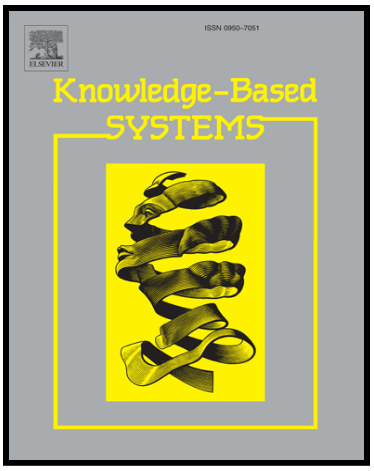
Cellular Teaching-Learning-Based Optimization Approach for Dynamic Multi-Objective Problems

Amin Birashk , Javidan Kazemi Kordestani ,
Mohammad Reza Meybodi

PII: S0950-7051(17)30538-5
DOI: 10.1016/j.knosys.2017.11.016
Reference: KNOSYS 4111

To appear in: *Knowledge-Based Systems*

Received date: 22 March 2017
Revised date: 9 November 2017
Accepted date: 12 November 2017



Please cite this article as: Amin Birashk , Javidan Kazemi Kordestani , Mohammad Reza Meybodi , Cellular Teaching-Learning-Based Optimization Approach for Dynamic Multi-Objective Problems, *Knowledge-Based Systems* (2017), doi: 10.1016/j.knosys.2017.11.016

Cellular Teaching-Learning-Based Optimization Approach for Dynamic Multi-Objective Problems

Amin Birashk[a], Javidan Kazemi Kordestani[a,*], Mohammad Reza Meybodi[b]

[a]*Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*

[b] *Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran*

a.birashk@gmail.com, javidan.kazemi@gmail.com[*], mmeybodi@aut.ac.ir

Abstract

Many real-world optimization problems involve several conflicting objectives that must be optimized simultaneously. Furthermore, most optimization problems have a dynamic structure and change over time. In addition to trying to establish trade-offs among conflicting objectives and explore a diverse set of solutions on a Pareto-optimal front, a dynamic multi-objective optimization (DMOO) algorithm tries to detect changes and track them, using the knowledge of prior environments to converge to the new Pareto-optimal front more quickly. In this paper, a cellular automata-based approach is first proposed for managing and evaluating solutions during the optimization process. Then, using the above approach and the teaching-learning-based optimization algorithm, two new algorithms are introduced for DMOO problems. The first algorithm works to optimize the objectives all at once in a multi-objective manner, while the second algorithm uses the vector evaluated technique to evolve solutions in collaborative single-objective optimization units, and then analyzes them from a multi-objective perspective. These algorithms have been evaluated and compared with some other DMOO algorithms for some standard benchmark problems. The results indicate their superiority in many of the experiments.

Keywords: *Dynamic multi-objective optimization problems, Multi-objective teaching-learning-based optimization, TLBO, Vector evaluated.*

1. Introduction

Optimization plays a crucial role in almost all fields of science and engineering. Many real-world optimization problems are typically composed of multiple conflicting objectives such that optimizing the solutions of one objective may reduce the optimality of the solutions provided for the other objectives. Therefore, unlike single-objective problems that have a single optimal solution, these problems have a set of optimal solutions called the Pareto-optimal set (POS).

The corresponding points of POS in the objective space, form a set which is called the Pareto-optimal front (POF). Each available optimal solution in the POS is desirable for different objectives with an associated ratio. Users adopt one of the solutions based on the relative importance they assign to the objectives. In multi-objective optimization (MOO), the performance of solutions relative to one another is determined by the concept of domination. According to this concept, a solution dominates another solution if it results in an equal or better value for each of the objectives and a better value for at least one of them. MOO algorithms operate based on the domination concept and always try to achieve a set of solutions that are uniformly distributed on the POF.

Many meta-heuristic algorithms have been introduced for MOO. Zhou et al. [1] classified multi-objective evolutionary algorithms (MOEAs) into six categories:

I. **MOEAs based on decomposition:** This approach is based on conventional aggregation approaches in which an MOO problem is decomposed into a number of single-objective optimization problems [2], [3].

II. **MOEAs based on preference:** The decision-maker may be only interested in preferred solutions, and therefore these algorithms try to guide the search towards the region of the Pareto front of interest to the decision-maker [4].

III. **Indicator-based MOEAs:** Indicator-based MOEAs use an indicator such as hypervolume [5] to guide the search, particularly to perform solution selection [6].

IV. **Hybrid MOEAs:** Many MOEAs are formed by hybridizing multiple techniques to utilize their advantages for dealing with complicated problems [7], [8].

V. **Memetic MOEAs:** As a special case of hybrid MOEAs, memetic MOEAs incorporate local search methods [9].

VI. **MOEAs based on coevolution:** These algorithms evolve multiple subpopulations simultaneously and, according to one explanation, they use the idea of dividing and conquering to break down a problem into a set of sub-problems at the level of individual coding [10], [11].

In addition to the above MOEAs, some researchers have applied other types of metaheuristics to deal with MOO. For example, VEPSO [12] and maximinPSO [13] use a PSO algorithm in their search phase. NICA [14] is another MOO algorithm which benefits from immune clonal operation. An Alife-inspired algorithm proposed by Amato and Farina [15] for MOO problems, is another example of these types of algorithms.

