A Dynamic Multi-objective Particle Swarm Optimization Algorithm Based on Adversarial Decomposition and Neighborhood Evolution

Jinhua Zheng\textsuperscript{a,b}, Zeyu Zhang\textsuperscript{a,*}, Juan Zou\textsuperscript{a}, Shengxiang Yang\textsuperscript{c}, Junwei Ou\textsuperscript{a}, Yaru Hu\textsuperscript{a}

\textsuperscript{a}Key Laboratory of Intelligent Computing and Information Processing (Ministry of Education), Xiangtan University, Xiangtan, Hunan, 411105, China
\textsuperscript{b}Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang Normal University, Hengyang, 421002, China
\textsuperscript{c}School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U.K.

Abstract

Many multi-objective optimization problems in the real world are dynamic, with objectives that conflict and change over time. These problems put higher demands on the algorithm's convergence performance and the ability to respond to environmental changes. Confronting these two points, this paper proposes a dynamic multi-objective particle swarm optimization algorithm based on adversarial decomposition and neighborhood evolution (ADNEPSO). To overcome the instability of the traditional decomposition method for the changing Pareto optimal front (POF) shape, the proposed algorithm utilizes the complementary characteristics in the search area of the adversarial vector, and the two populations are alternately updated and co-evolved by adversarial search directions. Additionally, a novel particle update strategy is proposed to select promising neighborhood information to guide evolution and enhance diversity. To improve the ability to cope with environmental changes, an effective dynamic response mechanism is proposed, including three parts: archive set prediction, exploration of global optimal information, and retention of excellent particles.

\textsuperscript{*}Corresponding author

Email addresses: jzheng@xtu.edu.cn (Jinhua Zheng), 774807187@qq.com (Zeyu Zhang), zoujuan@xtu.edu.cn (Juan Zou), syang@dmu.ac.uk (Shengxiang Yang), junweiou@163.com (Junwei Ou), huyaru199@gmail.com (Yaru Hu)
to accelerate convergence to the Pareto optimal set (POS) in the new environment. The proposed algorithm is tested on a series of benchmark problems and compared to several state-of-the-art algorithms. The results show that AD-NEPSO performed excellently in both convergence and diversity, and is highly competitive in dealing with dynamic problems.

**Keywords:**
Dynamic multi-objective optimization, Particle swarm optimization, Adversarial decomposition

1. Introduction

In recent years, the field of multi-objective optimization has attracted more and more attention. Multi-objective optimization problem (MOPs) aims to optimize two or more conflicting targets simultaneously, and involve multiple fields such as industry [1], biological sciences [2], artificial intelligence [3], etc. Usually, multi-objective evolutionary algorithms (MOEAs) [4][5][6] and Multi-objective particle swarm optimizations (MOPSOs) [7][8][9] are considered to be alternatives for tackling MOPs, which attempt to search for a Pareto optimal set (POS) consisting of the best possible tradeoffs among the objectives. The mapping of the POS in the objective space is called the Pareto optimal front (POF).

Multi-objective optimization problems can be classified into two categories: static multi-objective optimization problems (SMOPs) and dynamic multi-objective optimization problems (DMOPs). Compared with the SMOPs, the DMOPs is more challenging because objective functions, constraints, and parameters may be time variant [10]. Moreover, dynamic multi-objective optimization algorithms (DMOAs) [11] also have been widely applied in many real-world engineering scenarios, such as scheduling [12][13], finance [14], control [15][16][17], planning [18][19], design [20] and machine learning [21]. In order to effectively solve a DMOP, two basic requirements should be considered: (1) When the environment changes, the algorithm should provide an effective dynamic response
mechanism to track new optima and adapt to the new environment. (2) Before
the change, the algorithm should ensure the ability to rapidly find the current
POS, which is also crucial to tracking performance. Based on these two points,
in this paper, we propose a dynamic multi-objective particle swarm optimization
algorithm based on adversarial decomposition and neighborhood evolution
(ADNEPSO).

Due to the superior performance of the multi-objective evolutionary algo-
rithm based on decomposition (MOEA/D) [22], in recent years, some studies
have focused on applying decomposition-based methods to DMOPs [23] [24].
However, a decomposition-based algorithm’s performance strongly depends on
the shapes of the POF [25], especially the orientation of the POF, which makes
the algorithm’s performance unstable in dynamic environments where the POF
shape changes over time. One solution might be to adaptively adjust the weight
vector in the evolutionary process [26] [27]. However, due to the high change fre-
cquency and change severity characteristics of the dynamic environment, there
are not enough generations to adjust between the two changes. At the same
time, different from the static environment, the POF shape may change after
the environmental change, and the vector direction must be re-adjusted, which
causes a lot of extra overhead. Bearing these reasons in mind, the proposed
ADNEPSO applies adversarial decomposition to particle swarm optimization
to enhance the algorithm’s stability. Adversarial decomposition [28], unlike
traditional single reference point-based decomposition, uses two adversarial ref-
ence points $z^{id}$ and $z^{nd}$ to generate two sets of uniform vectors, and evolves
through the complementary characteristics of the search directions. To take
advantage of these characteristics, two populations $N_{id}$ and $N_{nd}$ are associated
with reference points $z^{id}$ and $z^{nd}$ respectively, and co-evolve using alternate
updates.

On the other hand, particle swarm optimization has shown effectiveness in
solving DMOPs [29] [30], but determining how to avoid falling into the local op-
timal and how to enhance diversity are also challenges. Therefore, in order to
ensure diversity in the evolution process, we propose a novel neighborhood evo-
lution method. Every subproblem is assigned a global optimal particle $G_i$, which is defined as the best position for the subproblem $i$ in the neighborhood traversal. The neighborhood optimal information is used as the parent’s guide position and speed update. Moreover, a neighborhood selection strategy is used to find more promising parents. Finally, an effective dynamic response mechanism is proposed that combines historical information prediction and global optimal information exploitation to help quickly respond to environmental changes.

The rest of the paper is structured as follows. Section 2 introduces the background, including some basic definitions and related work. The proposed ADNEPSO is presented in detail in Section 3. Section 4 provides the test problems, metrics, and parameter settings. Section 5 shows the experimental statistical results and analysis. A further discussion of the algorithm is offered in Section 6. Finally, conclusions are drawn in Section 7.

2. Background

2.1. Dynamic Multi-objective Optimization

DMOPs can be described in terms of various nature of dynamisms [10][31][32]. We mainly consider that DMOPs [11][33] can be mathematically defined as follows:

$$\begin{align*}
\text{minimize } & F(x, t) = (f_1(x, t), f_2(x, t), \ldots, f_M(x, t))^T, \\
\text{s.t. } & x \in \Omega.
\end{align*}$$

(1)

where $M$ is the number of objectives; $t$ means time or environment variable. And $x = (x_1, x_2, \ldots, x_n)^T$ is the decision variable vector. Decision space $\Omega = [L_1, U_1] \times [L_2, U_2] \times \cdots \times [L_n, U_n]$, $L_i$ and $U_i$ are the lower and upper bounds of the $i$-th decision variable, respectively.

Definition 1. Pareto Dominance [34]: At time $t$, for any two individuals $x_1$ and $x_2$ in the population, at time $t$, if $x_1$ and $x_2$ satisfy the following conditions:
∀i ∈ {1, ..., m} : f_i(x_1, t) ≤ f_i(x_2, t),
(2)
∧ ∃j ∈ {1, ..., m} : f_j(x_1, t) < f_j(x_2, t).

Then say that $x_1$ dominates $x_2$, denoted as $x_1 \prec x_2$.

**Definition 2. Pareto Optimal Set** [1]: Given a MOP, at $t$ time, there is no individual $x' \in \mathbb{R}^n$ dominating individual $x$ in the objective space. Then $x$ is the optimal solution or the non-dominated solution. All the non-dominated solutions constitute the Pareto optimal set ($POS_t$), which is denoted as

$$POS_t = \{ x \in \mathbb{R}^n | \neg \exists x' \in \mathbb{R}^n, x' \prec x \}. \quad (3)$$

**Definition 3. Pareto Optimal Front**: At $t$ time, the mapping of $POS$ in the objective space is called Pareto optimal front $POF$, which is defined as follows:

$$POF_t = \{ f(x, t) | x \in POS_t \}. \quad (4)$$

**Definition 4. Ideal point** $z^{id}$: The ideal point $z^{id} = (z_1^{id}, z_2^{id}, ..., z_m^{id})$, where $z_i^{id}$ is the infimum of $f_i(x)$ for every $i \in \{1, 2, ..., m\}$.

