Dynamic Multi-objective Optimization via Sliding Time Window and Parallel Computing

Qinqin Fan[1](✉), Yihao Wang[1], Okan K. Ersoy[2], Ning Li[3], and Zhenzhong Chu[4]

[1] Logistics Research Center, Shanghai Maritime University, Shanghai 201306, China
[2] School of Electronic and Computer Engineering, Purdue University, West Lafayette, IN 47906, USA
[3] Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai Jiao Tong University, Shanghai 200240, China
[4] Logistics Engineering College, Shanghai Maritime University, Shanghai 201306, China

Abstract. Tracking changing Pareto front (PF) in the objective space and Pareto set (PS) in the decision space is an important task in dynamic multi-objective optimization (DMO). Similarly, maintaining population diversity and reusing previous evolutionary information are useful to explore promising regions and to find high-quality solutions quickly in time-varying environments. To this end, a sliding time window based on parallel computing (STW-PC) is introduced in the present study. In the STW-PC, obtained time-sequence solution sets aim to preserve the diversity and facilitate a fast convergence since problems in successive time/environments are usually related. The parallel computing method is also employed to reduce the computational time. The STW-PC is incorporated into a multi-objective evolutionary algorithm and is compared with two competitors on 12 dynamic multi-objective optimization problems. The results show that the STW-PC can both improve the tracking performance of the selected algorithm in different degrees of changes, and significantly reduce the calculation time compared with transfer learning.

Keywords: Evolutionary computations · Dynamic multi-objective optimization · Sliding time window · Parallel computing · High-performing computing

1 Introduction

Dynamic multi-objective optimization problems (DMOPs) have been commonly found in various fields [1–6]. Compared with static optimization problems, their objective functions, constraints, decision space sizes or other parameters may change with time or under different environments. Therefore, two important tasks in the dynamic optimization are: (1) detect the change of optimization environments, and (2) find high-quality solutions quickly in dynamic or uncertain environments. For the former, how to effectively identify changes of models/environments is important. It should be noted that

Y. Tan and Y. Shi (Eds.): ICSI 2021, LNCS 12690, pp. 45–57, 2021.
https://doi.org/10.1007/978-3-030-78811-7_5

the change detection is not the focus in the current study. For the latter, the population diversity and the convergence speed are two important performance indicators for dynamic multi-objective evolutionary algorithms (DMOEAs). To improve the population diversity, diversity maintenance/improvement and multi-population [7] are two main approaches in the DMO. Moreover, memory-based and prediction-based methods are useful to promote convergence. Essentially, they belong to information reuse or knowledge transfer.

As stated above, the tracking performance of DMOEAs is impacted by population diversity and convergence speed, and thus these issues should be focused upon.

(1) Although various methods have been proposed to improve/maintain population diversity, the number of maintainable individuals is usually not enough in previous studies due to the limit of population/archive size [8]. Additionally, the current model may have a great correlation with recent "local" models on most actual dynamic optimization problems, but not with old "local" models. In other words, solutions of a dynamic optimization problem in recent successive environments are often relevant. Therefore, how to utilize the recent evolutionary information is an important step to find high-quality solutions quickly under dynamic environments.
(2) Prediction-based methods are promising for solving DMOPs, but most of them need to satisfy the independent identical distribution (IID) hypothesis [7]. Therefore, alleviating the limitation on the IID hypothesis is necessary. Additionally, their run time may be unacceptable in some cases, especially when the time complexity is high.

To solve the above-mentioned issues, a sliding time window based on parallel computing (STW-PC) is proposed to solve DMOPs in the current study. In the STW-PC, the sliding time window [9], which is an effective approach in predictive control, is used to save successive evolutionary information (i.e., trust regions [10]) with a random initial population to assist selected algorithms in improving their tracking performance in continuously changing environment. Intuitively, it can both improve the population diversity and speed up the convergence. Moreover, the parallel computing method is utilized to reduce its computational time. To demonstrate the effectiveness of the STW-PC, it is compared with transfer-learning-based MOEA and another MOEA on 12 DMOPs [11]. The results show that the STW-PC not only assists other algorithm in improving tracking performance, but also significantly reduces the computational time compared with the transfer learning method.

The remaining sections of this paper are organized as follows: the DMO, the performance metric, and the sliding time window are introduced in Sect. 2. Section 3 reviews related studies on DMOEAs. The STW-PC is presented in Sect. 4. Experimental results and analyses are reported in Sect. 5. Section 6 discusses conclusions and describes future studies.