In real-world applications, most MOO problems change over time and form a branch of optimization known as dynamic multi-objective optimization (DMOO). With the change in environment, previous solutions lose their credibility. These changes are usually small, and the environment is not altered completely. Therefore, algorithms can use the knowledge acquired in previous environments to track changes and converge more quickly to the new POF. Accordingly, dynamic optimization algorithms try to detect the changes and then discover the new position of desirable solutions using their acquired knowledge.

A typical DMOO problem can be described as follows:

$$\begin{aligned} \text{Minimize:}\quad & f(\vec{x},t) = \{f_1(\vec{x},t), f_2(\vec{x},t), \dots, f_{n_k}(\vec{x},t)\} \\ \text{Subject to:}\quad & g_i(\vec{x},t) \le 0 \ ,\ i = 1 \dots n_g \\ & h_j(\vec{x},t) = 0 \ ,\ j = 1 \dots n_h \\ & x_i \in [lb_i, ub_i]^D \end{aligned} \tag{1}$$

where f is a set of objectives at time t, $\vec{x} = (x_1, x_2, \dots, x_D)$ is the *D*-dimensional decision variable vector, g and h are functions that determine the constraints of the problem at time t, and lb_i and ub_i are the box constraints corresponding to the i^{th} dimension of the search space, that is, $\forall\ i \in \{1, \dots, D\}, -\infty < l_i < u_i < \infty$.

In DMOO problems, as in MOO problems, the concepts of dominance and Pareto optimality are relevant. The most important concepts are defined as follows:

Pareto Dominance: A decision vector x_1 is said to dominate another decision vector $x_2 (x_1 \prec x_2)$ at time t if both of the following conditions are satisfied:

- For all objectives, the solution x_1 is no worse than the solution x_2.
  $$f_k(x_1,t) \le f_k(x_2,t), \quad \forall k = 1, \dots, n_k \tag{2}$$
- For at least one objective, the solution x_1 is strictly better than the solution x_2.
  $$f_k(x_1,t) < f_k(x_2,t), \quad \exists k = 1, \dots, n_k \tag{3}$$

where n denotes the number of objectives.

Dynamic POS: the POS at time *t*, denoted *POS*POS_t, is the union of all Pareto-optimal decision vectors at time *t*:

$$POS_t = \{x_t^* \in F | \nexists x_t \in F : x_t \prec x_t^*\} \tag{4}$$

where *F* denotes the feasible search space, and x_t^* refers to a Pareto-optimal solution at time *t*.

Dynamic POF: for the objective vector $f_t(x)$ and Pareto-optimal set POS_t, the POF at time t is defined as:

$$POF_t = \left\{f_t = \left(f_1(x^*, t), \dots, f_{n_k}(x^*, t)\right) | x^* \in POS_t\right\} \tag{5}$$

In DMOO problems, environments are classified into four types [16]:

- Type I: POS changes while POF remains unchanged.
- Type II: POS and POF both change.
- Type III: POF changes while POS remains unchanged.
- Type IV: POS and POF remain unchanged but other aspects of the problem such as constraints are subject to change.

Generally, only the first three types are considered in designing DMOO algorithms, and this is the case in this research also.

In this paper, a new method based on cellular automata (CA) is first proposed to manage and evaluate solutions and to maintain the diversity of solutions in MOO. Then, two new algorithms are presented for DMOO, both of which use a teaching-learning-based optimization (TLBO) algorithm in the process of optimization as well as a CA-based approach to managing solutions.

The first algorithm performs the optimization process in multi-objective manner, while the second algorithm benefits from the vector evaluated method to optimize solutions in some collaborative units in a single-objective manner, and evaluates the solutions in a supervisor unit.

2. Background and Related Work

In this section, different metaheuristic optimization algorithms for DMOO problems are discussed, according to the classification proposed in [17]. Algorithms that aggregate the objective functions and transform a problem to a dynamic single-objective problem are not considered here.

2.1. Algorithms Without Adapting the Problem

2.1.1. Adapted Static MOO Algorithms for DMOOPs

One of the first algorithms proposed to solve DMOO problems without using a weighted aggregation of objectives was HMCEDA, which was introduced by Farina et al. [16]. This hybrid algorithm uses a two-membered evolution strategy for global optimization. Once the evolution strategy begins to converge, a gradient-based algorithm or a simplex Nelder-Mead search algorithm [18] is used.

Avdagic et al. [19] extended the MOGA algorithm to solve DMOO problems. MOGA [20] was the first genetic algorithm-based multi-objective algorithm. The use of Pareto ranking results is a weakness of this algorithm, because assigning equal fitness to all the solutions of each dominated front, without considering the density of solutions, may lead to unwanted bias.

Deb et al. [21] extended the NSGA-II algorithm [22], [23], one of most successful algorithms in MOO, for DMOO problems. In their approach, after detecting a change in the environment using sentinel solutions, first the entire population is re-evaluated and then one of the following two steps is taken to increase diversity and track changes: (*a*) a part of the population is replaced with random solutions (DNSGA-II-A) or (*b*) a part of the population is randomly selected and mutated (DNSGA-II-B).

Shang et al. proposed the CSADMO algorithm [24], based on clonal selection theory. At each iteration, it uses a clonal selection operator [25] on an antibody population to form a new antibody population. Then, the clonal mutation operator applies nonuniform mutation [26] to the antibody population. Next, the algorithm finds all non-dominated solutions among the mutated antibodies and their parents. The non-dominated solutions form a new antibody population and the dominated solutions are eliminated. The algorithm uses the crowding distance [22] criterion to prune the archive of non-dominated solutions.

