Computational study on effectiveness of knowledge transfer in dynamic multi-objective optimisation

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

DOI:
https://doi.org/10.1109/CEC48606.2020.9185907

Computational Study on Effectiveness of Knowledge Transfer in Dynamic Multi-objective Optimization

Gan Ruan[§], Leandro L. Minku[§], Stefan Menzel[†], Bernhard Sendhoff[†] and Xin Yao[§‡]
[§] CERCIA, School of Computer Science, University of Birmingham, UK
[†] Honda Research Institute Europe GmbH, Offenbach 63073, Germany
[‡] Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China.
{GXR847, L.L.Minku, X.Yao}@cs.bham.ac.uk, {stefan.menzel, bernhard.sendhoff}@honda-ri.de

***Abstract*—Transfer learning has been used for solving multiple optimization and dynamic multi-objective optimization problems, since transfer learning is believed to be able to transfer useful information from one problem instance to help solving another related problem instance. This paper aims to study how effective transfer learning is in dynamic multi-objective optimization (DMO). Through computation time analysis of transfer learning, we show that the 'inner' optimization problem introduced by transfer learning is very time-consuming. In order to enhance the efficiency, two alternatives are computationally investigated on a number of dynamic bi- and tri-objective test problems. Experimental results have shown that the greatly enhanced efficiency does not result in much degeneration on the performance of transfer learning. Considering the high computational cost of transfer learning, it is likely that the original purpose of using transfer learning in DMO might be negated. In other words, the computation time saved in optimization is eaten up by computationally expensive transfer learning. As a result, there is less gain than expected in the overall computational efficiency. To verify this, experiments have been conducted, regarding using computational cost of transfer learning to optimize randomly generated solutions. The results have demonstrated that the convergence and diversity of final solutions generated from the random solutions are significantly better than those generated from transferred solutions under the same total computational budget.**

***Keywords*—Evolutionary Algorithms; Transfer Learning; Dynamic Multi-objective Optimization; Prediction-based Method**

I. Introduction

Transfer learning [1] is a kind of machine learning method that is able to transfer the knowledge from a source task to a target task. This inherent characteristic of transfer learning makes it intuitive to apply transfer learning to explore useful experience that have been obtained in one task/problem to solve another related task/problem. The reason is that some similar or related problems/tasks may share some common features, which help to transfer experience from one problem to help solving another problem. As a result, computational resources can be significantly saved when solving similar problems later.

One of the initial attempts at the application of transfer learning to evolutionary computation is by Gupta et. al [2] on a framework of evolutionary multi-tasking optimization (EMT). Subsequently, additional work around transfer optimization for EMT has been proposed. For instance, Da et. al [3] designed an online learning method to seamlessly curb the negative influence of transfer learning in EMT.

Although transfer learning has recently achieved some successes in the field of EMT, it has seldomly been studied in evolutionary multi-objective optimization (EMO). Generally speaking, dynamic multi-objective optimization problems (DMOPs) [4] are a kind of multi-objective optimization problems which involve a series of problems whose objectives change over time [5] [6]. As we normally assume that the changing problems are related, there are good opportunities for the application of transfer learning in dynamic multi-objective optimization (DMO). One originally published paper on DMO introduced transfer learning-based dynamic multi-objective optimization algorithms (Tr-DMOEAs) [7] and an improved version of Tr-DMOEAs [8] was also published. Another work [9] applied the TrAdaboost-based algorithm for transfer regression task [10] to generate an initial population in response to changes in DMOPs.

The key challenge in DMO is how to constantly trace a changing Pareto optimal front (POF) and/or Pareto optimal set (POS) before the next environment change [11]. Aiming at this goal, researchers have proposed a prediction-based method [11], [12]. This kind of method predicts what the good solutions in the next environment are after learning the regularity of the environment changes. In most prediction-based approaches, it is implicitly assumed that the evolution of the solutions used to train and test the prediction model obeys a fixed independent and identical probability distribution. However, this is not always true under dynamic environments in optimization, since the environmental changes may result in different evolution patterns over time. Consequently, the prediction model based on the incorrect assumption may cause inaccurate prediction of optimal solutions. Transfer learning, which does not make this assumption, is a good candidate for solving DMOPs if it can learn and exploit the relationship among different problems.

The main idea behind Tr-DMOEA is to transfer solutions in the Pareto front (POF) of the previous environment to generate

an initial population for the next environment, through a domain adaption method called transfer component analysis (TCA). Specifically, Tr-DMOEA employs the TCA to learn a mapping from the objective space of problems to the latent space, where the difference between the source and target problems is minimal. Then, solution in the target problem is found through minimizing the distance of the optimized solution in the source problem and the found solutions in the target problem.

Experimental studies [7] have shown the superiority of Tr-DMOEAs over the state-of-the-art in DMO. Even though the existing transfer learning achieves a certain level of improvement, it is unclear how efficient it is. The time complexity of TCA and primal dual interior point method for solving the inner problem (equation (2)) has been presented in [7]. However, several parameters in the interior point method are unclear. Thus, the computational cost of solving the inner optimization problem is unknown. It is also unclear whether its efficiency could be enhanced. The original aim of using transfer learning in DMO is to find good solutions such that the population can reach the optimum as soon as possible in the optimization process. If the computational cost of transfer leaning is high, it prevents the purpose of using transfer learning. If this is the case, the cost of transfer learning can be used for optimization, which might obtain better results than transfer learning.

Aiming at investigating these three points, in this paper, we first analyze the computation time of TCA and that of solving the inner problem (equation (2)) with the interior point method, which consumes most of the cost in Tr-DMOEA. Besides, another two Tr-DMOEA variants are computationally studied, in which active set and sequential quadratic programming (sqp) methods are used to solve the inner problem (equation (2)), respectively. Lastly, the cost spent on transfer learning will be used to optimize a random population to compare the quality of transferred solutions and those originating from randomization after optimization, which further aims to figure out whether the purpose of using transfer learning in DMO is guaranteed.

The reminder of this paper is organized as follows: Section II briefly introduces how transfer learning is applied in DMO. In Section III, we analyze computation time of three Tr-DMOEA variants with different inner optimization methods. Qualities of solutions obtained by these three variants are compared in Section IV through experimental studies. In Section V, we provide details on what happens to solutions quality if computation time of transfer learning is used to optimize randomly generated solutions. Section VI states the summary and some potential future work.

II. Transfer Learning-Based Dynamic Multi-objective Evolutionary Algorithm

In this section, we briefly introduce the foundations of Tr-DMOEA. The detailed process of Tr-DMOEA can be found in [7].

In Domain Adaption Learning (DAL) [13], a transfer learning method, it is assumed that a transformation should be found to a latent space where the difference between the distributions of source and target domain is minimized. Once this transformation is found, it can act as a bridge to connect the source domain and the target domain. Then solutions that have been optimized in one domain can be transferred to be good solutions in another domain through this bridge.