2 Background

2.1 Dynamic Multi-objective Optimization

A dynamic multi-objective optimization problem can be defined as follows:

$$\min_{\mathbf{x}\in\boldsymbol{\Omega}} \boldsymbol{F}(\boldsymbol{x}, t) = \langle f_1(\boldsymbol{x}, t), f_2(\boldsymbol{x}, t), \ldots, f_m(\boldsymbol{x}, t)\rangle, \quad (1)$$

$$x_j \in \left(x_j^{\text{low}}, x_j^{\text{high}}\right),\ j = 1, 2, \ldots, D,$$

where $\boldsymbol{x}$ denotes a decision vector in the feasible search space $\Omega \subset R^D$; and t ($t = 1, 2, \ldots, T$) denotes a time/environment variable; x_j^{low} and x_j^{high} are the lower and upper bounds of x_i, respectively; D denotes the dimensionality of the DMOP and m represents the number of objectives. Compared with a static multi-objective optimization problem, the DMOP consists of m time-varying objective functions under different time or environments.

Definition 1 (Dynamic dominance relation): for each time step or environment, A vector $\boldsymbol{u} \in R^m$ is said to dominate another vector $\boldsymbol{v} \in R^m$, which is denoted as $\boldsymbol{u} \succ \boldsymbol{v}$, if $\forall n \in \{1, 2, \ldots, m\}$, $u_n \leq v_n$ and $\boldsymbol{u} \neq \boldsymbol{v}$.

Definition 2 (Dynamic Pareto optimal set): for each time step or environment, if there are no other solutions can be Pareto optimal to $\boldsymbol{x}^* \in R^D$, $\boldsymbol{x}^*$ is called the Pareto optimal solution. The set of all Pareto optimal solutions at t is called the Pareto set (PS), denoted as $\boldsymbol{X}_t^*$. The PS of the DMOP can be denoted as $\boldsymbol{X}^* = \boldsymbol{X}_1^* \bigcup \boldsymbol{X}_2^* \ldots \bigcup \boldsymbol{X}_T^*$.

Definition 3 (Dynamic Pareto front): for each time step or environment, the Pareto front (PF) can be defined as $PF_t = \left\{F(\boldsymbol{x}_t^*) \middle| \boldsymbol{x}_t^* \in \boldsymbol{X}_t^*\right\}$. Like Definition 2, the PF of the DMOP can be denoted as $PF = PF_1 \bigcup PF_2 \ldots \bigcup PF_T$. Similar to a static MOP, the main target of the DMO is to find a PF with good diversity and convergence at t.

2.2 Performance Metric

To evaluate the performance of DMOEAs, various performance metrics have been proposed [7, 12]. Overall, they adopt same performance indicators used in the static multi-objective optimization under each time step or environment. However, DMOPs are time-varying, thus using a cumulative performance is a common way to assess the overall performance of DMOEAs during all time steps or environments. Two IGD variants utilized in [7, 13] are introduced in this section.

An IGD variant (MIGD) is defined as follows:

$$MIGD\left(A, PF^*, C\right) = \frac{1}{|T|}\sum_{t\in T} \frac{\sqrt{\sum_{v\in PF_t^*} d(v, A_t)^2}}{\left|PF_t^*\right|}, \quad (2)$$

where A_t and PF_t^* denote the obtained PF approximation and a set of uniformly distributed points along the true PF at t, respectively; $d(v, A_t)$ is the minimum Euclidean distance between v and points in A_t; $\left|PF_t^*\right|$ represents the number of non-dominated individuals in PF_t^*; C denotes a parameter setting in the DMOP, and $|T|$ represents the number of time steps in T. A smaller value of the MIGD means that a DMOEA can achieve better PF approximations during all time steps.

2.3 Sliding Time Window

The STW, which divides the entire time window into $\triangle$ parts (called time slot), is an effective yet simple approach to solve dynamic optimization problems [14–18]. However, it has been utilized in the DMO rarely. In the STW, the size of the time window is the most important factor. If it is too large, a good result may be obtained but more computational resources are consumed. On the contrary, a small one reduces the computational budget, but it may significantly influence the performance of the STW. Therefore, its size should be set based on computational resources and final solution precision. We assume that the time window is $W(t, \triangle)$, $(1 \leq \triangle \leq T)$, which includes $\triangle$ time slots.

An example of the STW is illustrated in Fig. 1. Suppose $\triangle = 3$. When $t = t_0$, $W(t_0, \triangle) = \Phi$. If $t = t_1$, $W(t_1, \triangle)$ is equal to the first time slot. Like the above process, different time slots will be in W at different t. The number of time slots in W is equal to $\triangle$. It is observed from Fig. 1 that W is employed to store recent information or "local" models. That is why the STW can be utilized to deal with a large number of dynamic problems, especially when successive time steps/environments are relevant.