In 2013, Shang et al. introduced QICCA for DMOO problems. QICCA is a multi-population algorithm that uses “clonal selection” and “immune clonal” operators on an artificial immune system. In this algorithm, there is a competition and cooperation process between populations for exchanging information. In addition, a quantum updating operation was introduced for this algorithm to increase the population search ability and improve the distribution quality of non-dominated solutions [27].

Zhang proposed an artificial immune system-based algorithm called MOIA [28]. Immune operators are used to enable the adaption of the algorithm to the dynamics of its environment. Furthermore, Zhang introduced an environment-recognition rule that classifies the current environment as a new, similar or identical environment in comparison to the previous environment. Based on the result of this classification, the algorithm decides which operators to use.

Zeng et al. [29] introduced an evolutionary algorithm based on an orthogonal design. It uses two types of crossover operators: orthogonal crossover [29] and linear crossover [30]. The algorithm selects one of these operators randomly and applies it to a randomly selected pair of parents. Moreover, a clustering technique is used to select solutions for a new population.

Zheng proposed a new DMOO algorithm called DMOEA [31]. DMOEA uses geometrical Pareto selection to manage the archive of non-dominated solutions. In this approach, the slope of the line that connects the solution to the auxiliary point is calculated for each non-dominated solution. A candidate solution will be added to the archive if there is no solution in the same slope region in the archive, otherwise, it will replace a solution from same slope region only if its distance to the auxiliary point is greater.

Tan and Goh introduced the competitive-cooperative EA (COEA) to solve DMOO problems [32]. In COEA, each subpopulation optimizes only one decision variable. The most suitable subpopulation for each decision variable is selected using a competitive mechanism. Then, subpopulations optimize their own specific decision variable during the cooperative phase of the algorithm.

In another paper, these authors extended COEA to detect and track changes [33], referring to this new version as a dynamic COEA (dCOEA). After detecting a change, the algorithm starts the competitive mechanism. If a stochastic solution wins over the representative solution of a subpopulation, the subpopulation is re-initialized in the region from which the winner is sampled. It also stores a fixed number of archive solutions in a memory and uses some of them randomly in subsequent environments.

Multi-strategy ensemble MOEA (MS-MOEA) was introduced by Wang and Li [34]. After detecting changes, each population solution is re-initialized either by using a random position or by adding values from a Gaussian distribution. The algorithm uses either (i) SBX and PM or (ii) DE operators, in the evolution process.

IDMOEA is a DMOO algorithm introduced by Chen et al. [35]. IDMOEA considers diversity as an additional optimization objective. After detecting changes, a new population is created using the best solutions from the population and the archive. In the evolution phase, each new solution is generated by applying crossover to two selected parents and then mutation to their offspring. IDMOEA manages the size of its archive by using the magnitude of diversity measures.

Liu and Zeng proposed DBOEA for solving DMOO problems [36]. This algorithm uses the DE/rand/1/bin [37] operator and fast bi-objective non-dominated sorting.

DNMOEA/HI is another DMOO algorithm, introduced by Li et al. [38]. It uses evolutionary operators and benefits from an HV indicator to prune the excess population. DNMOEA/HI combines the Pareto strength value with a density estimator to perform fitness assignment.

Lechuga proposed the first PSO-based DMOO algorithm by extending the MOPSO [39] for dynamic problems [40]. In this algorithm, after detecting changes, the personal best position of the particle is set to the current position if it dominates the current position, or in some cases without specifying any conditions.

Li et al. extended the maximinPSO algorithm for solving DMOO problems [13]. The maximinPSO algorithm uses a fitness function based on the maximin strategy of game theory, and takes dominance and diversity into account. The adapted maximinPSO uses the social-only model of PSO and has no change detection procedure.

Greeff and Engelbrecht extended the VEPSO [41] algorithm for DMOO problems [12]. The new algorithm, DVEPSO, has multiple sub-swarms, and each sub-swarm attempts to optimize one objective using the PSO algorithm. According to a topology, knowledge is shared between the sub-swarms and used to update the velocity of particles.

2.1.2. New Population-Based Algorithms Specifically Introduced for DMOOs

Amato and Farina introduced an Alife-inspired algorithm to solve DMOO problems [15]. The individuals of the algorithm can reproduce, meet each other or fight with each other. Each individual meets every other individual with equal probability. In a meeting, the individuals either reproduce or fight. Reproduction results in two offspring that are added to the population. If fighting occurs, the loser individual that is dominated by the other, or the individual that has more neighbors in a specified neighborhood, is removed from the population.

DMOAP is another new algorithm proposed by Huan et al. solely for DMOO problems [42]. This algorithm is based on P systems. A P system is a membrane structure whose membranes contain objects that can evolve. DMOAP consists of membranes that are grouped into m+1 subsystems in solving an m-objective problem. Each of the m sub-systems optimizes one of the objectives in a single-objective manner, while the final sub-system optimizes all objectives synchronously.

2.2. Algorithms Converting Dynamic MOO Problems into Multiple Static MOO Problems

Wang and Dhang introduced an approach that divides the time period of the DMOO problem into several time intervals [43]. In each interval, the problem is considered as a static MOO. Each static MOO is transformed to a two-objective MOO whose objectives are increasing the diversity and the quality of non-dominated solutions.

Isaacs et al. proposed a memetic algorithm for solving DMOO problems [44]. This MA algorithm uses SQP to evolve the individuals. The algorithm optimizes each objective separately. The range of each objective is subdivided into small intervals, and a single-objective problem is solved for each interval. The algorithm employs an orthogonal epsilon-constrained formulation to solve MOO problems using the SOO problems described above.