The distance between the distributions of the source and target domain can be calculated through the Maximum Mean Discrepancy (MMD) [14], which evaluates the distance between two distributions in the Reproducing Kernel Hilbert Space. Let p and q be two Borel probability distributions defined on a domain $\mathcal{X}$. $FS = \{Fs_1, ..., Fs_m\}$ and $FT = \{Ft_1, ..., Ft_n\}$ are observations drawn from p and q. Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$. f can be written as $f(x) = \langle \phi(x), f \rangle$ in a RKHS, where $\phi(x) : \mathcal{X} \to \mathcal{H}$. The estimated MMD in RKHS can be calculated as:

$$MMD(\mathcal{F}, p, q) := \left\| \frac{1}{m} \sum_{i=1}^{m} \phi(Fs_i) - \frac{1}{n} \sum_{i=1}^{n} \phi(Ft_i) \right\|_{\mathcal{H}}^2 \quad (1)$$

In Tr-DMOEA, the distribution of the source and target domains under consideration is the distribution of the objective vectors of source and target solutions. Therefore, in Tr-DMOEA, FS and FT are the objective vectors of randomly generated solutions in the source environment s (i.e., the problem before a change) and target environment t (i.e., the problem after a change), respectively. The function ϕ is defined as $\phi(F) = W^T K(F)$, where W is a transformation matrix which maps the objective vector into the latent space, and $K(F)$ is defined as follows, where $\kappa(\cdot, \cdot)$ is a kernel function [15]:

$$K(F) = [\kappa(Fs_1, F), ..., \kappa(Fs_m, F), \kappa(Ft_1, F), ..., \kappa(Ft_n, F)]^T$$

Once W is found, Tr-DMOEA will initialize the population in the target environment with solutions whose objective vectors are close to that of any good solution from the source environment in the latent space. For that, it needs to find solutions t_k whose objective vector is close to that of a solution s_l from POF of the problem in the source environment (POF_s), i.e., making the following formula minimal:

$$||\phi(Fs_l) - \phi(Ft_k)|| \quad (2)$$

In this paper, we call the problem of searching for the solution t_k of this formula 'inner problem' in transfer learning.

III. Computation Time Analysis of Three Tr-DMOEA Variants

In this section, we give the time complexity of Tr-DMOEA and the computation time analysis of each component in Tr-DMOEA. Both analyses prove that the used minimization method is the most time-consuming part in Tr-DMOEA. Then, two other Tr-DMOEA variants with different minimization methods and the original Tr-DMOEA are compared regarding the computation time.

A. Efficiency of Transfer Learning in DMO

The time complexity of TCA and primal dual interior point has been analyzed in [7]. The major time cost of TCA is spent on the eigenvalue decomposition. It costs $O(d(m+n)^2)$ time when d nonzero eigenvectors are to be extracted, where m and n are the numbers of the solutions which are generated in the search space of source and target problems. For the primal dual interior point method, suppose the constraint matrix A has n rows and m columns, and $n < m$, it has $O(\sqrt{m}L)$ iterations and $O(m^3L)$ arithmetic operations, where L is the total number of bits of the input. It is clear that the time complexity of the interior point is larger than that of TCA.

In order to verify the cost of TCA and interior point method from the perspective of computation time, an initial experiment is conducted regarding how much computation time these two parts consume in a single run. The experimental design is as follows:

- RMMEDA [16] is selected as the optimization algorithm. Here, only one problem dMOP2 is used as the test problem as it is only used to reflect the proportion of computation time of each component in Tr-DMOEA.
- Population iterates for 50 generations both before and after an environmental change, and there is only one change.
- The changing severity is set as 10. The population size is set as 200.
- For the TCA parameters, the Gaussian kernel function is set as the default value and the expected dimensionality is set to be 20. The value of μ is set as 0.5.
- Computation time of each function of the whole algorithm will be recorded by the Profile environment of MATLAB 2018b.

Computation time of each part of the Tr-RMMEDA is recorded and presented in Table I. It is clear from the table that the interior point method consumes the most computation time, which confirms the complexity analysis of TCA and interior point in previous one section.

TABLE I
COMPUTATION TIME OF EACH PROCESS OF TR-RMMEDA WITH ONE RUN, ONE CHANGE AND ONE PARAMETER ON PROBLEM DMOP2.

Process	Computation time	Proportion
Interior point	427.455 s	93.9%
RMMEDA	25.53 s	5.6 %
TCA	2.408 s	0.5%
Sum	455.393 s	100 %

B. Computation Time Comparison of Three Tr-DMOEA Variants

Given that the computation time of solving the inner optimization problem in Tr-DMOEA is very large, it is unclear whether the efficiency of solving the inner problem (equation (2)) could be enhanced. To explore this, other two popular optimization methods are used here, which are the active set [17] and sequential quadratic programming (sqp) [17] methods. We have conducted a set of experiments to validate the efficiency and effectiveness of these three Tr-DMOEA variants.

Here, we give a brief description of active set and sqp method:

- The active set method [17] considers an optimization problem with n constraints $g_1(x) \geq 0, ...g_n(x) \geq 0$. For a point x in the feasible region, a constraint is called active at x if $g_i(x) = 0$ and inactive if $g_i(x) > 0$. The procedures for the active set are as follows: solve the equality problem defined by the active set (approximately) with a solution x_*; compute the Lagrange multipliers of the active set; remove a subset of the constraints with negative Lagrange multipliers; search for the new boundary based on x_* along the feasible region boundary formed by the active set and add the constraint related to the new boundary to the approximated active set. Iterate these procedures until the optima is found.
- Sqp [17] is an iterative method for constrained nonlinear optimization. At each iteration, a basic sequential quadratic programming algorithm defines an appropriate search direction as a solution to the quadratic programming subproblem.

Note that all three optimization methods (interior point, sqp and active set) used in this paper are from the optimization toolbox of MATLAB 2018b. The specific implementation of these three methods and the difference between the sqp and the active-set algorithms can be found in MATLAB's online documents[1].

The IEEE CEC 2015 Benchmark problems [18] are selected as test problems, which comprise 12 bi- and tri-objective problems with different features. For the parameters of these problems, there are 20 changes. In order to study the effectiveness of Tr-DMOEAs in different dynamics, there are three dynamics with different severity of change (i.e., n_t = 1, 10 and 20). They represent large, medium and small environment changes. τ_t refers to the frequency of change, which means within each change, the population is forced to run τ_t generations. Different combinations of n_t and τ_t are shown in Table II. At the beginning of the algorithm, the population iterates for 50 generations, which enables the population to converge.

TABLE II
CONFIGURATIONS OF BENCHMARK FUNCTION PARAMETERS. n_t, τ_t AND τ_T ARE THE SEVERITY OF CHANGE, FREQUENCY OF CHANGE AND MAXIMUM NUMBER OF ITERATIONS, RESPECTIVELY.

	n_t	τ_t	τ_T
C1	10	5	150
C2	10	10	250
C3	10	25	550
C4	10	50	1050
C5	1	10	250
C6	1	50	1050
C7	20	10	250
C8	20	50	1050

The experimental setup is as follows:

[1]https://ww2.mathworks.cn/help/optim/ug/constrained-nonlinear-optimization-algorithms.html.

TABLE III

MEAN AND STANDARD DEVIATIONS OF COMPUTATION TIME (IN SECONDS) OF THREE OPTIMIZATION METHODS FOR MINIMIZING THE DISTANCE OF SOURCE AND TARGET PROBLEMS IN THE LATENT SPACE WITH 20 ENVIRONMENT CHANGES FOR ALL PROBLEMS WITH 2 PARAMETER SETTINGS.