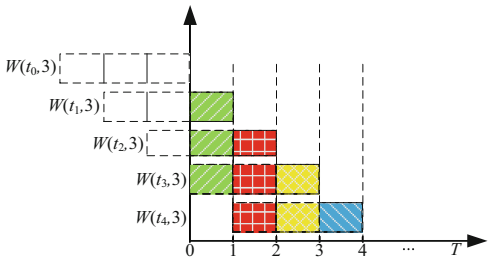


Fig. 1. Changes of the time window in the sliding time window when t is from t_0 to t_4 ($\triangle = 3$).

3 Related Work

Because objective functions or parameters in DMOPs will change over time or in different environments, their PFs or PSs, or both may be varying.

To improve the population diversity, researchers have proposed many advanced DMOEAs to solve DMOPs. Diversity-maintain/improvement-based and multi-population-based approaches are two main methods. For example, Deb et al. [19] proposed two dynamic NSGA-II variants to solve dynamic problems, in which randomly generated individuals and mutated individuals are used to replace some individuals in the current population. Yang [20] used memory-based and elitism-based immigrant methods to improve the search capability of the GA in a nonstationary environment. In this algorithm, the random immigrant approach is mainly used to increase the population diversity and the global exploration ability. However, Jiang et al. [21] pointed out that a population diversity improvement/maintain strategy may not be useful for solving complex dynamic optimization problems since it cannot provide available evolutionary information from past experiences. Besides improving the exploration capability of

search engines, a multi-population strategy is an effective approach to improve/maintain the population diversity in changing environments. For example, Branke et al. [22] used two populations to balance the exploration and exploitation capabilities. Different from the above studies, Liu et al. [23] used a co-evolutionary method to share information among different populations, and a similarity detection approach to detect the environmental changing. The results show that their proposed algorithm is a promising method to solve DMOPs.

To speed up convergence, memory-based and prediction-based methods have been proposed. In [24], an archive is used to save best individuals. If the environmental change is detected, some individuals in the current population would be replaced by historical individuals stored in the archive. Similar to the above work, a dynamic competitive-cooperation co-evolutionary algorithm (dCOEA) [25] is proposed to solve DMOPs. In the dCOEA, a temporal memory is utilized to store historical evolutionary information. Stroud [26] proposed a Kalman-extended GA (KGA) method to solve dynamic optimization problems, in which information provided by the Kalman formulation is employed to determine which operator should be adopted. In [27], the Kalman filter is used to learn previous evolutionary information and then produce useful candidate solutions. The experimental results show that the algorithm can help to improve the exploitation capability of evolutionary algorithms in a dynamic environment. Unlike previous studies, Zhou et al. [28] introduced a population prediction method to divide a Pareto set into two parts, and obtain a predicted center point and an estimated manifold. Therefore, this method can provide a good predicted population under different environments. Because the quality of an initial population is important for DMOEAs to find a high-quality solution set, a transfer learning approach [7] is utilized to generate an initial population at each time step. Their results confirm that their proposed algorithm can effectively reuse previous population information and assist MOEAs in tracking time-varying PFs or PSs or both. Ruan et al. [29] stated that an original transfer learning may not be an effective method to solve DMOPs in some cases. Therefore, novel strategies and kernel functions are studied. Recently, Wang et al. [30] proposed an ensemble learning based prediction strategy to enhance the prediction precision and improve the tracking performance of algorithms.

4 Sliding Time Window Based on Parallel Computing

To solve DMOPs effectively, maintaining the diversity and reusing historical information are important in DMOEAs. The STW is an effective approach to solve problems in an uncertain or dynamic environment, such as the predictive control and the modeling of time-varying systems. Therefore, the STW-PC is proposed in the present study. Its basic framework is shown in Algorithm 1.

At each time step t, the first step is to randomly generate an initial population in $\boldsymbol{\Omega}$, i.e. $\boldsymbol{P}_t$ (**line 2**), which is mainly used to improve the exploration capability of selected algorithms and provide additional evolutionary information. Subsequently, $\triangle$-1 obtained solution sets in time steps [t-$\triangle$ + 1, t-1] (denoted as $\boldsymbol{P}_{t-\Delta+1}, \boldsymbol{P}_{t-\Delta+2}, \ldots, \boldsymbol{P}_{t-1}$) and $\boldsymbol{P}_t$ stored in W are used as $\triangle$ initial populations at t. Meanwhile, a DMOEA/MOEA is employed to simultaneously find $\triangle$ PF approximations and PSs via the parallel computing technique (**line 3**). After finding all approximate PFs and PSs, they are merged to

obtain final approximate PF and PS using various multi-objective selection methods at t (**line 4**). The flowchart of the STW-PC is shown in Fig. 2.