DMEA, which was introduced by Liu and Wang [45], [46], divides a single time period into several equal subperiods. In each subperiod, the DMOO problem is approximated by a static MOO problem. A new evolutionary algorithm, based on a new mutation operator, is used for the above static optimizations. The new mutation operator also helps the algorithm by automatically checking for changes.

2.3. Prediction-Based Approaches

Some DMOO algorithms use prediction approaches to speed up convergence by predicting new POSs after environmental changes. In addition, the success of this approach causes the algorithm to use fewer objective-function evaluations to achieve a certain level of convergence; this is valuable especially where objective functions are not easily available or computationally expensive.

Hanzakis and Wallace introduced the D-QMOO algorithm for DMOO problems [47]. D-QMOO is a steady-state clustering MOEA. Each cluster has a population that is subdivided into two subpopulations, referred to as the front and the cruft. The front and cruft contain non-dominated and dominated solutions respectively. Pruning for the front subpopulation is applied in such a way that the maximum HV is preserved. After detecting a change, forecasting is performed for some selected points on the POF using stochastic time series models.

Liu extended the DMEA [45] algorithm by using a core estimation of distribution model to estimate the POS in a new environment [48]. The proposed algorithm adds a transfer vector of average points in the last two environments to some of last non-dominated solutions, to generate estimated solutions for the new environment.

In 2008, Talukder and Kirley extended a variation operator [49] for DMOO algorithms [50]. The variation operator approximates the next POF and uses a Fourier transformation to inversely map the approximated POF to the POS. The authors used this in the NSGA-II [22] algorithm, by adding it after non-dominated sorting to create new predicted individuals in each iteration.

MO-EGS[51] is another MOO algorithm that was adapted by Koo et al. for dynamic problems [52]. The new algorithm estimates the direction and magnitude of the next change based on the history of previously discovered solutions, using a weighted average approach. Thus, the estimated direction and magnitude, which are calculated for the centroid of the archive, are used to update solutions and generate predicted solutions.

Zhou et al. introduced a population prediction strategy (PPS) that predicts the new position of the population [53]. In the above approach, the Pareto set is divided into two parts: a center point and a manifold. After detecting a change, the center points and the manifolds of previous environments are used to predict the next center and manifold respectively.

Liu et al. introduced a novel cooperative coevolutionary DMOO algorithm referred to as PNSCCDMO [10]. The main idea of the new approach is that it allows the decomposition of the optimization problem according to the search space of decision variables. Therefore, the algorithm groups variables according to the similarity of their constraints. Each group has its own population and uses simulated binary crossover [54] and polynomial mutation [55] operators for the evolution process. During evolution, groups cooperate with each other. Furthermore, the algorithm uses a modified linear regression prediction strategy to generate predicted solutions after detecting changes.

3. TLBO Algorithm

The TLBO algorithm is a metaheuristic optimization algorithm proposed by Rao et al. [56] that is inspired by the process of teaching-learning in classrooms. The population is considered as members of a classroom. The process of the TLBO algorithm is divided into two distinct phases called the “teacher phase” and the “learner phase”. In the teacher phase, students acquire knowledge from the teacher, and in the learner phase, the students try to increase their knowledge through interaction among themselves. Each phase of TLBO is presented in detail below.

3.1. Teacher Phase

A good teacher is someone who tries to promote the knowledge level of students up to his or her own knowledge level. In practice, this is impossible, and a teacher can only partially increase the average knowledge of students. This increase depends on several factors. If M^k and T^k signify the points of "average knowledge of students" and "knowledge of teacher" respectively, then in the kth iteration of the TLBO algorithm, T^k tries to attract M^k to itself. Thus, in each iteration, the transfer vector of each solution is calculated as follows:

$$Difference_i^k = rand \times \left(T^k - TF \times M^k\right) \quad (6)$$

where *rand* denotes a random number from the interval [0,1], i is the index of the student and *TF* signifies the teaching factor. The teaching factor decides on the change of mean value. One of two values, 1 or 2, is selected randomly with equal probability for *TF* as follows:

$$TF = \text{round}[1 + rand \times (2-1)] \quad (7)$$

Thus, according to the following equation, the transfer vector is used to update the solutions:

$$x_{i,\text{new}} = x_i^k + Difference_i^k \quad (8)$$

Finally, $x_{i,\text{new}}$ replaces x_i^k if it has a better fitness value.

3.2. Learner Phase

In addition to training by the teacher, students can increase their knowledge through interaction with each other. To model the interactions of students, for each student (x_i^k), another student (x_j^k) is chosen randomly ($i \neq j$). Then, the new solution is calculated as follows:

$$x_{i,\text{new}} = \begin{cases} x_i^k + rand \times \left(x_i^k - x_j^k\right) & if\ f\left(x_i^k\right) < f\left(x_j^k\right) \\ x_i^k + rand \times \left(x_j^k - x_i^k\right) & if\ f\left(x_j^k\right) < f\left(x_i^k\right) \end{cases} \quad (9)$$

Finally, $x_{i,\text{new}}$ replaces x_i^k if it has a better fitness value.