Prob.	C5			C6		
Methods	Interior	sqp	active	Interior	sqp	active
FDA4	6.38e+03(8.93e+04)	8.77e+02(1.28e+03)	**4.59e+02(7.89e+02)**	6.42e+03(1.34e+05)	8.31e+02(7.18e+02)	**4.25e+02(7.10e+02)**
FDA5	3.48e+03(1.46e+05)	3.64e+02(5.91e+02)	**2.00e+02(6.25e+02)**	5.67e+03(2.38e+05)	7.62e+02(8.12e+02)	**3.70e+02(1.58e+02)**
FDA5$_{iso}$	1.36e+03(8.46e+03)	4.18e+02(4.03e+02)	**2.69e+02(1.64e+02)**	1.31e+03(1.35e+04)	4.18e+02(2.77e+02)	**2.69e+02(7.39e+01)**
FDA5$_{dec}$	3.03e+03(1.56e+04)	3.46e+02(8.70e+02)	**1.44e+02(2.95e+02)**	4.74e+03(5.48e+04)	7.01e+02(4.91e+03)	**3.11e+02(2.48e+02)**
DIMP2	9.62e+02(2.11e+04)	2.80e+01(2.25e+01)	**2.11e+01(1.29e+01)**	6.74e+03(1.78e+06)	1.49e+02(4.52e+02)	**7.74e+01(2.76e+02)**
DMOP2	7.26e+02(8.73e+03)	3.15e+02(1.35e+03)	**1.03e+02(1.43e+02)**	5.35e+03(2.68e+05)	3.69e+02(1.11e+03)	**2.05e+02(2.84e+01)**
DMOP2$_{iso}$	1.34e+03(5.58e+03)	5.97e+02(2.33e+04)	**4.44e+02(1.25e+03)**	1.44e+03(5.92e+03)	5.78e+02(2.20e+04)	**4.33e+02(1.47e+03)**
DMOP2$_{dec}$	5.81e+02(3.71e+03)	5.07e+01(5.06e+01)	**1.86e+01(3.25e+00)**	3.84e+03(6.01e+04)	3.47e+02(4.55e+02)	**1.97e+02(1.84e+01)**
DMOP3	8.64e+02(2.20e+04)	2.98e+02(1.46e+03)	**1.13e+02(2.51e+02)**	5.45e+03(4.75e+05)	3.88e+02(4.98e+02)	**1.99e+02(1.07e+02)**
HE2	2.11e+03(5.02e+04)	4.35e+02(5.15e+03)	**3.28e+02(3.65e+03)**	1.35e+04(1.52e+06)	2.29e+03(7.51e+04)	**1.55e+03(5.45e+04)**
HE7	5.84e+03(3.36e+05)	1.28e+03(5.17e+03)	**1.15e+03(6.44e+03)**	1.07e+04(1.94e+06)	2.81e+03(1.29e+04)	**2.47e+03(4.82e+04)**
HE9	2.91e+03(1.76e+05)	8.51e+02(5.40e+02)	**6.68e+02(1.71e+03)**	6.47e+03(7.73e+05)	1.79e+03(6.05e+02)	**1.50e+03(1.06e+04)**

There are 20 independent runs. Within each run, for each problem with each parameter setting, there are 20 changes. After each change, three optimization algorithms (interior point, sqp and active set) are used to find the minimal distance, respectively. The computation time of each algorithm is recorded for each change. The values in this table are the mean and standard deviation of total computation time of 20 changes under 20 runs. The smallest computation time for each problem is highlighted in bold.

1) The population size is set as 200, as in previous work [7];
2) In three Tr-DMOEA variants, three different optimization methods interior point, active set and sqp methods are used, respectively. The other steps of the Tr-DMOEA are the same as in the previous work [7].
3) RMMEDA [16] as the optimization algorithm. RMMEDA is a regularity model-based multi-objective estimation of distribution algorithm. It is able to make the population converge quickly before the next change, avoiding the cases that unconverged solutions affect the results. The state-of-the-art Tr-DMOEA [7] is adopted.

The computation time of three different optimization methods in these three Tr-DMOEA variants is only recorded when solving the inner problem (equation (2)), for all test problems with all parameter setting. The specific computation time comparisons of these three optimization methods in DMO, the comparison results of computation time of three Tr-DMOEA variants are shown in Table III. Due to space limitations, only results of C5 and C6 are presented. Results of other parameter settings are similar to those of C5 and C6.

It is clear from Table III that for all test problems with those parameter settings, the interior point method consumes the most time among the three compared methods to solve the inner optimization problem, while the active set method is the most efficient optimization algorithm. In addition, the time interior points method consumes is several times over that another two methods consume. This shows that another two inner optimization methods can greatly improve the efficiency of transfer learning in DMO.

Friedman and Nemenyi statistical tests [19] were carried out across benchmark problems. The computation time that all algorithms get on one problem with one parameter setting is regarded as an observation of the test. Therefore, there are 96 (12 problems and 8 parameters) observed data. Friedman detects significant differences in median accuracy with a p-value of 2.0311e-42. The Nemenyi post-tests are shown in Figure 1. This shows that the sqp and active set method are significantly more efficient than the interior point method.

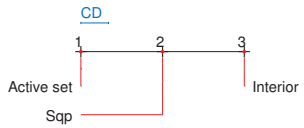

Fig. 1. Friedman ranking among computation time of three optimization methods (active set, sqp and interior point) from left to right. Friedman test's p-value is 2.0311e-42. Any pair of approaches whose distance between them is larger than CD are considered to be significantly different based on the Nemeyi post-hoc test.

IV. ANALYSIS OF THREE TR-DMOEA VARIANTS

In order to evaluate the influence of the improved efficiency on the solution quality, in this section, transferred solutions of these Tr-DMOEAs in the first generation after change are first compared. The optimized solution of these Tr-DMOEAs in the last generation after change are then compared.

A. Solutions Quality Comparison in the First Generation

Inverted Generational Distance (IGD) [20] is used to compare the quality of solution sets obtained by these three Tr-DMOEAs. IGD can measure the diversity and convergence of a solution set found by an algorithm, so it can give us a comprehensive understanding about the performance of the compared algorithms. MIGD [20] is a modified version of IGD, which is the average IGD values in all changes. The smaller the MIGD the better the algorithm.

For each Tr-DMOEA on one problem with one parameter setting, in the first generation after each change, a solution set obtained by transfer learning in the experiment of section III-B is used to calculate the IGD value. MIGD is the average of these 20 IGD values under 20 changes. Therefore, these three Tr-DMOEA variants will obtain one MIGD value on each problem for each parameter. Each algorithm is independently run 20 times on each problem instance with each parameter setting. The mean and standard deviation of MIGD values of transferred solutions obtained by the three Tr-DMOEAs in the first generation after changes are shown in Table IV.

In order to show the significant superiority of one method to others, Friedman and Nemenyi statistical tests were carried out across benchmark problems following Demsar's recommendation [19]. The MIGD values that all algorithms get on

TABLE IV
MEAN AND STANDARD DEVIATIONS OF MIGD VALUES OF **TRANSFERRED SOLUTIONS IN THE FIRST GENERATION** AFTER CHANGE OBTAINED BY THREE TR-RMMEDA VARIANTS WITH DIFFERENT INNER OPTIMIZATION ALGORITHMS.