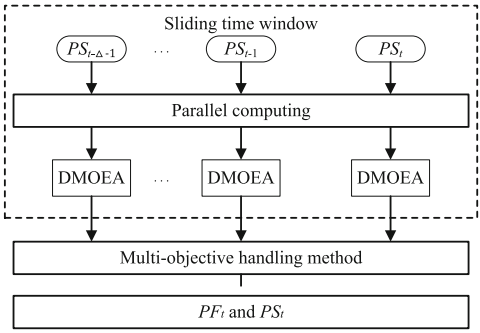


Fig. 2. Flowchart of the STW-PC.

Algorithm 1 STW-PC

Input: W: time window ;
$\triangle$: number of time slots in W ;
T: number of environment changes;

1: **for** t =1 to T **do**
2: Generate an initial population $\boldsymbol{P}_t$ in Ω;
Based on $\triangle$ initial populations in $W(t,\Delta)=\{\boldsymbol{P}_{t-\Delta+1},\boldsymbol{P}_{t-\Delta+2},\cdots,\boldsymbol{P}_t\}$, a selected
3: DMOEA/MOEA is used to find $\triangle$ approximate PFs at the same time via parallel computing technique;
4: Obtain $\boldsymbol{P}_t = Selection(\boldsymbol{P}_{t-\Delta+1}\cup\boldsymbol{P}_{t-\Delta+2}\cdots\cup\boldsymbol{P}_t)$ and $PF_t=\{F(x_t^*)|x_t^*\in\boldsymbol{P}_t\}$;
5: **end for**

Output: $PF=\{PF_1,PF_2\cdots,PF_T\}$ and $PS=\{\boldsymbol{P}_1,\boldsymbol{P}_2\cdots,\boldsymbol{P}_T\}$.

5 Experimental Results and Analyses

To demonstrate the effectiveness of the proposed STW-PC, a regularity model-based multi-objective estimation of distribution algorithm (RM-MEDA) [31], and a transfer-learning-based algorithm are selected in the current study. Moreover, the STW-PC is incorporated into the RM-MEDA (named as STW-PC*), and 12 bi- and tri-objective DMOPs proposed in IEEE CEC2015 are used. All compared algorithms are coded in Matlab and run on a Windows 10 operating system (64 bit).

The parameter configurations of 12 DMOPs are shown in Table 1. n_t, τ_T and τ_t denote the severity of change, the maximum number of generations, and the frequency of change, respectively. It is observed from Table 1 that each DMOP changes 20 times (i.e., $\frac{\tau_T}{\tau_t}$) during the whole time T. For all compared algorithms, the population size is set to 200, and the maximum numbers of generations are set to $20\times\tau_t+50$ for transfer-learning-based algorithm and $20\times\tau_t+200$ for the STW, respectively. The main reason is that, unlike the transfer learning, no useful information can be provided by the STW in the first time step. Therefore, giving more computational budgets can

improve the performance of the STW. However, from the perspective of computational time, the run time of the STW will not increase significantly, that is demonstrated in Table 1. Additionally, each compared algorithm is run for 20 independent times on all functions.

Table 1. Parameter configurations on 12 DMOPs

	n_t	τ_t	τ_T
C1	10	50	1000
C2	1	50	1000
C3	20	50	1000

5.1 Test the STW-PC Using MIGD

The mean MIGD values of all compared algorithms and the ratios of their improvement (ROI) are presented in Table 2. Note that the results of compared algorithms shown in Table 2 are directly taken from Ref. [7], except for the STW-PC*. The best mean value is highlighted in bold; [] denotes the value of ROI.

For the RM-MEDA, as shown in Table 2, the STW-PC* outperforms the RM-MEDA and the Tr-RM-MEDA on 35 and 31 cases, respectively. Also, it has a high success ratio to help the RM-MEDA solve DMOPs. Compared with the transfer learning method, the STW-PC is a more competitive approach to assist the RM-MEDA in adapting continuously changing environments. At the same time, the STW-PC* produces much better results (i.e., the ROI value is greater than 50%) on 13 and 6 cases when compared with the RM-MEDA and the Tr-RM-MEDA. Additionally, the Tr-RM-MEDA surpasses the STW-PC* on only one function HE7, in which PSs are the same during the whole time steps. The results presented in Table 2 indicate that the proposed algorithm slightly perform worse than the Tr-RM-MEDA on HE7 with three parameter configurations.