Satapathy and Naik [57] introduced an improved TLBO based on the opinion that the teacher has the most influence on students' improvement. Therefore, they attempted to model the more-effective role of the teacher in the learner phase by adding a component to equation 9 as follows:

$$x'_{i,\text{new}} = x_{i,\text{new}} + 0.5 \times (1 + rand) \times \left(T^k - x_i^k\right) \quad (10)$$

4. Cellular Automata

Cellular automata (CA) are abstract dynamic systems consisting of a large number of identical simple cells that are distributed in a grid-like structure and can produce complex phenomena using some simple rules [58]. Time passes in discrete steps, and at each instant of time each cell has one state out of a finite set of states. The new state of each cell is determined using the local rules of the CA. The local rules, which are usually the same for all cells of these automata, determine the new state of the cell based on its previous state and the previous states of some of its neighboring cells. In the other words, CA rules determine how cells are influenced by their neighboring cells. In this way, each cell is limited to local interactions with its neighbors, and therefore it cannot handle fast overall communications.

In basic CA, the states of all cells in each time step are updated simultaneously. The state of all cells together determines the overall state of a CA. The overall structure of a CA can be thought of as similar to a parallel-processing computer. The simple structures of CA produce complex patterns after a few steps of growth, which shows their potential to simulate complex natural phenomena [58].

5. Proposed Algorithms

The process of solving DMOO problems is composed of two main phases. The first phase is related to MOO, when the algorithm tries to find a set of solutions that are diversely distributed and are as close as possible to the POF. The second phase is executed after detecting a change in the environment, when the algorithm tries to track the changing POF and converge faster by using its knowledge of previous environments. In this section, a CA-based method is introduced for evaluating and managing solutions. Then, two new algorithms that are referred to as cellular TLBO (CTLBO) and vector evaluated CTLBO (VECTLBO) are introduced for solving DMOO problems.

5.1. CA-Based Approach

In MOO, determining the relative desirability of solutions and extracting a diverse set of non-dominated solutions from the population are steps of particular importance. The method introduced in this section uses CA simultaneously to detect non-dominated solutions, describe the relative desirability of solutions and manage the diversity of non-dominated solutions. The structures of CA have great potential to benefit from parallel processing, and the CA-based approach can use this feature to decrease processing time. In this approach, the CA is defined according to the objective space and embedded in it. At each iteration of the optimization algorithm, the CA evolves according to predefined rules, and this evolution results in some analysis for the region of each cell and its solutions. The following section describes the rules and their aims in detail.

5.1.1. Rules of CA

Each cell of a defined CA consists of four sub-states. In this section, the rules of these sub-states are introduced.

- Activator rule

  The first sub-state of the cell is activation. The active cell is a cell that has been processed in previous steps of evolution, and all its sub-states are determined. The activator rule is the first rule that is applied to inactive cells. The remaining rules will apply to the current cell only if the cell becomes active after applying the activator rule. Thus, if a cell is active, all its sub-states are specified and it will not be evaluated in the next generations. The following describes the activator rule.

  At each iteration of the optimization algorithm, a CA is formed based on the current population of the algorithm and then evolved using the neighborhood shown in Figure 1. This neighborhood is defined based on the domination concept of MOO. Obviously, the neighborhood can be extended in a straightforward way for objective spaces with higher dimensions.

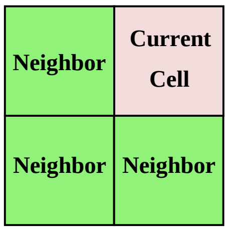


Figure 1: Neighborhood used in the CA-based approach

  The activator rule is defined as follows:

  Activator rule: The current cell will be activated only if all its neighbors are active:

$$\text{Active}(A)=\begin{cases}1 & \forall C \in \text{Neighbor}(A): \text{Active}(C) = 1\\ 0 & \text{otherwise}\end{cases} \quad (11)$$

- Domination rule

The other sub-state of cells is the domination sub-state, where cells can exist in one of three states: “dominated”, “non-dominated” and “neutral.” The neutral cells are the cells that are not in the domination area of the Pareto front. Before starting the evolution process, an auxiliary row in which all the cells are active and neutral is considered for each dimension of the CA. Therefore, a cell will be neutral if there is no solution in it and all its neighbors are neutral. If the current cell includes any solution and all its neighbors are neutral, it will be labeled “non-dominated”. Finally, the “dominated” label is assigned to a cell if it has at least one dominated or non-dominated neighbor. In Figure 2, the cells labeled A, B and C are neutral, non-dominated and dominated cells respectively.

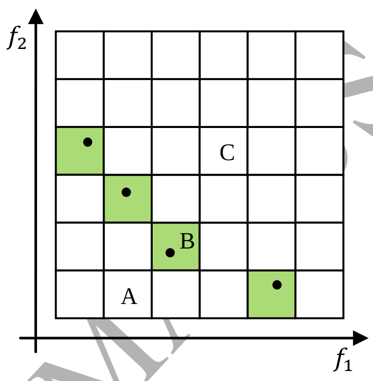


Figure 2: Examples of neutral (A), non-dominated (B) and dominated (C) cells

Thus, by applying the domination rule, dominated and non-dominated cells are identified. The domination sub-state helps the algorithm to find non-dominated solutions and provides basic information for the proximity sub-state. The domination rule is defined as follows:

Domination rule: Assuming that “ndom” and “dom” indicate “non-dominated” and “dominated” labels respectively, and N∈Neighbor(A), the domination rule is defined as follows:

$$\mathrm{Dom}(A) = \begin{cases} \mathrm{ndom} & \exists Sol \in A \text{ and } \forall N{:}\, Dom(N) = \mathrm{neutral} \\ \mathrm{dom} & \exists N{:}\, \mathrm{Dom}(N) \neq \ \mathrm{neutral} \\ \mathrm{neutral} & \mathrm{otherwise} \end{cases} \qquad (12)$$

- Proximity rule

The proximity of a solution to the POF is a descriptor of its quality. In this way, the proximity of a cell to the Pareto front can be used as an approximation of its solution’s proximity. Figure 3 demonstrates the proximity of cells by an example. In this figure, cell A is closer than cell B to the Pareto front; therefore, solutions in cell A are more desirable.