Prob.	C5			C6		
Methods	Interior	sqp	active	Interior	sqp	active
FDA4	**9.85e-02(3.99e-06)**	1.00e-01(6.48e-06)	1.03e-01(1.32e-05)	**1.06e-01(7.04e-06)**	1.12e-01(1.35e-05)	1.18e-01(2.15e-05)
FDA5	1.28e+00(7.92e-03)	6.23e-01(1.65e-03)	**5.72e-01(1.46e-03)**	1.22e+00(8.41e-03)	5.15e-01(7.53e-04)	**5.07e-01(3.91e-04)**
$FDA5_{iso}$	**5.39e-01(2.67e-03)**	5.42e-01(6.59e-03)	6.79e-01(4.93e-03)	**5.07e-01(1.15e-03)**	5.31e-01(2.75e-03)	6.42e-01(5.70e-03)
$FDA5_{dec}$	2.11e+00(4.11e-02)	1.13e+00(1.44e-02)	**9.91e-01(1.92e-02)**	2.69e+00(4.38e-02)	6.93e-01(7.07e-04)	**6.36e-01(2.28e-03)**
DIMP2	**1.53e+01(2.25e-01)**	1.79e+01(4.41e-01)	1.69e+01(2.74e-01)	**1.21e+01(1.53e-01)**	1.44e+01(9.16e-02)	1.44e+01(2.01e-01)
DMOP2	3.92e+01(3.28e-01)	**1.82e+01(2.64e-04)**	1.82e+01(4.64e-04)	3.52e+01(1.01e+00)	1.81e+01(2.52e-04)	**1.81e+01(1.85e-04)**
$DMOP2_{iso}$	**8.50e-02(1.29e-05)**	8.85e-02(8.31e-06)	8.68e-02(6.61e-06)	**1.07e-01(2.36e-05)**	1.10e-01(7.90e-06)	1.10e-01(1.18e-05)
$DMOP2_{dec}$	**8.86e+00(2.38e+00)**	1.57e+05(4.70e+11)	8.08e+05(1.95e+12)	6.72e+00(4.26e-01)	**2.76e+00(4.07e+00)**	2.85e+00(4.32e+00)
DMOP3	3.70e+01(3.75e-01)	**1.82e+01(4.23e-04)**	1.82e+01(5.34e-04)	3.39e+01(8.30e-01)	1.82e+01(5.21e-04)	**1.82e+01(5.19e-04)**
HE2	7.20e-01(4.59e-02)	5.89e-01(1.69e-02)	**4.72e-01(2.68e-02)**	2.36e-01(9.37e-05)	2.81e-01(2.57e-04)	**2.25e-01(3.64e-04)**
HE7	**3.05e-01(3.48e-05)**	3.46e-01(6.97e-05)	3.48e-01(6.82e-05)	**2.85e-01(2.29e-05)**	3.18e-01(2.12e-05)	3.17e-01(3.26e-05)
HE9	3.12e-01(5.27e-05)	**3.07e-01(1.73e-05)**	3.19e-01(8.23e-05)	**2.69e-01(3.58e-05)**	2.84e-01(1.08e-05)	2.85e-01(1.55e-05)

For each problem with each parameter setting, three Tr-RMMEDA variants with different inner optimization methods (interior point, sqp and active set) get initial populations after each change. MIGD is the average of IGDs of these initial populations with 20 independent runs under 20 changes. The better values with smaller average that the method gets are highlighted in bold face.

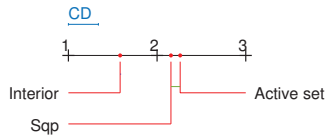

Fig. 2. Friedman ranking among MIGD values of three transferred solution sets obtained by Tr-DMOEAs with different inner optimization methods (interior point, sqp and active set) in the first generation after change from left to right. Friedman test's p-value is 1.7723e-6. Any pair of approaches whose distance between them is larger than CD are considered to be significantly different based on the Nemeyi post-hoc test.

one problem with one parameter setting are regarded as an observation of the test. Therefore, there are 96 (12 problems and 8 parameters) observed data. Friedman detects significant differences in average accuracy with a p-value of 1.7723e-6. The Nemenyi post-tests are shown in Figure 2.

Only results of C5 and C6 are presented due to space limitation. It has been observed that in most cases the Tr-DMOEA with the interior point method achieves the best results, compared with the two other Tr-DMOEA variants. The Friedman and Nemenyi statistical tests show that Tr-DMOEA with interior point significantly outperforms the other two variants.

B. Solutions Quality Comparison after Optimization

Similarly, in the experiment of section III-B, each Tr-DMOEA will get a solution set on one problem with one parameter setting, after optimization for τ_t generations. Each algorithm is independently run 20 times on each problem instance with each parameter setting. The obtained solution set is used to calculate the IGD value. Also, the IGD values of 20 changes are averaged to get the MIGD for each problem with one parameter. The mean and standard deviation of MIGD values of transferred solutions obtained by three Tr-DMOEAs in the last generation after change are shown in Table V.

Friedman and Nemenyi statistical tests [19] were carried out across benchmark problems. The MIGD values that all algorithms get on one problem with one parameter setting are regarded as an observation of the test. Therefore, there are 96 (12 problems and 8 parameters) observed data. Friedman detects significant differences in average accuracy with a p-value of 1.1162e-4. The Nemenyi post-tests are shown in Figure 3.

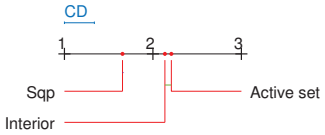

Fig. 3. Friedman ranking among MIGD values of three optimized solution sets obtained by Tr-DMOEAs with different inner optimization methods (sqp, interior point and active set) in the last generation after change from left to right. Friedman test's p-value is 1.1162e-4. Any pair of approaches whose distance between them is larger than CD are considered to be significantly different based on the Nemeyi post-hoc test.

Only results of C5 and C6 are presented in Table V. It has been observed that in most cases the Tr-DMOEA with sqp method achieves the best results, compared with two other Tr-DMOEA variants. The Friedman and Nemenyi statistical tests show that Tr-DMOEA with sqp significantly outperforms the other two variants.

After getting the computation time and solution quality comparison results of three Tr-DMOEA variants, it can be concluded that the interior point method in the original Tr-DMOEA is ineffective and extremely inefficient. Although interior point method can achieve the best transferred solutions among three compared methods, it consumes several or even more than ten times of what two other methods consume. In addition, transferred solutions found by interior point method become significantly worse than those found by sqp method after optimization, which shows that it is not beneficial for the interior point method to improve the efficiency of optimization, compared with sqp method.

Considering the influence of τ_t on the quality of optimized solutions obtained by three Tr-DMOEAs, it is clear from Table V that when changes are medium (C1 - C4), the solution quality of that with sqp increases as τ_t increases from 5 to 10 and to 25. However, when τ_t = 50, Tr-DMOEA with interior point method remains the best, the same as what these methods perform in the first generation after change. As for another two cases when changes are huge (C5 - C6) and small (C7 - C8), the Tr-DMOEA with interior point becomes worse while

TABLE V
MEAN AND STANDARD DEVIATIONS OF MIGD VALUES OF **OPTIMIZED SOLUTIONS IN THE LAST GENERATION** AFTER CHANGE OBTAINED BY THREE TR-RMMEDA VARIANTS WITH DIFFERENT INNER OPTIMIZATION ALGORITHMS.