**Definition 5. Nadir point** $z^{nd}$: The nadir point $z^{nd} = (z_1^{nd}, z_2^{nd}, ..., z_m^{nd})$, where $z_i^{nd}$ is the supremum of $f_i(x), x \in POS$ for every $i \in \{1, 2, ..., m\}$.

In addition, Farina et al. [13] and others classified DMOPs into four types based on different combinations of $POS_t$ and $POF_t$ dynamic changes: (1) Type I: $POS_t$ changes while $POF_t$ remains unchanged. (2) Type II: Both $POS_t$ and $POF_t$ changed. (3) Type III: $POS_t$ remains unchanged while $POF_t$ changes. (4) Type IV: Both $POS_t$ and $POF_t$ remain unchanged.

2.2. Related Works

In the past few years, dynamic multi-objective optimization (DMO) has attracted a lot of attention from researchers and many DMOAs have been proposed to solve DMOPs. According to the response mechanism and diversity
strategy adopted, the existing methods can be divided into the following categories:

(1) **Diversity introduction**: Diversity introduction aims to help the population that had previously converged in the previous environment to jump out of the current position and explore new regions. Cobb et al. [35] proposed a triggered hypermutation method, which increases the probability of mutation when the environment changes to encourage population divergence. Deb et al. adopted different diversity strategies in two versions of dynamic fast and elitist multiobjective genetic algorithm (DNSGA-II) [12]; one is to randomly initialize some individuals after the change occurs, and the other is to introduce diversity via Gaussian mutation. Vivek et al. [36] proposed an adaptive mutation operation called variable local search (VLS), which gradually increases the intensity of mutation without detecting changes, and the operator makes the diversity level of the introduced population better match with the degree of environmental change. In general, methods such as random initialization and hypermutation can effectively enhance population diversity, however, the information obtained prior to a change is lost after these operations [37].

(2) **Diversity maintenance**: The technique of maintaining diversity is not only applied when the environment is changed, but it also focuses on preserving the inherent diversity of the population throughout the whole optimization process. Random immigration [38] [39] [40] is considered to be a typical and effective method. The main idea is to inject new genetic materials into the evolutionary population at fixed intervals. Chang et al. [41] proposed a query-based machine learning method. When a particle is considered to be trapped in a local optimum, it actively adapts the ratio of quantum particles and neutral particles to achieve diversity without analyzing the distribution of optima in the solution space. However, when the optimization process has been misguided, the method cannot adjust the quantum radius adaptively. Jiang et al. proposed a steady-state and generational evolutionary algorithm (SGEA) [42], which employs the stable tracking ability of the steady-state algorithm to maintain diversity. In the method, half of the outdated solutions with good distribution are reused,
and the other half reinitialized solutions in the new population are produced by a guess of the new location of the changed POS. SGEA has the ability to effectively track changed POF, but may hard to recover the linked decision variables and significant diversity loss. In addition, there are other diversity maintenance techniques [43][44][45] that effectively utilize the inherent characteristics of algorithms.

(3) Prediction strategy: After the change occurs, the prediction-based strategy collects historical population information and estimates the position of the POS at the next moment through some prediction models. Hatzakis and Wallace [46] proposed a feed-forward prediction strategy (FPS). The re-initialized population includes three parts: prediction solution set, non-dominated solution set, and random initialization solution set. Zhou et al. [47] developed a population prediction strategy (PPS) based on a regularity model-based multiobjective estimation of distribution algorithm (RM-MEDA) [48] to solve DMOPs. In PPS, the population is divided into population centers and manifolds, and an autoregressive time series forecasting technique is applied to guide the estimation of new locations. Cao [23] proposed a novel differential prediction model in which the second-order difference model takes into account uniformly accelerated modes of centroid motion, and the main novelty is that the second-order difference model can collect more historical information to estimate the new locations in the new environment. Muruganantham et al. [49] combined the Kalman filter model with (multiobjective evolutionary algorithm with decomposition based on differential evolution) MOEA/D-DE and designed an effective dynamic multi-objective optimization algorithm (called evolutionary dynamic multiobjective optimization via kalman filter prediction) MOEA/D-KF. The prediction mechanism helps to guide the search toward the changed optima. When the environment changes, a scoring scheme is devised to hybridize the KF prediction with a random reinitialization method. Moreover, there are some other efficient dynamic prediction models, such as inverse modeling [50], transfer learning [51], mixture-of-experts prediction [52].

(4) Memory mechanisms: The memory mechanism usually stores previous
solutions in the memory pool. When the environment changes, especially when the environmental conditions are similar to the previous environment, the past solutions are reintroduced to improve the performance of the algorithm. Existing research shows that memory-based methods are often more effective for DMOPs where the environment changes periodically [51]. Branke [53] proposes a memory enhanced evolutionary algorithm, which stores the best individuals in the archive. When a change is detected, the stored individuals replace the same number of current solutions. Hu et al. [54] proposed a method based on environmental sensitivity called IEC. IEC establishes an information exchange mechanism in a dynamic environment through two parts: micro-changing decision and macro-changing decision. Besides, IEC adopts a memory strategy that associates weight vectors, the maximum 30% of population members will be replaced by historical solutions. Wang et al. [55] investigated different multi-objective dynamic optimization schemes, including restart strategies, explicit memory mechanisms, local search memory strategies, and mixed memory schemes. In Koo’s MO-EGS (a predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment) [56], a memory item consisting of centroid and centroid variance is added to the external memory pool. If the external memory pool is full, the oldest memory item is being replaced.

(5) Multipopulation approaches: As the name suggests, multiple subpop-ulation methods use multiple subpopulations to simultaneously explore different regions. Subpopulations share and cooperate with each other to improve convergence and diversity. Goh et al. proposed a competitive-cooperative coevolutionary paradigm called DCOEA [37]. Its main idea is to develop an iterative process of competition and cooperation, through different subpopulations to optimize each subcomponent, so that coevolutionary algorithms can deal with static and dynamic multi-objective problems. Helbig [13] developed a dynamic vector evaluated particle swarm optimization algorithm (DVEPSO). In the method, each objective is individually optimized by one swarm and the swarm share knowledge of each other about the objects they are solving. Liu [29] proposed a novel
coevolutionary technique based on multi-swarm particle swarm optimization, in which a new speed update equation and effective boundary constraint strategy are developed.

3. Proposed ADNEPSO

In this section, we introduce the details of the proposed algorithm ADNEPSO. First, the concept of adversarial decomposition is given. Second, the overall framework of the algorithm is introduced, and then show the details of the particle update and neighborhood Gbest update. Finally, the content of the dynamic response mechanism is given.

3.1. Adversarial Decomposition

Most decomposition-based algorithms utilize a single reference point to generate uniform vectors, which makes the algorithm behave differently when there are different POF shapes (especially when the shape is concave or convex). As shown in Fig.1, the solid line represents \(z^{id}\) vector, and the solution converged on the POF is marked with blue. The dotted line represents \(z^{nd}\) vector, and the corresponding solution is marked with green. It can be found that when the POF is concave, the \(z^{id}\) vector can obtain a relatively uniform solution set, but when the POF is convex, the obtained solution set is no longer uniform, while the solution set obtained by the \(z^{nd}\) vector shows the opposite. In other words, the \(z^{id}\) vector and \(z^{nd}\) vector can complement each other’s poorly distributed regions, enhancing the algorithm’s adaptability to POF shapes.

MOEA/D provides several reputable scalar functions, the weighted sum approach (WS), tchebycheff approach (TCH), and boundary intersection approach (PBI). Considering WS cannot effectively deal with non-concave problems and TCH’s search ability is stronger than PBI when there are fewer generations \[57\], a modified version of TCH \[28\] is used in this paper, which is expressed as follows:

\[
g(x|w, z^*) = \max_{1 \leq i \leq m} \left\{ \left( f_i(x) - z_i^* \right) / w_i \right\} + \alpha \sum_{i=1}^{m} \left( f_i(x) - z_i^* \right) / w_i, \quad z^* = z^{id} \text{ or } z^{nd}.
\]
In the modified version, the relationship between the weight vector $w$ and the convergence direction is adjusted through division, and the new addition term $\alpha \sum_{i=1}^{m} \left( f_i(x) - z_i^* \right) / w_i$ can reduce the weak domination solution to some extent, where $\alpha$ is an augmentation and set to $10^{-6}$ by [58] recommendation.