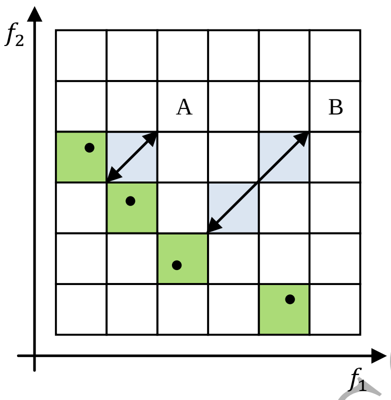


Figure 3: Proximity comparison of two cells (A, B)

The proximity rule is designed with the aim of approximating the distance between the solution and the POF. The distance value for non-dominated cells is set to zero, and it is calculated for dominated cells based on number of cells between the cell and the nearest non-dominated cell. The proximity rule is defined as follows:

Proximity rule: Assuming that the proximity sub-state of cell A is indicated by P_A, the value of the sub-state is determined using the following equation:

$$P_A = \begin{cases} 0 & \text{if } A \text{ is ndom} \\ \min(P_{\text{neighbors}}) + \varepsilon & \text{if } A \text{ is dom} \end{cases} \tag{13}$$

Where $P_{\text{neighbors}}$ indicates set of proximity values of neighbors, and ε is a small number.

- Attractor determination rule

The final sub-state of cells is the attractor sub-state. It is defined with the aim of approximating the hypervolume measure for non-dominated solutions, using the CA. If a cell is dominated by only one non-dominated cell, it is attracted to that cell, otherwise, it is non-attracted. Thus, a cell's attractor sub-state indicates its attractor. In Figure 4, cell A and the blue cells are in the attraction area of non-dominated cell A.

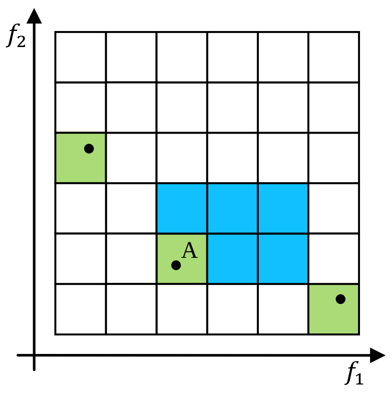


Figure 4: A non-dominated cell and its attracted cells (cell A and blue cells)

Assuming that A and B each indicate a cell in a CA, the attractor determination rule is defined as follows:

Attractor determination rule:

$$\text{Attractor}(A) = \begin{cases} B & A \text{ is dominated only by } B \\ A & A \text{ is a non-dominated cell} \\ \emptyset & \text{otherwise} \end{cases} \tag{14}$$

Accordingly, the domination sub-state helps the optimization algorithm to identify the Pareto front and the dominated area. Furthermore, the proximity sub-state presents a description for the distance of solutions from the Pareto front. The attractor sub-state helps the algorithm approximate the hypervolume measure instead of calculating it, as the calculation process is quite time-consuming.

It is generally expected that the algorithm provides a specified number of non-dominated solutions that are sampled from different parts of the Pareto front. In this paper “archive” refers to a fixed-size set where an algorithm collects its suggested non-dominated solutions. Thus, the archive keeps the output solutions of the algorithm available. The other goal of using CA is managing the diversity of suggested non-dominated solutions in the archive, as described below.

5.1.2. CA and Managing the Diversity of Solutions

As described in the above section, the CA identifies dominated and non-dominated cells during its evolution. Its cellular structure makes it possible for the optimization algorithm to manage the diversity of archive solutions. For this purpose, after detecting non-dominated cells, a primary solution is identified as representative for each of them and added to the archive of suggested solutions. Thus, the algorithm collects a diverse set of non-dominated solutions for the archive. It is obvious that the size of cells plays a considerable role in determining the number of archive solutions and their diversity. By reducing the size of cells, the number of archive solutions and the uniformity of their distribution will be increased, but it also causes an increase in execution complexity because of the increase in the number of cells. Therefore, the size of cells is chosen based on a trade-off between acceptable execution complexity and the expected number and uniformity of archive solutions.

The choice of a primary solution for each non-dominated cell can be made in different ways. For the current approach, a simple method proposed in [59] is used. According to this method, the cell solution that is the closest solution to the minimum point of the cell is chosen for the archive. This is illustrated in Figure 5. As shown in this figure, solution B is closer to the minimum point of the cell and is therefore chosen for the archive. Clearly, any dominated solution such as C in Figure 5 will always have a longer distance than the solution that dominates it and will therefore not be chosen.

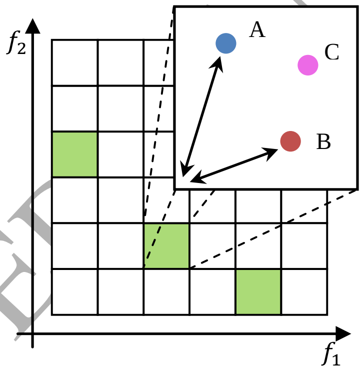


Figure 5: Choosing a representative solution for a non-dominated cell

5.1.3. Challenges of the CA-Based Approach

There are two important challenges for the proposed approach. The first is related to the distribution of the POF in objective space. If points of the POF are presented only in some portions of the objective space, achieving a desirable separability of archive solutions and extracting the specified number of solutions requires smaller cells. This subsequently causes an increase in the number of cells and the execution complexity. The next section introduces a solution to address this challenge.