Prob.	C5			C6		
Methods	Interior	sqp	active	Interior	sqp	active
FDA4	5.06e-02(8.71e-09)	**4.87e-02(1.80e-08)**	4.97e-02(2.80e-08)	5.65e-02(8.32e-09)	**5.53e-02(1.22e-08)**	5.60e-02(9.64e-09)
FDA5	5.33e-01(1.27e-04)	3.84e-01(8.14e-06)	**3.79e-01(2.42e-05)**	**1.91e-01(3.94e-06)**	2.08e-01(1.19e-05)	2.31e-01(8.89e-06)
$FDA5_{iso}$	2.22e-01(4.64e-06)	**1.83e-01(1.02e-05)**	2.05e-01(2.02e-05)	1.61e-01(1.42e-06)	**1.51e-01(5.54e-06)**	1.55e-01(3.01e-06)
$FDA5_{dec}$	8.39e-01(1.48e-03)	**6.48e-01(5.58e-05)**	6.53e-01(3.82e-04)	**2.79e-01(7.93e-06)**	3.51e-01(2.10e-05)	3.53e-01(6.43e-06)
DIMP2	**7.28e+00(7.10e-04)**	7.93e+00(4.81e-03)	8.34e+00(4.94e-03)	4.21e+00(2.69e-04)	**4.11e+00(3.85e-03)**	4.29e+00(1.44e-04)
DMOP2	2.36e+01(1.07e-02)	**1.80e+01(1.24e-07)**	1.81e+01(2.31e-06)	1.85e+01(1.15e-02)	1.80e+01(7.61e-09)	**1.80e+01(1.48e-10)**
$DMOP2_{iso}$	**8.98e-02(1.90e-10)**	9.04e-02(1.94e-10)	9.05e-02(9.52e-10)	**1.12e-01(8.90e-11)**	1.12e-01(3.06e-10)	1.12e-01(1.22e-10)
$DMOP2_{dec}$	1.20e+00(4.20e-03)	**3.50e-01(8.52e-05)**	4.16e-01(1.15e-03)	2.37e-01(4.19e-05)	**1.11e-01(3.23e-08)**	1.18e-01(3.16e-05)
DMOP3	2.32e+01(1.65e-02)	**1.80e+01(2.98e-07)**	1.80e+01(1.93e-06)	1.86e+01(3.18e-03)	**1.80e+01(1.53e-11)**	1.80e+01(1.36e-08)
HE2	3.27e-01(1.26e-04)	2.28e-01(2.91e-04)	**1.88e-01(2.47e-04)**	7.80e-02(3.22e-07)	7.12e-02(2.55e-06)	**6.78e-02(2.48e-06)**
HE7	1.12e-01(4.66e-07)	1.11e-01(4.26e-07)	**1.09e-01(2.36e-07)**	4.82e-02(4.88e-08)	**4.65e-02(3.88e-08)**	4.66e-02(2.40e-08)
HE9	2.48e-01(3.91e-07)	**2.39e-01(2.06e-07)**	2.44e-01(3.44e-06)	2.21e-01(5.59e-08)	**2.17e-01(6.19e-08)**	2.18e-01(1.94e-07)

For each problem with each parameter setting, three Tr-RMMEDA variants with different inner optimization methods (interior point, sqp and active set) get optimized populations by RMMEDA in the last generation of each change. MIGD is the average of IGDs of these populations with 20 independent runs under 20 changes. The better values with smaller average that the method gets are highlighted in bold face.

another two variants become better.

V. SOLUTIONS QUALITY AFTER OPTIMIZATION FOR WHICH TRANSFER LEARNING COST IS USED

It has been experimentally shown that the existing Tr-DMOEA is extremely time-consuming. It is still unclear whether it is worthy to consume such long time to use transfer learning in DMO. This section is presented to explore this.

A. Computation Time of Transfer Learning Used for Optimization

In order to figure out whether it is worthwhile to use transfer learning in DMO, this section designs an experiment to verify it. The main idea behind this experiment is to use the computation time of transfer learning to optimize randomly generated solutions.

Whenever there is a change, transfer leaning is used to get the transferred initial population, while another initial population is randomly generated in the search space. The costs of transfer learning and random generation are recorded, termed as C_{tr} and C_{ran}. The cost is the running time determined based on the stopwatch timer in Matlab, where the Matlab command 'tic' and 'toc' starts and ends the timer, respectively. Then, the cost C_{tr} - C_{ran} is used to optimize the randomly generated population. During the optimization, both transferred population and random one will iterate for τ_t generations to get two optimized solutions.

All test problems in [4] and all parameter settings in II are used, which is the same as those in III-B. Also, for each test problems, there are 20 environmental changes. In addition, at the beginning of the algorithm, the population iterates for 50 generations, which enables the population to converge. Each algorithm is independently run 20 times on each problem instance with each parameter setting.

The sepecific experiment setup is as follows:

- The population size is set as 200, as in previous work [7];
- In Tr-DMOEA, all the transfer learning procedures including the interior point method are used, which are the same as the original.
- RMMEDA [16] is used as the optimization algorithm.

B. Quality Comparison of Transferred And Random Solutions with Same Cost Budget

The specific experimental design has been introduced in the previous section. In this section we compare the quality of optimized solutions with transfer and optimized solutions from random generation, both of which are obtained under the same computation time budget.

a) Solution Quality Comparison in the First Generation: Here, we compare the quality of transferred solutions and those solutions which are optimized using the transfer learning computation time. IGD is used to measure the diversity and convergence of those two solution sets. MIGD is the average of these 20 IGD values under 20 changes for two solution sets. The mean and standard deviation of MIGD values are presented in Table VI, in which 'Transfer' and 'Optimize' refer to the MIGD values of these two solution sets. The better values that the method gets are highlighted in bold face. In order to indicate the significance between transferred solutions and optimized solutions using transfer learning computation time, the Wilcoxon rank sum test with the significance level 0.05 is carried out across problem instances, as recommended by Demsar [19]. MIGD value of 'Transfer' and 'Optimize' for each problem with one parameter is regarded as one observation data for the test. The result with h = 1 and p = 1.2624e-08 shows that random solutions optimized using transfer learning computation time are significantly better than transferred solutions.

b) Solution Quality Comparison after Optimization: Here, we compare the quality of optimized solutions with transfer learning and those solutions which originate from random generation. Also, IGD is used to measure the diversity and convergence of those two solution sets. MIGD is the average of these 20 IGD values under 20 changes for two solution sets. The mean and standard deviation of MIGD values are presented in Table VII, in which 'Transfer' and 'Optimize' refer to the MIGD values of these two solution sets. The better values that the method gets are highlighted in bold face.

TABLE VI

MEAN AND STANDARD DEVIATIONS OF MIGD VALUES OF TRANSFERRED SOLUTIONS AND THOSE SOLUTIONS OPTIMIZED USING THE COMPUTATION TIME OF TRANSFER LEARNING AFTER BEING RANDOMLY GENERATED.