### 3.2. Overall Framework

The basic framework of the proposed algorithm is shown in Algorithm 1. AD-NEPSO uses two populations $N_{id} = (x_{id1}, x_{id2}, \ldots, x_{idN})$ and $N_{nd} = (x_{nd1}, x_{nd2}, \ldots, x_{ndN})$ to maintain evolution. $N_{id}$ and $N_{nd}$ are respectively associated with two reference points $z_{id}$ and $z_{nd}$. The corresponding globally best position vectors are denoted as $Gb_{id} = (G_{id1}, G_{id2}, \ldots, G_{idN})$ and $Gb_{nd} = (G_{nd1}, G_{nd2}, \ldots, G_{ndN})$. Where $G_i$ is defined as the best position among those that particles in the neighborhood of a particle have traversed. In the initial generation, $N_{id}$ starts to evolve first; the offspring are generated through the particle update (line 15 of Algorithm 1), and the neighborhood is updated through the neighborhood Gbest update (line 17 of Algorithm 1). To effectively utilize the similarity of the control region of the adversarial vector, during the neighborhood update.

---

**Figure 1**: Adversarial decomposition adapts to different POF shapes
Algorithm 1 The overall framework of ADNEPSO

**Input:** $N$(population size), $\text{gen}^{\text{max}}$(maximum number of generations), $T$(neighborhood size), set time period $t = 0$, set $\text{gen}=0$.

**Output:** Population $P_t$;

1. Random initialization population $P_0$, then $N_{id} \leftarrow P_0$, $N_{nd} \leftarrow P_0$;
2. Initialize globally best position vector $G_{bid} \leftarrow N_{id}$, $G_{bnd} \leftarrow N_{nd}$;
3. Initialize $z^{id}$ and $z^{nd}$, weight vector $W = \{w^1, w^2, \ldots, w^n\}$ and the neighborhood $B\{w^i\}$ of each vector, set $\text{flag}_{id}(i) = 0 (i \in N_{id})$, $\text{flag}_{nd}(i) = 0 (i \in N_{nd})$(record whether the subproblem was updated in the last generation);
4. while $\text{gen} < \text{gen}^{\text{max}}$ do
5. if there is an environmental change then
6. dynamic response($G_{bid}, G_{bnd}, \text{archive } A^{t-1}$);
7. $t = t+1$;
8. end if
9. if $\text{gen}\%2 == 0$ then
10. $N \leftarrow N_{id}, Gb \leftarrow G_{bid}, Gb' \leftarrow G_{bnd}, \text{flag} \leftarrow \text{flag}_{id}, \text{flag}' \leftarrow \text{flag}_{nd}$;
11. else
12. $N \leftarrow N_{nd}, Gb \leftarrow G_{bnd}, Gb' \leftarrow G_{bid}, \text{flag} \leftarrow \text{flag}_{nd}, \text{flag}' \leftarrow \text{flag}_{id}$;
13. end if
14. for $i= 1$ to $N$ do
15. particle update($x_i(x_i \in N), Gb, \text{flag}$); % Generate a new particles $\hat{x}_i$
16. update $z^{id}$ and $z^{nd}$;
17. neighborhood Gbest update($\hat{x}_i, Gb, Gb', \text{flag}, \text{flag}'$);
18. $x_i \leftarrow \hat{x}_i$;
19. end for
20. $\text{gen}=\text{gen}+1$;
21. end while
22. $P_0 \leftarrow \text{Combine } N_{id} \text{ and } N_{nd} \text{ and truncate the population size to } N$

process, part of the $G_i^{nd}$ in the adversarial population $N_{nd}$ is updated. Then population $N_{nd}$ begins to evolve in the second generation; these updated $G_i^{nd}$
guide the generation of new particles, and the offspring also update the $G_{id}$ of the adversarial population $N_{id}$. Fig 2 describes the meaning of alternating updates, ADNEPSO evolves population $N_{id}$ in odd generations and evolves $N_{nd}$ population in even generations, and $G_{bd}$ and $G_{bnd}$ are used to complete the information interaction and coevolution. Furthermore, before each generation cycle, if an environmental change is detected, an additional dynamic response (line 6 of Algorithm 1) mechanism is implemented. In the following subsections, more details about each component of ADNEPSO are given.

![Figure 2: Alternating update](image)

### 3.3. Particle Update

The purpose of the particle update is to produce a promising offspring. In the traditional particle swarm search method, the speed of the particle determines the update direction. Assuming $x_i$ is the i-th particle in the population $N$, $x_i$ is updated by using a velocity $v_i$. The new position $\hat{x}_i$ of the particle can be calculated as follows:

$$
\hat{v}_i = \omega v_i + c_1 r_1 (P\text{best}_i - x_i) + c_2 r_2 (G\text{best}_i - x_i),
$$

$$
\hat{x}_i = x_i + v_i.
$$

(6)

where $P\text{best}_i$ and $G\text{best}_i$ are respectively the position information of personal best and global best particles for $x_i$. $\omega$ is inertia weight to balance the global
Algorithm 2 particle update

Input: $x_i (x_i \in \text{population}_N), Gb = (G_1, G_2, \ldots, G_N), \text{flag}$;
Output: offspring particle $\hat{x}_i$;

1: Generate a random number $p = \text{rand}(0, 1)$, Calculate the number of vectors $T_s$ with $\text{Rank}(n) = 1$ in the neighborhood, and calculate $p_s$ by the Eq.8;
2: if $(p \leq p_s) \land (T_s \neq 0)$ then
3: Randomly choose a $G_n$ from $B\{w_i\}$ with $\text{flag}(n) = 1$ ;
4: else
5: Randomly choose a $G_n$ from $B\{w_i\}$ with $\text{flag}(n) = 0$ ;
6: end if
7: Calculate $\hat{x}_i$ by Eq.7;
8: if $G_i$ has not been updated for more than two generations then
9: Perform an additional mutation operation on by $\hat{x}_i$ formula;
10: end if
11: Output $\hat{x}_i$;

search and the local search; $c_1, c_2$ are two learning factors; $r_1$ and $r_2$ are two randomly generated numbers within the range of $[0, 1]$. Regarding the parameter selection and stochastic stability of PSO, the works proposed by Erskine et al. [59] is worthy of reference. A Considering the guidance of neighborhood information on population evolution and the maintenance of diversity, in this paper, Eq.6 is modified as follows:

$$
\hat{v}_i = \omega \cdot v_i(t) + c_1 r_1 \left( G_i - x_i \right) + c_2 r_2 \left( G_n \in B\{w_i\} - x_i \right),
\hat{x}_i = x_i + v_i.
$$

(7)

where $G_i$ is the global best particle of the i-th subproblem; $B\{w^i\}$ represents the neighborhood of the i-th subproblem and $G_n$ is a global best particle selected from the neighborhood $B\{w^i\}$. It should be noted that the step size $(G_i - x_i)$ ensures that the particle $x_i$ can search for the optimal direction of the current subproblem, and $(G_n - x_i)$ provides a search direction from the $x_i$ to the neighborhood optimal particle. Because of the proximity of reference vectors,
particles in the neighborhood have similar evolutionary trends. Therefore, the information provided by a good $G_n$ can often improve the convergence efficiency of the entire neighborhood.

On the other hand, how to choose an appropriate $G_n$ also deserves attention. In general, during the iterative process of the algorithm, the performance and value provided by each subproblem are different \[60\][61]. For a subproblem $i$, if the $G_i$ was updated in the last evaluation, then we consider that the subproblem $i$ has obtained some new convergence and distribution information, which is likely to have a guiding effect on other particles in the neighborhood with similar evolutionary trends. In addition, this selection method promotes the diversity of the parents because the particles that were the parent in the previous generation will not have a great probability to still be selected as the parent in the current generation. We use a selection probability $p_s$ to evaluate what kind of neighborhood information should be selected. $p_s$ can be calculated as follows

\[
\begin{cases}
    p_s = m + (1 - m) \left( \frac{T_s}{T} \right)^{\frac{1}{2}} & T_s \neq 0, \\
    p_s = 0 & T_s = 0.
\end{cases}
\]  

(8)

where $T$ is the neighborhood size and $T_s$ is the number of subproblems in the neighborhood that were updated in the last generation. $m \in [0, 1]$ is a predefined parameter for adjusting the weight of neighborhood information, and in this paper, $m$ is set to 0.1. In the early stage of evolution, there are numerous updated subproblems in the neighborhood, and the algorithm has a greater probability of selecting the parent from these updated subproblems. When the population tends to converge, the update frequency of the subproblems in the neighborhood decreases. At this time, a smaller $p_s$ can avoid the loss of diversity caused by the heavy use of some subproblems. The pseudo-code of the particle update procedure is presented in Algorithm 2. The selection probability $p_s$ is calculated at first; the $Rank$ value is used to record the update of the particles. If the $G_i$ was updated during the last evaluation, then the $Rank(i)$ is set to 1. Furthermore, if $G_i$ remains unchanged for more than two generations, then an additional mutation operation \[57\] is used to help the particles effectively jump
out of the local optimum.