If a cell is dominated or non-dominated, this does not mean that all its solutions are dominated or non-dominated. This challenge arises in problems that have POFs with high and/or low slopes. This condition causes the Pareto front to pass through several consecutive cells in a row or column. Thus, among these consecutive cells, only the first cell will be considered as a non-dominated cell. Although reducing the size of cells renders it easier to realize the domination correspondence between the cell and its solutions, it causes an increase in execution complexity. To solve this challenge, it is possible to define a multilevel CA and use the additional levels, which have smaller cells, to separate areas with high or low slopes.

5.1.4. Considerations of Configuration and Implementation

The number of cells in the CA and the number of CA generations in each evolution are factors that have a considerable impact on the execution complexity of the CA's evolution. In analyzing the evolution process of the CA, it is understood that in each generation, only cells whose neighbors have been activated in a former generation are activated. Therefore, it is unnecessary to evaluate the activation rule for all cells in each generation. Thus, the evolution process of the CA is implemented in such a way that in each generation, the automaton evaluates only inactive cells whose neighbors are active.

The other solution that is introduced to reduce execution complexity is defining the floating size for dimensions of the CA. According to this solution, a CA is defined in such a way that the number of its cells is fixed and corresponds to an expected number of archive solutions. At the beginning of the optimization process, the CA is embedded in a part of the objective space in which the population of the algorithm is present. During the execution of the algorithm and convergence of non-dominated solutions, the size of the CA is changed in such a way that it is embedded only in promising areas of the objective space, where non-dominated solutions are present. Thus, the size of the CA is reduced gradually in the early steps of the optimization, and finally, it is fixed. Therefore, the size of cells is reduced during convergence, which results in the accurate separability of the Pareto front and favorable management of diversity.

If there is no knowledge about promising areas of the objective space, CA will be embedded in presence area of the population for some steps at the beginning of optimization. Significantly, it avoids premature convergence and failure to identify any part of the POF.

During the optimization process, it is possible that a new generated solution is outside the CA. If the minimum point of the CA does not dominate the above solution, the CA will be changed in the next formation in such a way that it covers the area where the above solution is present. If the new solution is dominated by the minimum point of the CA, however, it will not cause changes in the CA. Thus, it is possible that discovering new parts of the Pareto front causes an increase in the size of the CA in some steps.

The other method which tries to reduce execution complexity applies some changes to the attractor determination rule and the evolution process. Accordingly, the attractor determination rule is redefined as follows:

Attractor determination rule (modified version):

$$\text{Attractor}(A) = \begin{cases} B & A \text{ is dominated only by } B \text{ and } B \text{ is dominated only by one neighbor} \\ A & A \text{ is a non-dominated cell} \\ \emptyset & \text{otherwise} \end{cases} \tag{15}$$

Figure 6 illustrates a non-dominated cell and its attracted cells. In this figure, cell A and the blue cells are attracted to cell A.

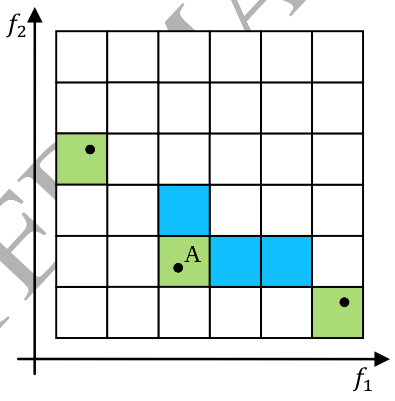


Figure 6: A non-dominated cell (A) and its attracted cells (A and blue cells) according to the new definition

By comparing Figures 4 and 6, it can be concluded that the new definition of the attractor rule is similar to the previous version, and presents values equivalent to HV values of non-dominated solutions. Utilizing the new attractor rule, the evolutionary process will continue until all cells that have been activated in the current generation have taken a value Ø for their attractor sub-state. Thus, a reduction in execution complexity is achieved. Figure 7 illustrates a CA that has evolved under the above considerations. In this figure, the white cells are not evaluated and therefore the execution complexity of their evaluation is eliminated.

The new evolution process causes the proximity sub-state to be unspecified for unprocessed cells. The algorithm convergence usually occurs in such a way that the majority of solutions are in cells close to the Pareto front. Thus, assigning a large value for the proximity sub-state of unprocessed cells is not unreasonable.

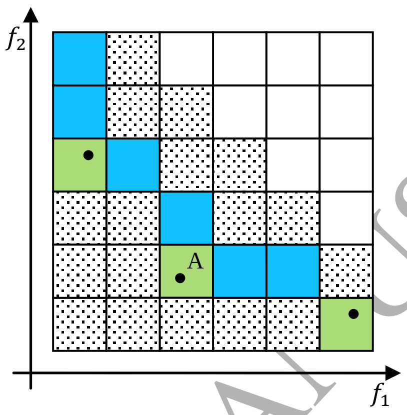


Figure 7: Final state of CA after evolution under new considerations.
White cells have not been evaluated.

5.2. CTLBO Optimization Algorithm

In this section, a new algorithm based on the TLBO algorithm is introduced for DMOO. This algorithm optimizes all objectives simultaneously without aggregation or decomposition. It uses the CA-based approach proposed above for ranking and managing solutions; hence, it is called cellular TLBO or CTLBO.