Prob.	C1		C2		C3		C4	
Methods	Transfer	Random	Transfer	Random	Transfer	Random	Transfer	Random
FDA4	8.80e-02(2.50e-04)	**7.30e-02(2.45e-08)**	8.20e-02(2.05e-04)	**7.27e-02(2.01e-07)**	9.72e-02(3.95e-04)	**7.29e-02(5.38e-09)**	8.53e-02(2.38e-04)	**7.30e-02(4.06e-09)**
FDA5	2.62e-01(1.83e-03)	**1.77e-01(4.85e-06)**	2.03e-01(1.68e-03)	**1.73e-01(8.88e-07)**	1.94e-01(1.61e-03)	**1.75e-01(3.22e-06)**	1.89e-01(2.54e-03)	**1.74e-01(2.77e-06)**
FDA5$_{iso}$	**5.80e-02(6.97e-05)**	2.59e-01(1.57e-06)	**5.91e-02(3.24e-05)**	2.56e-01(1.67e-06)	**6.15e-02(5.93e-05)**	2.69e-01(2.35e-06)	**6.18e-02(5.33e-05)**	2.52e-01(1.85e-06)
FDA5$_{dec}$	5.41e-01(2.23e-02)	**2.06e-01(7.60e-06)**	5.79e-01(2.71e-02)	**1.98e-01(4.12e-06)**	5.20e-01(1.16e-02)	**2.00e-01(3.92e-07)**	5.13e-01(9.76e-03)	**1.90e-01(1.41e-06)**
DIMP2	9.61e+00(2.37e+00)	**3.57e+00(9.26e-04)**	9.97e+00(1.07e+00)	**3.53e+00(6.77e-05)**	9.03e+00(3.57e+00)	**3.54e+00(5.98e-04)**	9.77e+00(1.32e+00)	**3.56e+00(6.11e-04)**
DMOP2	4.05e-01(3.69e-02)	**1.46e-02(1.59e-04)**	3.02e-01(1.78e-02)	**1.81e-02(2.69e-04)**	2.34e-01(5.82e-03)	**5.12e-03(4.52e-06)**	2.29e-01(5.17e-03)	**1.15e-02(8.32e-05)**
DMOP2$_{iso}$	**2.19e-03(7.48e-09)**	4.45e-03(3.35e-10)	**2.32e-03(7.88e-08)**	4.44e-03(6.10e-10)	**2.49e-03(1.48e-07)**	4.44e-03(1.48e-10)	**2.55e-03(1.35e-07)**	4.44e-03(2.34e-10)
DMOP2$_{dec}$	6.45e-01(1.24e-01)	**2.54e-02(1.87e-05)**	5.12e-01(4.01e-02)	**2.28e-02(2.81e-05)**	3.48e-01(5.99e-03)	**2.35e-02(2.17e-05)**	2.94e-01(5.18e-03)	**1.48e-02(4.17e-06)**
DMOP3	3.14e-01(5.55e-02)	**3.28e-03(5.27e-11)**	1.38e-01(3.28e-03)	**3.28e-03(5.63e-11)**	1.39e-01(1.32e-02)	**3.29e-03(1.43e-11)**	7.20e-02(4.84e-03)	**3.30e-03(8.28e-11)**
HE2	4.59e-01(2.49e-02)	**5.98e-02(6.63e-08)**	4.97e-01(1.38e-02)	**5.89e-02(2.62e-08)**	3.23e-01(1.07e-03)	**5.77e-02(5.15e-08)**	2.70e-01(5.19e-03)	**5.66e-02(8.82e-10)**
HE7	2.24e-01(1.94e-04)	**3.70e-02(1.41e-08)**	2.29e-01(2.97e-04)	**3.72e-02(3.44e-09)**	2.48e-01(3.49e-04)	**3.70e-02(1.38e-08)**	2.64e-01(3.56e-04)	**3.71e-02(7.82e-10)**
HE9	3.30e-01(2.51e-04)	**2.34e-01(7.89e-08)**	3.06e-01(4.54e-04)	**2.34e-01(1.43e-07)**	2.83e-01(2.29e-04)	**2.33e-01(1.82e-07)**	2.80e-01(4.85e-04)	**2.32e-01(7.33e-08)**
Prob.	C5		C6		C7		C8	
Methods	Transfer	Random	Transfer	Random	Transfer	Random	Transfer	Random
FDA4	7.23e-02(1.18e-04)	**5.99e-02(4.24e-09)**	6.86e-02(1.20e-04)	**6.01e-02(5.67e-09)**	**7.01e-02(1.55e-04)**	7.32e-02(5.29e-09)	7.35e-02(3.09e-04)	**7.33e-02(5.46e-09)**
FDA5	1.35e+00(4.76e-01)	**1.43e-01(4.38e-07)**	1.86e+00(2.89e-01)	**1.44e-01(1.24e-06)**	1.24e+00(5.75e-01)	**1.70e-01(5.22e-07)**	6.70e-01(9.62e-03)	**1.74e-01(7.32e-06)**
FDA5$_{iso}$	5.33e-01(1.37e-02)	**1.78e-01(1.47e-05)**	5.44e-01(6.02e-02)	**1.85e-01(6.78e-06)**	6.49e-01(2.01e-02)	**2.51e-01(9.83e-06)**	5.51e-01(4.31e-03)	**2.49e-01(9.90e-06)**
FDA5$_{dec}$	2.28e+00(2.43e-01)	**1.59e-01(9.46e-06)**	2.33e+00(2.77e-01)	**1.59e-01(6.53e-06)**	2.14e+00(3.27e-01)	**1.87e-01(1.27e-06)**	1.21e+00(2.80e-01)	**1.90e-01(7.73e-06)**
DIMP2	1.09e+01(9.25e-01)	**3.84e+00(1.07e-04)**	9.81e+00(9.88e-01)	**3.83e+00(1.06e-04)**	1.30e+01(1.59e+00)	**3.78e+00(3.37e-04)**	1.27e+01(1.63e+00)	**3.78e+00(6.19e-04)**
DMOP2	1.10e+02(7.79e+01)	**1.81e+01(4.03e-04)**	1.13e+02(1.15e+02)	**1.80e+01(8.19e-05)**	2.70e+00(2.18e+00)	**5.15e-03(4.60e-06)**	3.64e+00(1.41e+00)	**5.26e-03(4.97e-06)**
DMOP2$_{iso}$	4.36e-01(5.21e-08)	**9.08e-02(4.66e-10)**	4.36e-01(9.49e-08)	**1.12e-01(8.16e-11)**	**2.67e-03(2.45e-07)**	4.44e-03(1.10e-10)	**2.89e-03(5.71e-07)**	4.45e-03(9.63e-11)
DMOP2$_{dec}$	5.38e+00(3.82e+01)	**1.74e-01(7.42e-10)**	3.94e+00(5.68e+00)	**1.94e-01(1.38e-10)**	2.04e+00(8.07e-01)	**2.33e-02(7.25e-05)**	2.69e+00(2.12e+00)	**1.71e-02(1.12e-05)**
DMOP3	1.12e+02(4.89e+01)	**1.80e+01(2.02e-08)**	1.13e+02(2.98e+02)	**1.80e+01(4.12e-09)**	1.44e+00(8.44e-01)	**3.33e-03(2.49e-11)**	2.22e+00(9.56e-01)	**3.30e-03(6.02e-11)**
HE2	1.25e-01(7.15e-04)	**5.64e-02(1.19e-08)**	1.21e-01(5.20e-04)	**5.49e-02(4.11e-10)**	6.06e-01(5.69e-03)	**5.84e-02(3.57e-08)**	5.02e-01(2.14e-03)	**5.66e-02(6.04e-10)**
HE7	1.72e-01(1.14e-03)	**3.42e-02(1.17e-08)**	1.82e-01(1.94e-04)	**3.42e-02(2.40e-08)**	2.53e-01(4.60e-04)	**3.70e-02(4.02e-09)**	2.73e-01(5.03e-04)	**3.70e-02(1.33e-08)**
HE9	2.76e-01(1.01e-04)	**2.09e-01(2.78e-07)**	2.58e-01(6.38e-04)	**2.08e-01(1.83e-07)**	3.19e-01(7.59e-05)	**2.34e-01(2.96e-08)**	2.79e-01(2.94e-04)	**2.31e-01(1.58e-07)**

There are 20 independent runs. Within each run, for each problem with each parameter setting, one solution set is the transferred solutions. Another solution set is obtained by optimization on randomly generated solutions using the computation time that transfer learning consumes. The better values with smaller average that the method gets are highlighted in bold face.