**Algorithm 3 neighborhood Gbest update**

**Input:** offspring particle \( \hat{x}_i, Gb = (G_1, G_2, \ldots, G_N), Gb' = (G'_1, G'_2, \ldots, G'_N), \) flag, flag';

1. **for** j = 1 to T **do**
2.   **if** \( g(\hat{x}_i|w, z) < g(G_j|w, z) \) **then**
3.     \( G_j \leftarrow \hat{x}_i; \)
4.     Tempflag(j) = 1;
5.   **end if**
6. **end for**
7. Find the subproblem with the smallest Euclidean distance in the adversarial vector group, with the index labeled S;
8. **for** k = 1 to T, \( k \in B\{w^S\} \) **do**
9.   **if** \( g(\hat{x}_i|w, z') < g(G'_k|w, z') \) **then**
10.     \( G_k \leftarrow \hat{x}_i; \)
11.     flag'(k) = 1;
12.   **end if**
13. **end for**
14. Reset flag = 0, flag \( \leftarrow \) Tempflag;
15. Reset Tempflag = 0;

3.4. Neighborhood Gbest Update

In ADNEPSO, the neighborhood Gbest update is applied to both the current population and adversarial populations, which is detailed in Algorithm 3. First, the offspring \( \hat{x}_i \) updates the neighborhood \( B\{w^S\} \) according to the aggregate value \( g(x|w, z^*) \), which is calculated by the previously mentioned equation Eq 5. To take advantage of the complementary and coevolution of the two populations, after updating the current neighborhood, we calculate the Euclidean distance between \( \hat{x}_i \) and the adversarial vector and find the closest adversary subproblem S, then \( B\{w^S\} \) is updated. It is worth noting that the Rank values of these
updated adversary subproblems are also recorded and guide evolution in the next generation. During the entire replacement process, only the neighborhood optimal particle $G_j$ is replaced, but the corresponding population particle $x_j$ is not changed. This is because the functions of $Gb$ and population $N$ are not the same. $Gb$ keeps the optimal solution and provides an effective search direction, while the population keeps diversity particles, which is very important for the random search ability.

3.5. Dynamic Response

**Algorithm 4** dynamic response

**Input:** $Gb_{id}, Gb_{nd}, \text{archive } A_{t-1}$;  
**Output:** $N_{id}, N_{nd}, A_t$; 
1: $Gb_{DS} \leftarrow (Gb_{id} \lor Gb_{nd})$;  
2: truncate $Gb_{DS}$ into the archive $A_t$ of size $N$;  
3: $A_t \rightarrow P_{\text{predict}}$;  
4: $(Gb_{DS} \setminus A_t) \land Gb_{id} \rightarrow P_{\text{retain}}$;  
5: $(Gb_{DS} \setminus A_t) \land Gb_{nd} \rightarrow P_{\text{exploit}}$;  
6: $N_{id}, N_{nd} \leftarrow P_{\text{predict}} \lor P_{\text{retain}} \lor P_{\text{exploit}}$;

An ideal dynamic algorithm should not only have efficient convergence performance but also have an effective environment response mechanism. When an environmental change is detected, the response mechanism should be able to maintain a good level of population diversity and relocate the population in promising areas close to the new POS [49]. Prediction-based methods often perform well when the dynamic environment exhibits a predictable pattern, but this method may fail or even mislead the direction of population evolution when the environment changes drastically. Reusing the old solution in the new environment can also effectively use historical information, whereas excessive reuse of the old solution causes the loss of population diversity so that the population falls into a local optimum and cannot find the POS of the new environment. Consequently, determining how to balance convergence and diversity is a sig-
significant part of designing an effective dynamic response mechanism. For these reasons, this paper adopts a novel dynamic handling mechanism, which includes three operations: predictive archive, exploit Gbest information, and retain particles. The execution process is described in Algorithm 4. In the remainder of this section, we discuss more related details.

Assuming the environment changes at time $t$, first merge $Gb_{id}, Gb_{nd}$ into $Gb_{DS}$ and truncate to archive $A_t$. The truncation operator is similar to that used in SPEA2 [62]. We describe the predictive archive, which uses an archive centroid representative of the entire archive to track and predict the movement of the POS [56]. Let $C^t$ represent the centroid of the archive $A_t$, it can be defined as:

$$C^t = \frac{1}{|A_t|} \sum_{x \in A_t} x.$$  \hspace{1cm} (9)

Then the estimated moving direction of the archive centroid can be expressed as follows:

$$D^t = C^t - C^{t-1}.$$  \hspace{1cm} (10)

Based on the moving direction, for each member $x^t \in A_t$, the new location in the decision space can be predicted as follows:

$$x^{t+1} = x^t + D^t.$$  \hspace{1cm} (11)

Fig. 3 describes this process. The $N$ solutions in the archive set are predicted to be in a new and promising location in the decision space. In order to further enhance the diversity, the remaining $Gb_{DS} - N$ solutions use different strategies according to their own population. The particles belonging to the $Gb_{id}$ are retained, and these retained particles can often help evolve the population with less intensity of environmental changes. Particles belonging to $Gb_{nd}$ are used to exploit new Gbest information to introduce diversity. Suppose that $G^t_i$ is the globally best particle of the $i$th subproblem at time $t$. We can define an exploit step-size as:

$$E^t_i = G^t_i - G^{t-1}_i.$$  \hspace{1cm} (12)
Then there is a particle $G^t_k$ in the $B\{u^t\}$ having the nearest distance to $G^t_i$, which can be found using the following equation:

$$G^t_k = \arg \min_{k \in B(u^t)} \|G^t_k - G^t_i\|_2.$$  \hspace{1cm} (13)

The search step size can be defined as:

$$S^t_k = G^t_k - G^t_i.$$ \hspace{1cm} (14)

Having obtained the exploit step-size and search step-size, the new position can be calculated as follows:

$$G^{t+1}_i = G^t_i + E^t_i + \text{rand}(0, 1)\|S^t_k\|\vec{e}.$$ \hspace{1cm} (15)
where $\mathbf{e}$ is a unit vector with a random direction. Fig.4 gives the explanation of how to exploit individuals. It can be observed that the exploit operation is searching for a certain sub-problem instead of the whole population. Compared with the overall translation of the POS manifold, the exploit operation can provide diversity more effectively. Randomly reinitializing the population is also beneficial to population diversity, but this method completely gives up historical information and appears to be a bit blind. Thus, we adopt a more moderate method to increase the chance of exploration and minimize the amount of time consumed for algorithms to converge as much as possible.
In addition, it is worth noting that due to the spatial uniformity of the truncation operator, the three parts can maintain a good distribution in the objective space. That is to say, for each neighborhood in the new population, particles from different operations are included, which is very helpful for the neighborhood update of particles in the new environment.

3.6. Computational Complexity Analysis

This section discusses the complexity of ADNEPSO in one generation. The main computing resources of ADNEPSO are consumed by the following aspects:

1) Suppose $M$ is the number of objectives and $N$ is the population size, then the estimation of nadir point requires to calculate the current non-dominated solution, which spends $O(MN^2)$ computations. (2) The update procedure of a particle (line 14 of Algorithm 1) requires $O(M)$ computations, so the entire population update takes $O(MN)$. (3) In the process of neighborhood Gbest update (line 16 of Algorithm 1), each set of vectors costs $O(MNT)$ computing power separately, and the complexity of finding the nearest adversarial subproblem for the offspring particle is $O(MN^2)$, and the entire neighborhood Gbest update procedure requests $O(MN(N+2T))$. Therefore, the overall computational complexity of ADNEPSO in each generation is $O(MN(N+2T))$.

4. Experimental setting

This section is devoted to the experimental design used to investigate the performance of ADNEPSO, including the benchmark problems, performance metrics, compared algorithms, and their parameter settings.

4.1. Benchmark Problems

Benchmark problems play an important role in evaluating and analyzing the performance of algorithms. We used 16 test problems: FDA1-5 [10], dMOP test set [37], and JY1-9 [33]. The first two test problem sets are the most commonly used, and their POS is linear. The JY problem contains some new characteristics, such as the nonlinear correlation of decision variables, discontinuous POF,
non-monotonic and time-varying variable links, and mixed change types. Some test instances, such as dMOP1-3, JY5, and JY7, have the characteristics of obvious concave and convex changes to the POF shape, so as to better investigate the performance of ADNEPSO. The time instance $t$ involved in these problems is determined by the equation:

$$t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_t} \rfloor,$$

where $\tau$ is the current generation count; $\tau_t$ is the frequency of environmental change, and $n_t$ is the severity of environmental change.