Based on the above comparisons, it can be concluded that the STW-PC is able to help other MOEAs/DMOEAs enhance their performances to track either changing PFs or changing PSs or both in continuously changing environments on different types of DMOPs.

5.2 Experimental Analysis

Impact of $\triangle$. Intuitively, a larger $\triangle$ value saves more evolutionary information in W, thus it provides more trust regions to guide the population evolution in a new optimization environment. To analyze the impact of $\triangle$, it was selected from the set $\{3, 4, 5, 6\}$, and the RM-MEDA and 12 DMOPs were used in experiments. Moreover, all parameter settings are the same as in the above experiments except for the $\triangle$ value. Note that the denominator of the ROI value is the MIGD calculated by $\triangle = 3$ in the following experiments.

Table 2. MIGD values of RM-MEDA, Tr-RM-MEDA, and STW-PC*

	C1			C2			C3		
	RM-MEDA	Tr-RM-MEDA	STW-PC*	RM-MEDA	Tr-RM-MEDA	STW-PC*	RM-MEDA	Tr-RM-MEDA	STW-PC*
FDA4	6.84E-02 **[30.4%]**	5.27E-02 **[9.7%]**	**4.76E-02**	6.93E-02 **[35.24%]**	5.01E-02 **[10.6%]**	**4.48E-02**	6.87E-02 **[32.2%]**	5.29E-02 **[11.9%]**	**4.66E-02**
FDA5	1.86E-01 **[63.3%]**	7.93E-02 **[14.0%]**	**6.82E-02**	2.68E-01 **[67.1%]**	**8.18E-02** **[-7.8%]**	8.82E-02	1.66E-01 **[60.7%]**	7.52E-02 **[13.2%]**	**6.53E-02**
FDA5$_{iso}$	6.59E-02 **[36.9%]**	6.61E-02 **[37.1%]**	**4.16E-02**	6.11E-02 **[37.3%]**	6.12E-02 **[37.4%]**	**3.83E-02**	6.53E-02 **[37.2%]**	6.54E-02 **[37.3%]**	**4.10E-02**
FDA5$_{dec}$	6.39E-01 **[87.1%]**	4.10E-01 **[80.0%]**	**8.25E-02**	3.80E-01 **[68.2%]**	1.31E-01 **[7.6%]**	**1.21E-01**	6.74E-01 **[87.9%]**	3.66E-01 **[77.8%]**	**8.14E-02**
DIMP2	4.66E + 00 **[20.4%]**	4.82E + 00 **[23.0%]**	**3.71E + 00**	5.01E + 00 **[21.8%]**	4.96E + 00 **[21.0%]**	**3.92E + 00**	4.88E + 00 **[19.5%]**	5.10E + 00 **[22.9%]**	**3.93E + 00**
DMOP2	4.10E-03 **[43.9%]**	2.70E-03 **[14.8%]**	**2.30E-03**	1.84E + 01 **[2.2%]**	1.87E + 01 **[3.7%]**	**1.80E + 01**	3.40E-03 **[38.2%]**	3.80E-03 **[44.7%]**	**2.10E-03**
DMOP2$_{iso}$	1.90E-03 **[56.4%]**	1.90E-03 **[56.4%]**	**8.29E-04**	**1.10E-01** **[0.0%]**	**1.10E-01** **[0.0%]**	**1.10E-01**	1.90E-03 **[56.3%]**	1.90E-03 **[56.3%]**	**8.30E-04**
DMOP2$_{dec}$	1.10E-01 **[90.0%]**	4.41E-02 **[75.1%]**	**1.10E-02**	2.30E-01 **[16.1%]**	2.21E-01 **[12.7%]**	**1.93E-01**	**1.07E-01** **[90.0%]**	5.40E-02 **[80.2%]**	**1.07E-02**
DMOP3	3.00E-03 **[30.0%]**	3.00E-03 **[30.0%]**	**2.10E-03**	1.83E + 01 **[1.6%]**	1.85E + 01 **[2.7%]**	**1.80E + 01**	3.30E-03 **[42.4%]**	2.40E-03 **[20.8%]**	**1.90E-03**
HE2	8.91E-01 **[93.0%]**	1.07E-01 **[41.9%]**	**6.22E-02**	6.97E-01 **[91.5%]**	7.26E-02 **[18.0%]**	**5.95E-02**	8.94E-01 **[93.1%]**	1.08E-01 **[43.1%]**	**6.15E-02**
HE7	4.40E-02 **[4.6%]**	**3.51E-02** **[-19.7%]**	4.20E-02	3.91E-02 **[11.5%]**	**3.33E-02** **[-3.9%]**	3.46E-02	4.36E-02 **[5.3%]**	**3.37E-02** **[-22.6%]**	4.13E-02
HE9	2.64E-01 **[11.0%]**	2.46E-01 **[4.5%]**	**2.35E-01**	2.36E-01 **[11.0%]**	2.18E-01 **[3.7%]**	**2.10E-01**	2.61E-01 **[10.0%]**	2.47E-01 **[4.9%]**	**2.35E-01**