In order to indicate the significance between optimized solutions with transfer learning and those from random generation, the Wilcoxon rank sum test with the significance level 0.05 is carried out. MIGD value of 'Transfer' and 'Optimize' for each problem with one parameter is regarded as one observation data for the test. The result with h = 1 and p = 0.0085 shows that solutions which originates from random generation are significantly better than optimized solutions with transfer learning in the last generation of optimization.

It can be concluded that under the same computation time budget, random solution after optimization will obtain better results than those from transfer learning, no matter whether the optimization process is conducted on transferred solutions or not. In the original paper, Tr-DMOEA is better than the algorithm with random solutions. The difference is that in the original paper, the same evaluation times and generations are given to these two algorithms, while the same computation time budget is given in this paper. This shows that it is not worth to consume such long time on transfer learning, while achieving worse results than consuming these computation time on optimizing. In other words, it vanishes the original purpose of using transfer learning in DMO, reducing the efforts of optimization.

On the other side, when comparing Tables VI and VII, it is clear that the improvement of transferred solutions are more than that of optimized solution from randomization after τ_t generations' optimization. In addition, as τ_t increases, the quality of transferred solutions becomes better while those from randomization remain unchanged. These two observations show that transferred solutions are far from the optimum of problems, while solutions from randomization have already been nearly Pareto optimal.

VI. CONCLUSION

This paper studies how efficient transfer learning is in DMO through time complexity and computation time analysis, showing that the interior point method in transfer learning [7] is extremely time-consuming when solving the inner problem (equation (2)). Another two inner optimization methods are computationally studied, to figure out whether the efficiency of transfer learning can be improved and whether the improved efficiency will affect the effectiveness. Experimental results shows that the sqp method achieve a better balance between the transfer learning efficiency and effectiveness of transferred solutions. Lastly, another experiment is conducted to verify whether the purpose of using transfer learning in DMO vanishes, which leverages the computation time of transfer learning to optimize randomly generated solutions. The results show that randomly generated solutions after being optimized using transfer learning time are greatly better than transferred ones, which therefore encourages the proposal of more efficient transfer learning algorithms for DMO. Even though the existing Tr-DMOEA is computationally consuming, it should be noted that it could still be a good alternative to solve problems with extremely expensive objective functions [21].

In the future, a potential work is to find another optimization algorithm to solve the inner problem (equation (2)) efficiently and effectively. In addition, other transfer learning methods in the field of machine learning can also be studied to solve DMOPs. The efficiency of transfer learning should be also considered. At last, it is important to explore real-world applications of transfer learning based DMO, e.g., in smart

TABLE VII

MEAN AND STANDARD DEVIATIONS OF MIGD VALUES OF OPTIMIZED SOLUTIONS WITH TRANSFER LEARNING AND RANDOM SOLUTIONS THAT FIRST ITERATE FOR SOME GENERATIONS WHICH CONSUME THE SAME COMPUTATION TIME AS TRANSFER LEARNING, AND THEN ARE OPTIMIZED THE SAME GENERATIONS AS THOSE WITH TRANSFER.