### 4.2. Performance Metrics

It is very important to use the correct tool to measure the performance of the algorithm. Therefore, to effectively evaluate the performance of convergence and diversity, the performance metrics that we adopt in our experimental studies are introduced in this section.

1) Inverted Generational Distance (IGD): IGD evaluates convergence and diversity by measuring the proximity of real POF and obtained POF, which can be defined as follows:

$$IGD_t(POF^*_t, POF_t) = \frac{\sum_{v \in POF^*_t} d(v, POF_t)}{|POF^*_t|}.$$

where $POF^*_t$ and $POF_t$ are the real POF at time $t$ and the POF obtained by the algorithm respectively, and $d(v, POF_t)$ is minimum Euclidean distance between $v$ and the point in $POF_t$. The smaller the IGD value, the better the convergence and the more uniform the distribution.

2) Hypervolume hiDifference (HVD): The HVD calculates the difference in hypervolume between $POF^*_t$ and $POF_t$.

$$HVD(POF^*_t, POF_t) = HV(POF^*_t) - HV(POF_t),$$

where $HV(S)$ is the hypervolume of a set $S$. The reference point for the computation of hypervolume is $(z^1_1 + 0.5, z^1_2 + 0.5, ..., z^M_1 + 0.5)$, where $z^j_1$ is the maximum
value of the $j$th objective of the $POF^*_t$, and $M$ is the number of objectives. A smaller HVD indicates the better overall performance of the algorithm.

3) Spacing (SP) [67]: The SP measures the uniformity of individuals in the obtained POF, and is computed as:

$$SP = \sqrt{\frac{1}{|POF_t| - 1} \sum_{i=1}^{|POF_t|} (d_i - \bar{d})^2}$$

where $d_i$ is the distance between the $i$-th individual in the POF and the member with the smallest distance, and $\bar{d}$ is the average of $d_i$.

4.3. Comparison Algorithms and Parameter Settings

Four state-of-the-art dynamic multi-objective optimization algorithms, FPS, PPS, MOEA/D, and MOEA/D-KF, are selected for the sake of comparison with ADNEPSO. A brief description of each algorithm in the comparison and related parameter settings follows.

(1) FPS [46]: FPS designed a feed-forward prediction strategy that responds to environmental changes. The forecasting model is created using the sequence of prior optimum locations, and the new population includes three parts: the solution predicted by the historical information through the Auto Regression (AR) model, partial non-dominated solutions in the current environment and some new randomly generated solutions.

(2) PPS: Zhou et al. [47] proposed a dynamic multi-objective evolutionary algorithm based on population prediction, which is called the population prediction strategy (PPS). The population is divided into population center and manifolds. The population center of the new moment and the prediction from the center of several historical moments as well as previous manifolds are used to predict the next manifold. Once the new center has been predicted, the manifold is directly shifted to the new PS model. This method has proven to have good convergence performance and competitiveness for dynamic optimization.

(3) MOEA/D [22]: MOEA/D is representative of a decomposition-based approach. The basic idea of MOEA/D is to decompose a MOP into a number of
single-objective optimization subproblems through aggregation functions and optimizes them simultaneously. Each solution is associated with a subproblem, and each subproblem is optimized by using information from its neighborhoods. It has been proven that the decomposition method brings excellent population convergence and diversity.

(4) MOEA/D-KF: Muruganantham et al. [49] introduced a prediction mechanism based on Kalman filtering in MOEA/D-DE and proposed a novel dynamic multi-objective optimization algorithm MOEA/D-KF. When the environment changes, the algorithm uses a scoring scheme to determine whether to use the KF-based prediction mechanism or a random initialization method.

(5) IEC: IEC is proposed in recent by Hu at al [54], which adopted a method based on environmental sensitivity changes and MOEA/D-DE to effectively track the moving POF or POS in dynamic optimization. In IEC, an effective U-test mechanism is used to determine whether to make a micro-changing decision or a macro-changing decision. Moreover, a new memory strategy is proposed to select population members replaced by historical information according to the position of the weight vector.

The relevant experimental parameters were set as follows. For all algorithms, the population size $N$ was set to 100, and the number of decision variables was set as 10. The environmental change was set to 50 times in each run and recorded the experimental results are based on 20 independent runs. The crossover probability was $p_c=1.0$ and its distribution index was $\eta=20$. The mutation probability was $p_m=1/n$ and its distribution $\eta=20$. For MOEA/D, a random initialization response mechanism is added to the static version. In ADNEPSO, the inertia weight $\omega$ was set to commonly used 0.4, the learning factor $c_1 = c_2 = 1$, and the neighborhood size $T = 20$. For change detection, 5% of the population was randomly selected and re-evaluate to detect environmental changes in each generation. In addition, to ensure that the number of populations was 100 in a three-objective problem, 1000 weight vectors were generated by simplex lattice design [68] and reduced to 100 using SPEA2 [62]. The Wilcoxon rank-sum test [69] was used to indicate significant differences at the 0.05 significance level.
5. Experimental Results and Analysis

Comparison results of ADNEPSO with five other popular algorithms in terms of IGD, HVD, and SP values are presented in tables 1, 2, and 3, respectively. In order to better observe the effect of change frequency on the algorithm, the changing severity $n_t$ is fixed to 10, and the change frequency $\tau_t$ is set to 20, 25, and 30, respectively. The (mean) algorithms with the best performance are highlighted in dark color and the second-best performance in light color. Furthermore, (‡) and † represent significant differences were compared by their means, ADNEPSO was better than or equivalent to the other algorithms.

5.1. Results on FDA and dMOP Problems

From Table 1, it can be seen that ADNEPSO obtained the smallest IGD value on all FDA and dMOP testing instances, and IEC obtained the all second-best value except FDA2 and FDA3. This may mean that ADNEPSO and IEC may be more advantageous when dealing with problems where the decision variables have a linear relationship. On the other hand, as $\tau_t$ increases, the performance of all algorithms improved significantly. The reason may be that smaller $\tau_t$ makes the algorithm not have sufficient generations to drive the population to find the approximate POS before the environment changes. For FDA3 and dMOP3, the distribution of the solution in the POF changes with time, and ADNEPSO shows strong competitiveness. In addition, since some of the excellent particles from the last generation are retained as new populations, for problems where POS remains unchanged, such as dMOP1, ADNEPSO often performed better.
<table>
<thead>
<tr>
<th>prob</th>
<th>(n, m)</th>
<th>EPS</th>
<th>FPO</th>
<th>MOEA/D</th>
<th>MOEA/D-KF</th>
<th>BMW</th>
<th>ADNEPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDA1</td>
<td>(20, 10)</td>
<td>1.500e-3 (2.835e-4)</td>
<td>1.500e-3 (2.835e-4)</td>
<td>1.500e-3 (2.835e-4)</td>
<td>1.500e-3 (2.835e-4)</td>
<td>1.500e-3 (2.835e-4)</td>
<td>1.500e-3 (2.835e-4)</td>
</tr>
<tr>
<td>FDA2</td>
<td>(20, 10)</td>
<td>1.400e-3 (2.110e-4)</td>
<td>1.400e-3 (2.110e-4)</td>
<td>1.400e-3 (2.110e-4)</td>
<td>1.400e-3 (2.110e-4)</td>
<td>1.400e-3 (2.110e-4)</td>
<td>1.400e-3 (2.110e-4)</td>
</tr>
<tr>
<td>FDA3</td>
<td>(20, 10)</td>
<td>1.500e-3 (3.352e-4)</td>
<td>1.500e-3 (3.352e-4)</td>
<td>1.500e-3 (3.352e-4)</td>
<td>1.500e-3 (3.352e-4)</td>
<td>1.500e-3 (3.352e-4)</td>
<td>1.500e-3 (3.352e-4)</td>
</tr>
</tbody>
</table>

The average HVD value, standard deviation, and statistical test results of the five algorithms are shown in Table 1. It can be observed that the HVD value is similar to IGD in most test problems. The only difference is that FPO gets the best HVD value instead of ADNEPSO for FDA3, which may be because compared with the ordinary HVD, the HVD indicator prefers boundary solutions in evaluating the algorithm's performance. In other words, FPO performs significantly better than ADNEPSO.
may perform better than in retaining boundary solutions. For IGD or HVD, the variance of ADNEPSO in most problems is smaller than that of other algorithms, which shows that ADNEPSO has good stability and robustness, while MOEA/D is not ideal and fluctuates greatly, which may be because MOEA/D has no good response mechanism to environmental change.