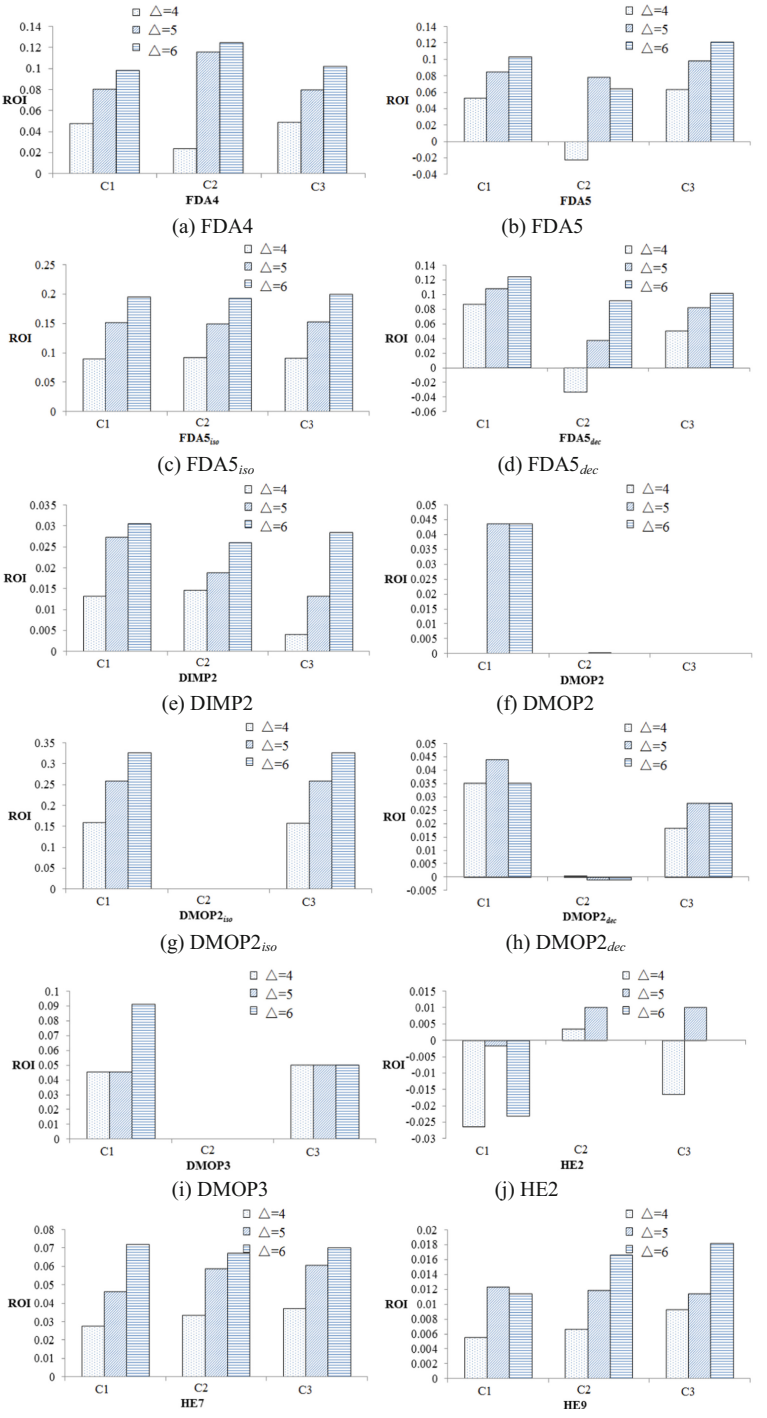
ROI values of all cases with three parameter configurations are illustrated in Fig. 3. From Fig. 3 (a)–(e), it is seen that a large $\triangle$ value is able to improve the tracking performance of the RM-MEDA. As shown in Fig. 3 (f), although a large $\triangle$ value can help the RM-MEDA adapt to changing environments when the parameter configuration is C1, $\triangle = 6$ does not produce better results than $\triangle = 5$. This is because the RM-MEDA can find high-quality solutions on DMOP2 with C1. Moreover, Fig. 3 (f) indicates the effectiveness of $\triangle$ is limited on environments C2–C3. The main reason may be that these two environments are hard for the RM-MEDA to search good results under limited computational resources, i.e., the maximum number of generations set to 50 is not enough for the RM-MEDA in each time step. It can be observed from Fig. 3 (g) that a large $\triangle$ value can enhance the tracking performance of the RM-MEDA when parameter configurations are C1 and C3. However, it cannot assist the RM-MEDA in improving the tracking performance when the parameter configuration is C2. This is because the STW-PC assumes that "local" models are related in successive time steps. If the severity of change is high, the performance of the STW–PC may reduce. In other words, historical individuals may not be able to provide valuable evolutionary information when environment is changing dramatically. Figures 3 (g) and (i) illustrate that a large $\triangle$ value does not improve the tracking performance of the RM-MEDA significantly. Like Fig. 3(g), the tracking performance of the proposed algorithm may be limited when the severity of environmental change is high such as C2. For HE1, HE7, and HE9, they belong to Type III, i.e., PSs are the same in different time steps. From Fig. 3 (j), we can see that a large $\triangle$ value may slightly reduce the performance of the STW-PC because a large $\triangle$ value will increase the number of individuals, which may influence the selection of individuals in PF. Additionally, it can be seen from Figs. 3 (k) and (l) that the performance of the STW–PC cannot be significantly influenced by values of $\triangle$ because the STW–PC can provide good initial population in each time step.