Prob.	C1		C2		C3		C4	
Methods	Transfer	Random	Transfer	Random	Transfer	Random	Transfer	Random
FDA4	**5.85e-02(6.23e-05)**	7.30e-02(2.52e-08)	**5.73e-02(4.51e-06)**	7.26e-02(3.46e-08)	**6.38e-02(6.46e-06)**	7.28e-02(5.53e-09)	**6.56e-02(1.88e-06)**	7.32e-02(4.77e-09)
FDA5	**1.17e-01(3.18e-04)**	1.82e-01(2.80e-06)	**7.23e-02(2.49e-05)**	1.74e-01(1.04e-06)	**7.33e-02(2.75e-06)**	1.82e-01(3.32e-07)	**7.77e-02(3.69e-06)**	1.76e-01(2.84e-06)
FDA5$_{iso}$	**6.12e-02(8.74e-06)**	2.61e-01(2.33e-06)	**7.45e-02(9.93e-06)**	2.57e-01(4.63e-06)	**8.48e-02(4.85e-06)**	2.77e-01(1.60e-05)	**8.76e-02(3.31e-06)**	2.62e-01(2.35e-06)
FDA5$_{dec}$	4.86e-01(9.24e-03)	**2.05e-01(4.34e-06)**	4.00e-01(8.08e-03)	**1.94e-01(3.20e-06)**	3.05e-01(5.72e-03)	**2.00e-01(1.61e-06)**	**1.91e-01(1.96e-03)**	1.93e-01(1.43e-06)
DIMP2	7.02e+00(8.61e-01)	**3.57e+00(9.26e-04)**	5.43e+00(4.27e-01)	**3.53e+00(6.75e-05)**	3.64e+00(1.26e-01)	**3.54e+00(5.96e-04)**	**3.18e+00(6.77e-03)**	3.56e+00(6.10e-04)
DMOP2	2.77e-01(2.33e-02)	**1.46e-02(1.57e-04)**	1.17e-01(1.35e-03)	**1.80e-02(2.68e-04)**	8.78e-03(4.43e-05)	**5.12e-03(4.50e-06)**	**4.17e-03(5.86e-06)**	1.15e-02(8.40e-05)
DMOP2$_{iso}$	**2.58e-03(5.02e-08)**	4.45e-03(1.12e-10)	**3.12e-03(1.45e-07)**	4.45e-03(1.06e-10)	**3.73e-03(6.53e-08)**	4.47e-03(1.32e-10)	**4.06e-03(2.81e-08)**	4.49e-03(1.42e-10)
DMOP2$_{dec}$	4.77e-01(4.35e-02)	**2.41e-02(1.87e-05)**	2.87e-01(2.41e-02)	**2.27e-02(2.69e-05)**	6.02e-02(4.78e-04)	**2.29e-02(2.37e-05)**	**1.27e-02(1.87e-04)**	1.48e-02(4.15e-06)
DMOP3	2.19e-01(4.21e-02)	**3.29e-03(3.28e-11)**	4.90e-02(3.69e-04)	**3.30e-03(3.16e-11)**	4.21e-03(5.04e-07)	**3.29e-03(3.09e-11)**	3.51e-03(5.93e-09)	**3.30e-03(1.43e-11)**
HE2	4.22e-01(1.83e-02)	**5.97e-02(7.59e-08)**	3.93e-01(8.25e-03)	**5.88e-02(2.45e-08)**	2.19e-01(2.25e-03)	**5.77e-02(4.34e-08)**	1.04e-01(6.91e-04)	**5.66e-02(4.21e-10)**
HE7	1.62e-01(9.35e-05)	**3.71e-02(1.93e-08)**	1.23e-01(8.81e-05)	**3.72e-02(4.49e-09)**	7.07e-02(5.22e-05)	**3.70e-02(3.81e-09)**	5.01e-02(9.82e-06)	**3.70e-02(9.77e-09)**
HE9	3.02e-01(1.43e-04)	**2.34e-01(7.38e-08)**	2.72e-01(1.82e-04)	**2.34e-01(1.30e-07)**	2.43e-01(4.81e-05)	**2.33e-01(1.45e-07)**	2.37e-01(2.59e-05)	**2.33e-01(1.64e-07)**
Prob.	C5		C6		C7		C8	
Methods	Transfer	Random	Transfer	Random	Transfer	Random	Transfer	Random
FDA4	**5.06e-02(2.99e-07)**	6.01e-02(1.04e-08)	**5.63e-02(2.88e-06)**	5.99e-02(1.14e-08)	**5.08e-02(1.33e-06)**	7.31e-02(7.13e-09)	**5.62e-02(1.89e-06)**	7.28e-02(2.19e-08)
FDA5	7.45e-01(5.23e-02)	**1.43e-01(3.21e-06)**	3.35e-01(3.64e-03)	**1.39e-01(2.60e-06)**	8.96e-01(2.55e-01)	**1.71e-01(1.07e-06)**	2.84e-01(2.00e-03)	**1.72e-01(6.59e-06)**
FDA5$_{iso}$	3.88e-01(8.47e-03)	**1.80e-01(1.55e-05)**	2.24e-01(1.08e-03)	**1.91e-01(4.60e-06)**	4.34e-01(6.13e-03)	**2.51e-01(1.01e-05)**	**2.48e-01(1.82e-03)**	2.61e-01(8.09e-06)
FDA5$_{dec}$	1.57e+00(4.95e-01)	**1.61e-01(2.65e-06)**	4.35e-01(9.41e-03)	**1.57e-01(3.00e-06)**	1.39e+00(3.41e-01)	**1.89e-01(5.32e-06)**	4.09e-01(9.33e-03)	**1.89e-01(1.93e-06)**
DIMP2	7.49e+00(7.21e-01)	**3.84e+00(1.03e-04)**	4.02e+00(3.28e-01)	**3.83e+00(1.06e-04)**	9.18e+00(1.04e+00)	**3.78e+00(3.36e-04)**	5.89e+00(1.17e-01)	**3.78e+00(6.20e-04)**
DMOP2	9.16e+01(4.48e+01)	**1.81e+01(4.05e-04)**	7.42e+01(1.45e+01)	**1.80e+01(8.33e-05)**	7.34e-01(1.39e-01)	**5.13e-03(4.52e-06)**	4.46e-02(1.11e-02)	**5.23e-03(4.74e-06)**
DMOP2$_{iso}$	4.36e-01(1.11e-08)	**9.08e-02(1.79e-10)**	4.36e-01(1.96e-09)	**1.12e-01(7.45e-11)**	**3.39e-03(1.99e-07)**	4.46e-03(4.45e-10)	**4.35e-03(7.25e-08)**	4.50e-03(2.86e-10)
DMOP2$_{dec}$	7.66e-01(1.60e+00)	**1.74e-01(4.99e-10)**	4.32e-01(8.19e-08)	**1.94e-01(1.74e-10)**	5.40e-01(3.17e-02)	**2.30e-02(6.80e-05)**	**8.76e-03(3.02e-05)**	1.68e-02(9.73e-06)
DMOP3	9.05e+01(1.17e+01)	**1.80e+01(1.38e-08)**	7.33e+01(1.01e+00)	**1.80e+01(1.28e-09)**	3.78e-01(7.07e-02)	**3.33e-03(1.51e-11)**	5.44e-03(3.04e-05)	**3.31e-03(2.85e-11)**
HE2	6.95e-02(1.68e-04)	**5.64e-02(1.11e-08)**	**5.36e-02(2.14e-06)**	5.49e-02(4.23e-09)	**5.38e-01(5.53e-03)**	5.84e-02(3.39e-08)	1.57e-01(1.19e-03)	**5.66e-02(5.72e-10)**
HE7	8.26e-02(1.95e-04)	**3.43e-02(7.46e-09)**	3.73e-02(1.17e-06)	**3.42e-02(2.07e-08)**	1.43e-01(2.91e-04)	**3.69e-02(3.33e-09)**	5.21e-02(8.34e-06)	**3.70e-02(2.96e-09)**
HE9	2.21e-01(1.54e-04)	**2.09e-01(2.57e-07)**	**1.80e-01(4.12e-05)**	2.08e-01(1.71e-07)	2.86e-01(2.78e-05)	**2.34e-01(3.44e-08)**	2.54e-01(9.71e-05)	**2.31e-01(1.70e-07)**

There are 20 independent runs. Within each run, for each problem with each parameter setting, 'Transfer' solution set means the optimized solutions with transfer learning. 'Random' solution set means random generated solutions that first iterate for some generations which consume the same computation time as transfer learning, and then are optimized the same generations as those with transfer. The better values with smaller average that the method gets are highlighted in bold face.

manufacturing and smart logistics.

ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186. The work was also supported by the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[2] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.

[3] B. Da, A. Gupta, and Y.-S. Ong, "Curbing negative influences online for seamless transfer evolutionary optimization," *IEEE Transactions on Cybernetics*, no. 99, pp. 1–14, 2018.

[4] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: test cases, approximations, and applications," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 5, pp. 425–442, 2004.

[5] J. Ou, J. Zheng, G. Ruan, Y. Hu, J. Zou, M. Li, S. Yang, and X. Tan, "A pareto-based evolutionary algorithm using decomposition and truncation for dynamic multi-objective optimization," *Applied Soft Computing*, vol. 85, p. 105673, 2019.

[6] S. Yang and X. Yao, *Evolutionary Computation for Dynamic Optimization Problems*. Springer, 2013, vol. 490.

[7] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 501–514, 2017.

[8] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff, and X. Yao, "When and how to transfer knowledge in dynamic multi-objective optimization," in *2019 IEEE SSCI*. IEEE, 2019, pp. 2034–2041.

[9] W. Zhenzhong, M. JIANG, G. Xing, F. Liang, H. Weizhen, and K. C. TAN, "Evolutionary dynamic multi-objective optimization via regression transfer learning," in *2019 IEEE SSCI*. IEEE, 2019, pp. 2375–2381.

[10] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, pp. 863–870.

[11] G. Ruan, G. Yu, J. Zheng, J. Zou, and S. Yang, "The effect of diversity maintenance on prediction in dynamic multi-objective optimization," *Applied Soft Computing*, vol. 58, pp. 631–647, 2017.

[12] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 1, pp. 40–53, 2013.

[13] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

[14] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A hilbert space embedding for distributions," in *International Conference on Algorithmic Learning Theory*. Springer, 2007, pp. 13–31.

[15] J. Shawe-Taylor, N. Cristianini *et al.*, *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[16] Q. Zhang, A. Zhou, and Y. Jin, "Rm-meda: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

[17] S. Wright and J. Nocedal, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[18] M. Helbig and A. Engelbrecht, "Benchmark functions for cec 2015 special session and competition on dynamic multi-objective optimization," *Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, Rep*, 2015.

[19] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.

[20] Q. Li, J. Zou, S. Yang, J. Zheng, and G. Ruan, "A predictive strategy based on special points for evolutionary dynamic multi-objective optimization," *Soft Computing*, vol. 23, no. 11, pp. 3723–3739, 2019.

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