The SP values obtained by the six algorithms are presented in Table 3 and the experimental results show that ADNEPSO outperformed other algorithms on most FDA and dMOP testing problems, which illustrates that ADNEPSO has better solution set uniformity. FPS achieved the best distribution in FDA2 and FDA3, specifically, among the four decomposition-based algorithms, the distribution of ADNEPSO is much better than the MOEA/D, MOEA/D-KF and IEC, especially when the POF shape has a concave-convex variation characteristic. FDA1 belongs to the first type I, whose POF shape remains a constant convex shape, while dMOP1 and dMOP2 belong to type II, whose POF shape changes back and forth between concave and convex. The adversarial decomposition applied in ADNEPSO shows its superiority when dealing with these problems. Apart from that, ADNEPSO clearly performed better than other algorithms on the FDA4, indicating that ADNEPSO can more effectively handle three objective DMOPs. Comparing the above three tables, we can draw the following conclusion: The reason that IEC is slightly inferior to ADNEPSO in comprehensive performance may be its weaker handling of distribution, and ADNEPSO has strong competitiveness in dealing with the dynamic problems of linear correlation of variables.

5.2. Results on JY Problems

Compared with the FDA and dMOP test suites, the JY test suites add some complex characteristics to make the problem more challenging, including mixed POF shape (convexity-concavity), non-monotonic and time-varying variable links, mixed change types and randomness of type changes. It can be seen intuitively from the experimental results that ADNEPSO achieved the best performance on the majority of JY test suites in terms of IGD, HVD, and SP.
<table>
<thead>
<tr>
<th>prob</th>
<th>r, n</th>
<th>FPS</th>
<th>MOEA/D</th>
<th>MOEA/D-KF</th>
<th>IEC</th>
<th>ADNEPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDA1</td>
<td>(10, 10)</td>
<td>1.315e-2 (6.813e-4)</td>
<td>7.066e-2 (2.835e-3)</td>
<td>1.135e-2 (4.813e-4)</td>
<td>6.94e-3 (1.99e-3)</td>
<td>5.931e-3 (1.99e-3)</td>
</tr>
<tr>
<td>PDA2</td>
<td>(10, 10)</td>
<td>4.319e-1 (7.626e-2)</td>
<td>2.685e-1 (1.238e-2)</td>
<td>1.910e+0 (1.873e-1)</td>
<td>5.895e-3 (1.99e-3)</td>
<td>7.380e-3 (1.99e-3)</td>
</tr>
<tr>
<td>PDA3</td>
<td>(10, 10)</td>
<td>3.750e-1 (2.136e-1)</td>
<td>4.824e-1 (2.998e-2)</td>
<td>4.975e-1 (2.136e-2)</td>
<td>3.265e-3 (1.99e-3)</td>
<td>3.105e-3 (1.99e-3)</td>
</tr>
<tr>
<td>PDA4</td>
<td>(10, 10)</td>
<td>5.789e-1 (7.626e-2)</td>
<td>5.400e-1 (3.186e-2)</td>
<td>9.365e-1 (3.186e-2)</td>
<td>7.406e-3 (1.99e-3)</td>
<td>7.524e-3 (1.99e-3)</td>
</tr>
<tr>
<td>PDA1</td>
<td>(25, 10)</td>
<td>5.110e-2 (4.126e-2)</td>
<td>5.303e-1 (4.126e-2)</td>
<td>7.209e-1 (4.126e-2)</td>
<td>6.206e-3 (1.99e-3)</td>
<td>5.560e-3 (1.99e-3)</td>
</tr>
<tr>
<td>PDA4</td>
<td>(25, 10)</td>
<td>4.206e-1 (8.942e-2)</td>
<td>6.189e-1 (8.942e-2)</td>
<td>1.960e-1 (8.942e-2)</td>
<td>1.633e-3 (1.99e-3)</td>
<td>1.473e-3 (1.99e-3)</td>
</tr>
<tr>
<td>dMOP1</td>
<td>(10, 10)</td>
<td>4.904e-2 (6.813e-4)</td>
<td>1.475e-1 (6.813e-4)</td>
<td>1.750e-1 (6.813e-4)</td>
<td>1.316e-3 (1.99e-3)</td>
<td>1.146e-3 (1.99e-3)</td>
</tr>
<tr>
<td>dMOP2</td>
<td>(10, 10)</td>
<td>3.206e-2 (6.813e-4)</td>
<td>1.120e-1 (6.813e-4)</td>
<td>1.397e-1 (6.813e-4)</td>
<td>1.396e-3 (1.99e-3)</td>
<td>1.213e-3 (1.99e-3)</td>
</tr>
<tr>
<td>dMOP3</td>
<td>(10, 10)</td>
<td>2.557e-1 (4.126e-2)</td>
<td>9.338e-1 (4.126e-2)</td>
<td>1.875e-1 (4.126e-2)</td>
<td>1.452e-3 (1.99e-3)</td>
<td>1.338e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY1</td>
<td>(20, 10)</td>
<td>8.109e-2 (3.352e-3)</td>
<td>4.026e-1 (3.352e-3)</td>
<td>1.620e-1 (3.352e-3)</td>
<td>1.440e-3 (1.99e-3)</td>
<td>1.280e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY2</td>
<td>(20, 10)</td>
<td>5.030e-2 (2.200e-3)</td>
<td>8.032e-1 (2.200e-3)</td>
<td>1.957e-1 (2.200e-3)</td>
<td>8.696e-3 (1.99e-3)</td>
<td>7.880e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY3</td>
<td>(20, 10)</td>
<td>1.312e-1 (4.126e-2)</td>
<td>2.496e-1 (4.126e-2)</td>
<td>7.904e-1 (4.126e-2)</td>
<td>7.050e-3 (1.99e-3)</td>
<td>6.470e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY4</td>
<td>(20, 10)</td>
<td>1.256e-1 (5.416e-3)</td>
<td>2.432e-1 (5.416e-3)</td>
<td>1.060e-1 (5.416e-3)</td>
<td>9.050e-3 (1.99e-3)</td>
<td>8.960e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY5</td>
<td>(20, 10)</td>
<td>5.288e-2 (3.065e-3)</td>
<td>5.975e-1 (3.065e-3)</td>
<td>5.975e-1 (3.065e-3)</td>
<td>6.206e-3 (1.99e-3)</td>
<td>5.560e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY6</td>
<td>(20, 10)</td>
<td>8.022e-2 (3.065e-3)</td>
<td>8.022e-1 (3.065e-3)</td>
<td>8.022e-1 (3.065e-3)</td>
<td>6.206e-3 (1.99e-3)</td>
<td>5.560e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY7</td>
<td>(20, 10)</td>
<td>1.050e-2 (2.303e-3)</td>
<td>1.050e-1 (2.303e-3)</td>
<td>1.050e-1 (2.303e-3)</td>
<td>6.206e-3 (1.99e-3)</td>
<td>5.560e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY8</td>
<td>(20, 10)</td>
<td>6.045e-1 (1.692e-1)</td>
<td>6.045e-1 (1.692e-1)</td>
<td>6.045e-1 (1.692e-1)</td>
<td>6.206e-3 (1.99e-3)</td>
<td>5.560e-3 (1.99e-3)</td>
</tr>
<tr>
<td>JY9</td>
<td>(20, 10)</td>
<td>1.205e+1 (1.238e+1)</td>
<td>1.205e+1 (1.238e+1)</td>
<td>1.205e+1 (1.238e+1)</td>
<td>6.206e-3 (1.99e-3)</td>
<td>5.560e-3 (1.99e-3)</td>
</tr>
</tbody>
</table>

† and ‡ indicate ADNEPSO performed significantly better than and equivalently to the corresponding algorithm, respectively.

values. JY1 is a typical type I problem; only the POS changes with time. It can test the convergence speed of the algorithm. ADNEPSO showed a strong convergence performance and won on all three indicators. JY2 is a simple type II problem, and ADNEPSO was slightly worse than MOEA/D-KF in IGD value and slightly better than MOEA/D-KF in HVD value. It can be seen that the two perform better , while FPS, FPS, and MOEA/D performed poorly. JY3
introduces time-varying non-monotonic dependencies between any two decision variables instead of linear dependencies, meaning that each variable has a different amount of change \[33\]. The results show that ADNEPSO is also competitive with other algorithms, which may benefit from good diversity protection technology. JY4 is constructed as a discontinuous and time-varying POF segment. It can be ob-