Based on the above observations and comparisons, the following conclusions are obtained:

(1) The STW–PC is an effective approach to solve DMOPs, especially when problems are related in successive time steps or environments.
(2) If the environment is changing dramatically, like other existing methods, the performance of the STW-PC will be reduced. Therefore, its performance is influenced by selected MOEAs/DMOEAs.

Computational Time of STW-PC* and Tr-RM-MEDA. In the STW-PC, obtained solution sets in W will be used as initial populations. However, enough time cannot be given before the next time step or environment change. Therefore, the computational time of the STW-PC is investigated on FDA4 with C1–C3 in this experiment. Additionally, the computational time of the Tr-RM-MEDA is also given. All experiments were ran on a laptop computer with Windows 10 operating system (64 bit) using MATLAB (R2016a). Moreover, two compared algorithms ran 20 times on each instance independently.

The mean values of run time reported in Table 3 indicate that the STW-PC requires much less run time than the transfer learning method in which the complexity of the



(a) FDA4 (b) FDA5

(c) FDA5$_{iso}$ (d) FDA5$_{dec}$

(e) DIMP2 (f) DMOP2

(g) DMOP2$_{iso}$ (h) DMOP2$_{dec}$

(i) DMOP3 (j) HE2

(k) HE7 (l) HE9

Fig. 3. Influence of different △ values on the STW-PC.

eigenvalue decomposition is high. Therefore, it is a promising method to solve DMOPs within a limit time.

Table 3. Mean run time (s) of STW-PC* and the Tr-RM-MEDA on FDA4.

FDA4	Tr-RM-MEDA	STW-PC*
C1	1.84E + 04	8.74E + 01
C2	1.93E + 04	8.65E + 01
C3	1.94E + 04	8.64E + 01

6 Conclusions and Future Work

A sliding time window based on parallel computing (STW-PC) is introduced in the current study. In the STW-PC, the STW is used to maintain the population diversity and to reuse past evolutionary information to improve the search efficiency when the time/environment changes. The results show that the STW-PC is an effective and promising approach to solve DMOPs. It is worth mentioning that the STW-PC can both significantly reduce the computational time and have better tracking performance when compared with the transfer learning method. Additionally, the STW-PC is a generic framework to solve different types of DMOPs, especially when "local" models in successive environments are relevant.

Acknowledgement. This work was partially supported by the Shanghai Science and Technology Innovation Action Plan (18550720100, 19040501600), the National Nature Science Foundation of China (No. 61603244).

References

1. Cruz, C., González, J., Pelta, D.: Optimization in dynamic environments: a survey on problems, methods and measures. Soft. Comput. **15**(7), 1427–1448 (2011)
2. Nguyen, S., Zhang, M., Johnston, M., et al.: Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Trans. Evol. Comput. **18**(2), 193–208 (2013)
3. Yan, X., Cai, B., Ning, B., et al.: Moving horizon optimization of dynamic trajectory planning for high-speed train operation. IEEE Trans. Intell. Transp. Syst. **17**(5), 1258–1270 (2015)
4. Yazici, A., Kirlik, G., Parlaktuna, O., et al.: A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints. IEEE Trans. Cybern. **44**, 305–314 (2013)
5. Karatas, M.: A dynamic multi-objective location-allocation model for search and rescue assets. Eur. J. Oper. Res. **288**(2), 620–633 (2021)
6. Fan, Q., Wang, W., Yan, X.: Multi-objective differential evolution with performance-metric-based self-adaptive mutation operator for chemical and biochemical dynamic optimization problems. Appl. Soft Comput. **59**, 33–44 (2017)

