10] R. Azzouz, S. Bechikh, L. B. Said, A multiple reference point-based evolutionary algorithm for dynamic multi-objective optimization with undetectable changes, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 3168–3175.

[11] L. Huang, I. H. Suh, A. Abraham, Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants, Information Sciences 181 (11) (2011) 2370–2391.

[12] S. Jiang, M. Kaiser, S. Yang, S. Kollias, N. Krasnogor, A scalable test suite for continuous dynamic multiobjective optimization, IEEE Transactions on Cybernetics 50 (6) (2019) 2814–2826.

[13] A. Gupta, Y.-S. Ong, L. Feng, Multifactorial evolution: toward evolutionary multitasking, IEEE Transactions on Evolutionary Computation 20 (3) (2015) 343–357.

[14] A. Gupta, Y.-S. Ong, L. Feng, Insights on transfer optimization: Because experience is the best teacher, IEEE Transactions on Emerging Topics in Computational Intelligence 2 (1) (2017) 51–64.

[15] B. Da, A. Gupta, Y.-S. Ong, Curbing negative influences online for seamless transfer evolutionary optimization, IEEE Transactions on Cybernetics (99) (2018) 1–14.

[16] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, A. K. Qin, Evolutionary multitasking via explicit autoencoding, IEEE Transactions on Cybernetics 49 (9) (2018) 3457–3470.

[17] K. C. Tan, L. Feng, M. Jiang, Evolutionary transfer optimization-a new frontier in evolutionary computation research, IEEE Computational Intelligence Magazine 16 (1) (2021) 22–33.

[18] M. Jiang, Z. Huang, L. Qiu, W. Huang, G. G. Yen, Transfer learning-based dynamic multiobjective optimization algorithms, IEEE Transactions on Evolutionary Computation 22 (4) (2017) 501–514.

[19] L. Feng, W. Zhou, W. Liu, Y.-S. Ong, K. C. Tan, Solving dynamic multiobjective problem via autoencoding evolutionary search, IEEE Transactions on Cybernetics (2020).

[20] M. Jiang, Z. Wang, L. Qiu, S. Guo, X. Gao, K. C. Tan, A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning, IEEE Transactions on Cybernetics 51 (7) (2020) 3417–3428.

[21] S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Transactions on Neural Networks 22 (2) (2010) 199–210.

[22] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff, X. Yao, When and how to transfer knowledge in dynamic multi-objective optimization, in: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2019, pp. 2034–2041.

[23] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff, X. Yao, Computational study on effectiveness of knowledge transfer in dynamic multi-objective optimization, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–8.

[24] A. Zhou, Estimation of distribution algorithms for con-

tinuous multiobjective optimization, Ph.D. thesis, University of Essex (2009).
[25] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multiobjective optimization, Springer, 2005.
[26] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, Evolutionary Computation, IEEE Transactions on 10 (5) (2006) 477–506.
[27] K. Deb, S. Karthik, et al., Dynamic multi-objective optimization and decision-making using modified nsga-ii: a case study on hydro-thermal power scheduling, in: Evolutionary Multi-Criterion Optimization, Springer, 2007, pp. 803–817.
[28] A. Muruganantham, K. C. Tan, P. Vadakkepat, Evolutionary dynamic multiobjective optimization via kalman filter prediction, IEEE Transactions on Cybernetics 46 (12) (2015) 2862–2873.
[29] K. Deb, Multi-objective optimization using evolutionary algorithms, Vol. 16, John Wiley & Sons, 2001.
[30] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.
[31] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. Da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.
[32] C.-K. Goh, K. C. Tan, A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization, IEEE Transactions on Evolutionary Computation 13 (1) (2009) 103–127.
[33] M. Li, X. Yao, Quality evaluation of solution sets in multiobjective optimisation: A survey, ACM Computing Surveys (CSUR) 52 (2) (2019) 1–38.
[34] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.
[35] T. Ray, K. Tai, C. Seow, An evolutionary algorithm for multiobjective optimization, Engineering Optimization 33 (3) (2001) 399–424.
[36] H. Jain, K. Deb, An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach, IEEE Transactions on Evolutionary Computation 18 (4) (2013) 602–622.
[37] X. Liao, Q. Li, X. Yang, W. Zhang, W. Li, Multiobjective optimization for crash safety design of vehicles using stepwise regression model, Structural and Multidisciplinary Optimization 35 (2008) 561–569.

PLACE PHOTO HERE

**Gan Ruan** is currently pursuing the Ph.D. degree in computer science at University of Birmingham, UK. He was a visiting PhD student at the Southern University of Science and Technology, Shenzhen 518055, China, when part of this work was done. His research interest is knowledge transfer in dynamic multi-objective optimization.

PLACE PHOTO HERE

**Leandro L. Minku** (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Birmingham, Birmingham, U.K., in 2010. He was a Lecturer in computer science with the University of Leicester, Leicester, U.K. He is currently an Associate Professor with the School of Computer Science, University of Birmingham, Birmingham, U.K. His main research interests are machine learning in non-stationary environments / data stream mining, online class imbalance learning, ensembles of learning machines, dynamic optimisation, and computational intelligence for software engineering. Dr. Minku is an Associate Editor-in-Chief of Neurocomputing, was a Senior Editor of the IEEE TNNLS, and is an Associate Editor of Journal of Systems and Software and of Empirical Software Engineering journal.

PLACE PHOTO HERE

**Stefan Menzel** Stefan Menzel received the Dipl.-Ing. degree in civil engineering from RWTH Aachen, Germany, in 1998, and the Ph.D. degree in civil engineering from TU Darmstadt, Germany, in 2004. Since 2004, he is with the Honda Research Institute Europe, Offenbach, Germany, where he is currently Chief Scientist with the Optimization and Creativity Group. His current research interests include evolutionary optimization with special focus on adaptive design representations, machine learning for knowledge transfer, generative AI, and multidisciplinary optimization for real-world applications.

PLACE PHOTO HERE

**Bernhard Sendhoff** (Fellow, IEEE) received the Ph.D. degree in applied physics from Ruhr-Universität Bochum, Bochum, Germany, in 1998. He was with Honda Research Institute Europe GmbH, Offenbach, Germany, from 2003 to 2010, as a Chief Technology Officer, and from 2011 to 2017, as a President. Since 2017, he has been an Operating Officer with Honda Research and Development Ltd., Tokyo, Japan, and since 2019, the CEO of the Global Network Honda Research Institutes. He is an Honorary Professor with the Technical University of Darmstadt, Darmstadt, Germany. He has authored or coauthored over 180 scientific publications. Dr. Sendhoff is a Senior Member of ACM and a member of SAE.

PLACE PHOTO HERE

**Xin Yao** (Fellow, IEEE) received the his BSc from the University of Science and Technology of China (USTC) in 1982, his MSc from the North China Institute of Computing Technology in 1985, and his Ph.D. from USTC, in 1990. He is currently Tong Tin Sun Chair Professor of Machine Learning at Lingnan University in Hong Kong, and a part-time Professor of computer science with the University of Birmingham, Birmingham, U.K. His major research interests include evolutionary computation, ensemble learning, and their applications. His work won the 2001 IEEE Donald G. Fink Prize Paper Award; 2010, 2016 and 2017 IEEE Transactions on Evolutionary Computation Outstanding Paper Awards; 2011 IEEE Transactions on Neural Networks Outstanding Paper Award; and many other best paper awards at conferences. He received a Royal Society Wolfson Research Merit Award in 2012, the IEEE CIS Evolutionary Computation Pioneer Award in 2013 and the 2020 IEEE Frank Rosenblatt Award.