<table>
<thead>
<tr>
<th>prob</th>
<th>n, m</th>
<th>PPS</th>
<th>MORA</th>
<th>MORA/DAK</th>
<th>RC</th>
<th>ADNEPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDA1</td>
<td>(25, 10)</td>
<td>8.676</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
</tr>
<tr>
<td>FDA2</td>
<td>(25, 10)</td>
<td>2.008</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
</tr>
<tr>
<td>FDA3</td>
<td>(25, 10)</td>
<td>2.008</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
</tr>
<tr>
<td>FDA4</td>
<td>(25, 10)</td>
<td>2.008</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
</tr>
<tr>
<td>dMOP1</td>
<td>(25, 10)</td>
<td>2.008</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
</tr>
<tr>
<td>dMOP2</td>
<td>(25, 10)</td>
<td>2.008</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
<td>3.496</td>
</tr>
</tbody>
</table>

\[ \dagger \] and \[ \dagger \] indicate ADNEPSO performed significantly better than and equivalently to the corresponding algorithm, respectively.

\[ 540 \] variables instead of linear dependencies, meaning that each variable has a different amount of change \[33\]. The results show that ADNEPSO is also competitive, which may benefit from good diversity protection technology. JY4 is constructed as a discontinuous and time-varying POF segment. It can be ob-
served that MOEA/D-KF and IEC obtained the best IGD value and HVD value in JY4 respectively while our algorithm is slightly poorer than them. One reason we surmise is that adversarial decomposition cannot deal with the discrete POF very well. JY5 has obvious POF change characteristics, and the POF gradually changes from convex geometry to concave geometry, and this characteristic can well examine the performance of our proposed algorithm. We notice that ADNEPSO performed clearly better than other algorithms. Especially, as can be seen from Table 3, among the four decomposition-based algorithms, ADNEPSO significantly improved the distribution, implying that ADNEPSO has superiority for the concave-convex problem. JY6 and JY7 introduce a more complex multimodal characteristic, in which the local optimal number of JY6 changes with time, while JY7 considers the change of POF on the basis of the fixed number of local optimal, which makes them sufficiently challenging. For JY6, it can be observed that MOEAD and ADNEPSO achieved better results. For JY7, ADNEPSO, MOEA/D-KF and IEC all performed well. Perhaps the main reasons are that both MOEA/D and KF have a re-initialization mechanism, and IEC has excellent environmental sensitivity detection mechanism and memory strategy. These method can better avoid falling into local optimal. In addition, the excellent performance of ADNEPSO also shows that the proposed diversity maintenance and introduction strategy is effective. The number of POF geometric segments of JY8 will vary over time. IGD and HVD statistical results show a difference on JY8, FPS had the best IGD performance while ADNEPSO had the best HVD value. As mentioned, the reason for the difference may be that ADNEPSO can better handle the dominant resistance solution, but is slightly inferior to FPS in distribution. Compared with other algorithms, ADNEPSO performed better in terms of convergence and distribution, which means that ADNEPSO also has good adaptability to mixed changes. JY9 is very novel and interesting because it switches cyclically in Type I, Type II, and Type III over time. Compared with other algorithms, ADNEPSO performed better in terms of convergence and distribution, which illustrates that ADNEPSO also has good adaptability to mixed changes.
Figure 5: Evolution curves of average IGD values for eight problems with $n_T=10$ and $\tau_T=30$. 
Figure 6: Evolution curves of average IGD values for eight problems with $n_T = 10$ and $\tau_T = 30$. 
5.3. Analyzing The Evolutionary Processes of The Compared Algorithms

In addition to analyzing the statistical results of IGD, HVD, and SP, we show some more intuitive results, including evolution curves of the average IGD values and POF obtained by six algorithms at different moments, which can more clearly test the algorithms’ comparative performance. The change frequency $\tau_t$ and changing severity $n_t$ of all the algorithms are fixed to (30,10).

Fig.5 and Fig.6 show the IGD evolution curve of all test problems. It can be observed that ADNEPSO maintained a small and stable IGD value during the entire iteration of majority problems, which means ADNEPSO has excellent environmental response capabilities. We also notice that PPS had a high IGD peak in the early stage of the algorithm in some test problems such as
FDA3, dMOP2, and JY1. A possible reason is that the prediction mechanism of the PPS needs to accumulate a lot of historical information in the first few generations. As a result, the performance of the PPS in the early stage of the algorithm is often unsatisfactory, but with more generations, the performance of PPS gradually becomes better and tends to be stable. Additionally, when POF convexity-concavity changes dramatically, such as in JY5, the IGD value of the decomposition-based algorithm had fluctuations (periodic peaks) with environmental changes, while ADNEPSO is more stable, which shows that the adversarial decomposition strategy in dealing with such problems achieved better results. IEC and MOEA/D-KF have similar curve shapes in many test instances (such as FDA2, JY4), which may be because both use MOEA/D-DE
as a static framework. For FDA4, the performance of ADNEPSO is far ahead of the other algorithms, which further shows that ADNEPSO can deal with three-dimensional problems more effectively. In JY3, the FPS performance was not very satisfactory, which may indicate that the FPS is not suitable for handling the non-monotonic decision variables characteristic.

Fig. 9 Fig. 11 show the solution set and true POF obtained by FDA1, dMOP2, JY1, JY5, and JY8 at 21 different moments in sequence, where the solution is marked with red dots, and the true POF is marked with blue curves. Also, it can be seen that ADNEPSO maintained good convergence and distribution at different moments. For FDA1 and dMOP2, MOEA/D was not able to search the entire PF for most of the time. This may be because MOEA/D cannot quickly
track environmental changes. For JY5 and JY8, the population of PPS almost did not converge in the first few generations. In JY5, when the decomposition-based algorithms MOEA/D, MOEA/D-KF deal with the convex POF, due to the fixed weight vector, a large number of solutions converged to the middle part of the POF, which made the overall population distribution uneven. It can be found that MOEA/D-KF and IEC have some dominant resistance solutions in FDA1 and JY8, while ADNEPSO has almost none, possibly benefiting from the two-population mechanism and a more reasonable scalar function.
6. Further Discussion

The influence of different environmental changes on the algorithm and the influence of different components of ADNEPSO are discussed in this section. The influence of different environmental change intensities and higher environmental change frequency on the algorithm, as well as the influence of different components of ADNEPSO are discussed in this section.

6.1. Influence of Change Severity

Like $\tau$, change severity $n_t$ is also an important parameter in DMOPs, which can also affect the performance and effect of the algorithm. In the experiment,
we set \( n_t = 5 \), \( n_t = 10 \), and \( n_t = 20 \) to study the change severity for the algorithm’s impact. The smaller \( n_t \) corresponds to the more drastic changes in the environment, making the problem more challenging. The IGD statistics of all algorithms under different \( n_t \) are recorded in Table 4. From the table, the sensitivity of each algorithm to change severity can be analyzed in detail.

It can be observed from the Table 4 that ADNEPSO performed better under different \( n_t \), and as \( n_t \) increased, the performance of all the algorithms improved. For the JY4 problem, ADNEPSO obtained the best results when the environmental changes were more severe, and the results were slightly worse than MOEA/D-KF and IEC when the environmental changes were gentle. For the JY8 problem, ADNEPSO only obtained the second-best value regardless of the severity. For relatively simple problems, ADNEPSO was not sensitive to changes in severity, which may be due to ADNEPSO’s good dynamic response mechanism and diversity maintenance capabilities. Regrettably, for JY6 and JY7, ADNEPSO lost its advantage in cases of severe environmental changes. One possible reason is that JY6 and JY7 have multi-modal characteristics. In this situation, ADNEPSO is more likely to fall into local optimum when the environment changes severely.

### 6.2. Influence of Higher Environmental Change Frequency

In the previous discussion, in order to verify the effect of changing frequency on algorithm performance, experiments were carried out on FDA, dMOP, and JY test series with \( n_t \) fixed to 10, and \( \tau_t \) set to 20, 25, and 30. In this section, a higher change frequency (\( \tau_t \) is set to 5 and 10) will be applied, and the experimental results are recorded in Table 5.