7. Jiang, M., Huang, Z., Qiu, L., et al.: Transfer learning-based dynamic multiobjective optimization algorithms. IEEE Trans. Evol. Comput. **22**(4), 501–514 (2018)
8. Tanabe, R., Ishibuchi, H.: A review of evolutionary multi-modal multi-objective optimization. IEEE Trans. Evol. Comput. **24**(1), 193–200 (2019)
9. Yan, W., Chang, J., Shao, H.: Least square SVM regression method based on sliding time window and its simulation. J. Shanghai Jiaotong Univ. **38**(4), 524–526 (2004)
10. Fan, Q., Yan, X., Zhang, Y., et al.: A variable search space strategy based on sequential trust region determination technique. IEEE Trans. Cybern. **PP**, 1–3 (2019)
11. Helbig, M., Engelbrecht, A.: Benchmark functions for CEC 2015 special session and competition on dynamic multi-objective optimization. Technical report (2015)
12. Zhang, Q., Yang, S., Jiang, S., et al.: Novel prediction strategies for dynamic multi-objective optimization. IEEE Trans. Evol. Comput. **24**(2), 260–274 (2019)
13. Muruganantham, A., Tan, K., Vadakkepat, P.: Evolutionary dynamic multiobjective optimization via Kalman filter prediction. IEEE Trans. Cybern. **46**(12), 2862–2873 (2015)
14. Akrida, E., Mertzios, G., Spirakis, P., et al.: Temporal vertex cover with a sliding time window. J. Comput. Syst. Sci. **107**, 108–123 (2020)
15. Ma, C., Li, W., Cao, J., et al.: Adaptive sliding window based activity recognition for assisted livings. Inf. Fusion **53**, 55–65 (2020)
16. Ferland, J., Fortin, L.: Vehicles scheduling with sliding time windows. Eur. J. Oper. Res. **38**(2), 213–226 (1989)
17. Kiviniemi, V., Vire, T., Remes, J., et al.: A sliding time-window ICA reveals spatial variability of the default mode network in time. Brain Connectivity **1**(4), 339–347 (2011)
18. Fagerholt, K.: Ship scheduling with soft time windows: an optimisation based approach. Eur. J. Oper. Res. **131**(3), 559–571 (2001)
19. Deb, K., Rao N., U., Karthik, S.: Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 803–817. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70928-2_60
20. Yang, S.: Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. Evol. Comput. **16**(3), 385–416 (2008)
21. Jiang, S., Kaiser, M., Wan, S., et al.: An empirical study of dynamic triobjective optimisation problems. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 1–8 (2018)
22. Branke, J., Kaußler, T., Smidt, C., et al.: A multi-population approach to dynamic optimization problems. In: Parmee, I.C. (ed.) Evolutionary Design and Manufacture, pp. 299–307: Springer, London (2000). https://doi.org/10.1007/978-1-4471-0519-0_24
23. Liu, R., Li, J., Mu, C., et al.: A coevolutionary technique based on multi-swarm particle swarm optimization for dynamic multi-objective optimization. Eur. J. Oper. Res. **261**(3), 1028–1051 (2017)
24. Branke, S.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 Conference on Evolutionary Computation, pp. 1875–1882 (1999)
25. Goh, C., Tan, K.: A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Trans. Evol. Comput. **13**(1), 103–127 (2008)
26. Stroud, P.: Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations. IEEE Trans. Evol. Comput. **5**(1), 66–77 (2001)
27. Rossi, C., Abderrahim, M., Díaz, J.: Tracking moving optima using Kalman-based predictions. Evol. Comput. **16**(1), 1–30 (2008)
28. Zhou, A., Jin, Y., Zhang, Q.: A population prediction strategy for evolutionary dynamic multiobjective optimization. IEEE Trans. Cybern. **44**(1), 40–53 (2013)

29. Ruan, G., Minku, L., Menzel, S., et al.: When and how to transfer knowledge in dynamic multi-objective optimization. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 2034–2041 (2019)
30. Wang, F., Li, Y., Liao, F., Yan, H.: An ensemble learning based prediction strategy for dynamic multi-objective optimization. Appl. Soft Comput. **96**, 106592 (2020)
31. Zhang, Q., Zhou, A., Jin, Y.: RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm. IEEE Trans. Evol. Comput. **12**(1), 41–63 (2008)