It can be observed from Table 5 that IEC obtains the best or the second best results in most of the test instances, followed by ADNEPSO, ADNEPSO obtained the remaining the best results and most of the second best results, and finally MOEA/D-KF, while MOEA/D, PPS, and FPS performed poorly. After an in-depth analysis of why the algorithms perform as they do, we believe that the reasons may be explained as follows: Since ADNEPSO needs to update the
<table>
<thead>
<tr>
<th>prob</th>
<th>FDA1</th>
<th>FDA2</th>
<th>dMOP1</th>
<th>dMOP2</th>
<th>dMOP3</th>
<th>JY1</th>
<th>JY2</th>
<th>JY3</th>
<th>JY4</th>
<th>JY5</th>
<th>JY6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(25, 5)</td>
<td>5.16e-3 (3.43e-4)</td>
<td>3.16e-3 (3.16e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
<td>4.51e-3 (4.51e-4)</td>
</tr>
<tr>
<td>(25, 10)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
<td>6.07e-3 (6.07e-4)</td>
</tr>
<tr>
<td>(25, 20)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
<td>7.04e-3 (7.04e-4)</td>
</tr>
<tr>
<td>(25, 50)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
<td>8.01e-3 (8.01e-4)</td>
</tr>
<tr>
<td>(25, 100)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
<td>9.00e-3 (9.00e-4)</td>
</tr>
</tbody>
</table>

† and ‡ indicate ADNEPSO performed significantly better than and equivalently to the corresponding algorithm, respectively.
<table>
<thead>
<tr>
<th>prob (n, m)</th>
<th>PPS</th>
<th>FPS</th>
<th>MORA/D</th>
<th>MORA/D-KF</th>
<th>IEC</th>
<th>ADNEPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDA1</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
<tr>
<td>FDA2</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
<tr>
<td>FDA3</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
<tr>
<td>FDA4</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
<tr>
<td>FDA5</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
<tr>
<td>FDA6</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
<tr>
<td>FDA7</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
<tr>
<td>FDA8</td>
<td>(10, 10)</td>
<td>3.066e-3 (1.935e-4)</td>
<td>3.642e-3 (1.402e-4)</td>
<td>2.392e-3 (1.429e-4)</td>
<td>1.366e-3 (9.873e-5)</td>
<td>1.118e-3 (7.651e-5)</td>
</tr>
</tbody>
</table>

Table 5: Mean and SD of IGD indicator obtained by six algorithms.

† and ‡ indicate ADNEPSO performed significantly better than and equivalently to the corresponding algorithm, respectively.

Population alternately, and this excellent information interaction ability will be inhibited when the number of generations is small. In other words, ADNEPSO needs more generations to "start" than other algorithms. As $\tau_1$ increases, the performance of ADNEPSO improves significantly. For example, in JY7, JY8 and JY9, ADNEPSO has surpassed IEC when $\tau_1=10$. In addition, the memory mechanism seems to be more suitable for a higher frequency of change than the prediction mechanism. In this environment, prediction is more likely to mislead the direction of evolution. This may be the reason for the poor performance of PPS, FPS, and KF, while IEC obtains the best performance by also benefit from an effective memory strategy.
### Table 6: Mean and SD of GD, IGD and HVD indicator obtained by four algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>IGD</th>
<th>IGD</th>
<th>HVD</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FDA1</strong></td>
<td>4.277e-3 (1.415e-5)</td>
<td>4.238e-3 (3.594e-4)</td>
<td>5.790e-3 (3.565e-4)</td>
<td>9.369e-3 (3.965e-5)</td>
</tr>
<tr>
<td><strong>FDA3</strong></td>
<td>3.788e-2 (8.835e-5)</td>
<td>5.725e-3 (5.438e-4)</td>
<td>6.138e-1 (2.776e-2)</td>
<td>3.050e-2 (1.182e-2)</td>
</tr>
<tr>
<td><strong>dMOP2</strong></td>
<td>4.126e-3 (2.706e-4)</td>
<td>4.081e-3 (2.564e-5)</td>
<td>4.474e-3 (1.455e-5)</td>
<td>4.130e-3 (3.966e-5)</td>
</tr>
<tr>
<td><strong>dMOP3</strong></td>
<td>4.101e-3 (2.274e-5)</td>
<td>4.216e-3 (4.799e-5)</td>
<td>4.435e-3 (1.716e-3)</td>
<td>4.101e-3 (5.616e-5)</td>
</tr>
<tr>
<td><strong>JY3</strong></td>
<td>3.148e-1 (2.056e-3)</td>
<td>3.234e-1 (3.421e-3)</td>
<td>3.149e-1 (6.269e-4)</td>
<td>3.148e-1 (3.335e-3)</td>
</tr>
<tr>
<td><strong>JY5</strong></td>
<td>6.260e-3 (2.176e-3)</td>
<td>6.566e-3 (1.751e-4)</td>
<td>3.367e-3 (3.333e-4)</td>
<td>3.414e-3 (3.570e-5)</td>
</tr>
<tr>
<td><strong>JY6</strong></td>
<td>2.831e-1 (1.374e-3)</td>
<td>2.707e-1 (5.837e-2)</td>
<td>6.556e-2 (1.456e-3)</td>
<td>2.035e-2 (1.799e-3)</td>
</tr>
<tr>
<td><strong>JY7</strong></td>
<td>1.431e+0 (5.837e-1)</td>
<td>9.428e-1 (1.546e-1)</td>
<td>8.256e-2 (1.082e-2)</td>
<td>7.506e-2 (1.046e-1)</td>
</tr>
</tbody>
</table>

† and ‡ indicate ADNEPSO performed significantly better than and equivalently to the corresponding algorithm, respectively.

### 6.3. Analysis of The Influence of Different Components of ADNEPSO

In this section, we mainly study the effect of different parts of the proposed algorithm on the algorithm. To explore this issue, we have given three additional versions of ADNEPSO. ADNEPSO-s1 does not use adversarial decomposition, but only uses ideal point vectors. ADNEPSO-s2 uses a completely random neighborhood selection method, while ADNEPSO-s3 uses a randomly initialized dynamic response mechanism. The statistical results of the three indicators in the four versions are shown in Table 5. All algorithms have $\tau_t$ set to 30 and $n_t$ set to 10.

From the statistical results, we find the performance of ADNEPSO is significantly better than the other three versions, which shows that each component of ADNEPSO has an effect on the performance of the algorithm. Comparing ADNEPSO-s1 and ADNEPSO, the SP index of the former is significantly worse than the latter, which denotes that adversarial decomposition mainly improves...
distribution and diversity. Comparing ADNEPSO-s2 and ADNEPSO, it is clear that the SP indicators of the two versions are the closest, especially in the FDA and dMOP series, while ADNEPSO performs better on IGD and HVD, which may mean that the proposed neighborhood selection strategy improves convergence, but the effect on distribution and diversity is not obvious. Comparing ADNEPSO-s3 and ADNEPSO, the results of the two differ the most in all versions, especially when dealing with multi-modal problems JY6 and JY7, which shows that a suitable dynamic response mechanism is very important for handling DMOPs, especially difficult in convergence and optimization problems. In general, each component of ADNEPSO plays its own role. In the static process, the algorithm accelerates the convergence of the algorithm through the neighborhood particle optimization of two adversarial populations and the parent selection strategy to ensure that the current optimal POF and POS are found as often as possible before the change. When changes in the environment occur, a dynamic response mechanism is initiated, using the historical centroid for information prediction. At the same time, in order to ensure the diversity of the new population, some particles are retained and potential areas are developed through global optimal information. The interaction of several components allows the algorithm to effectively respond to environmental changes and achieve rapid convergence.

7. Conclusions

In this paper, a novel particle swarm optimization algorithm based on adversarial decomposition and neighborhood evolution, ADNEPSO, has been proposed for solving dynamic multi-objective optimization problems. In order to solve the limitations of traditional decomposition methods when dealing with DMOPs and improve the stability of the response environment, the proposed algorithm introduces an effective adversarial decomposition strategy to maintain two population updates. In addition, a novel neighborhood update method has been proposed to improve the convergence ability of the algorithm by selecting the more optimal neighborhood optimal particles. In dealing with environmental
changes, in order to ensure the convergence and diversity of the new population, the dynamic response mechanism includes three parts: archive prediction, development of global optimal information, and retention of excellent particles. Experimentally, the proposed algorithm and the other four state-of-the-art algorithms have been evaluated by three performance metrics. The results show that ADNEPSO is very competitive in dealing with DMOPs, but at the same time it is dealing with severe environments and discontinuities problems are urgent to improve. Furthermore, our future work will design more adaptive methods. In addition, applying decomposition-based methods to DMOPs is a very promising direction, which is also worth further investigation.

Acknowledgement

This work was supported by the research projects: the National Natural Science Foundation of China under Grant Nos. 61502408, 61673331, 61772178 and 61403326, the postgraduate research and innovation Project of Hunan Province under Grant No. XDCX2019B057.

References


[3] A. Altan, Performance of metaheuristic optimization algorithms based on swarm intelligence in attitude and altitude control of unmanned aerial ve-


[26] M. Li, X. Yao, What weights work for you? adapting weights for any pareto front shape in decomposition-based evolutionary multiobjective op-