As usual, the algorithm starts by initializing the population with randomly generated solutions from the search space. Then, to gather basic knowledge about the environment, a few iterations of single-objective TLBO are performed for each objective. After that, the best solution of each optimization is added to the population.

According to the idea of the TLBO algorithm, CTLBO performs optimization by taking inspiration from the learning process in the classroom. Each iteration of CTLBO represents a certain number of classes, and a non-dominated solution is chosen as the teacher for each class. A few iterations of multi-objective TLBO are performed for each class. Afterward, the final populations of the classes and the main population of CTLBO are gathered in a set to be evaluated.

In each iteration of the algorithm, new classes are formed and the teacher of each class is selected from the archive based on the approximated HV values. A larger HV value for a solution indicates that the density of the archive solutions is lower in the area near the solution; hence, the algorithm selects solutions with larger HV values as teachers in the hope that they will lead the algorithm to discover solutions in previously undiscovered areas of the POF. The students for each class are selected randomly from two sets. A proportion of them are selected from the archive, and the rest are selected from the population. Because students are usually near different parts of the Pareto front, the class set will have a desirable diversity for exploration. In each iteration of the multi-objective TLBO, a new random solution is generated with probability 0.3, which has been obtained empirically, and used as a student in the mean-point calculation of the teacher phase. This introduces diversity and increases the exploration ability of the algorithm. To avoid being stuck in local optima and to increase exploration ability, CTLBO chooses a few solutions from the population randomly in each iteration and applies polynomial mutation to them. The generated solutions are added to the population before the evaluation step.

The evaluation of solutions is performed using the CA-based approach. In this way, by defining the CA according to previous descriptions and embedding it in a specified area of the objective space, each cell surrounds a subspace, and the solutions of the subspace become members of the cell. Then, the evolution process starts and the required sub-states are specified for cells. According to the method described in section 3.2.2, one non-dominated solution is chosen from each non-dominated cell and added to the archive of suggested solutions. As previously mentioned, this selection method works to increase the diversity quality of the archive. Because the archive solutions may be dominated by new solutions, they are evaluated along with the new solutions, and the archive is formed again from the beginning during the evaluation process. The proximity and domination sub-states of the cell provide a quality description for its solutions. According to sections 5.1.1 and 5.1.4, an approximation of the HV measure is calculated based on the attractor sub-state for each non-dominated cell, and the HV value is assigned to a representative solution of the cell in the archive.

The process of identifying environmental changes is performed at the beginning of each iteration. For this purpose, the set of solutions that was present in the previous iteration is re-evaluated. If any objective value has changed for a solution, it means that a change has occurred in the environment.

After detecting any change in the environment, the algorithm executes a procedure that attempts to track changes and speed up convergence to the new POF by using information from previous environments. The default procedure for this step creates a new population by collecting solutions from the previous archive, mutated solutions from the archive, random solutions and the optimum solutions of a local search on the endpoints of the previous Pareto front. Approximately 60% of the new population is selected randomly from the previous archive. After this, the algorithm selects solutions from the archive that preferably have not been selected in the previous step, and applies polynomial mutation to them. These solutions form approximately 10% of the new population. By performing a few steps of Nelder-Mead [18] local search on the endpoints of the Pareto front (optimum solutions of objectives in the archive), the algorithm tries to find the optimum point of each objective. The obtained solutions, equal to the number of objectives, are the next solutions that are added to the new population. Finally, the remaining population is selected randomly from the search space. Due to a lack of knowledge about the quality of solutions, the optimization steps are ignored in this iteration. Thus, after forming the new population, a CA is formed and embedded as in the initial step of the algorithm, and it starts its evolution process to evaluate solutions.

At the beginning of optimization for each environment, a CA is embedded in the population area for specified iterations. After the iterations, the embedding area of the CA is determined based on the location of non-dominated solutions. Thus, the CA focuses on the Pareto front and separates its various parts more accurately.

At the beginning of each iteration, the population size is equal to or less than a predefined maximum limit. Hence, at the end of each iteration and after removing duplicate solutions, if the population size exceeds the maximum limit, greedy pruning based on the quality and density of solutions will be performed on the population. The CA also helps the algorithm in this step by describing the proximity of solutions to the Pareto front and specifying dense areas using cells. Solutions may be duplicated in two forms: they may have the same decision vector or the same objective vector. The algorithm removes duplicate solutions of both forms.

Algorithm 1: CTLBO Algorithm

```
01.   Initialize Population randomly;
02.   Execute single-objective TLBO for each objective and add found solutions to Population;
03.   Determine the sub-space of objective space in which CA is embedded;
```

```
04. Form CA and map all solutions to its Cells;
05. Perform CA growing process;
06. Assign values of each cell to its solutions;
07. num_of_Classes = min(size(Archive), max_num_of_Classes);
08. Choose Sentinel solutions;
09. while algorithm is not terminated do
10.     Detect environmental changes by using Sentinel solutions;
11.     if change occurred then
12.         Perform change adaptation process;
13.     else
14.         Choose new Sentinel solutions;
15.         for each Class do
16.             Greedy choose a teacher based on hypervolume from Archive;
17.             Choose students: greedy (70%) and random (30%) from Population;
18.             Execute multi-objective TLBO for Class;
19.             Apply polynomial mutation on a randomly selected student and add it to output_Population;
20.         end-for
21.     end-if
22.     Determine the sub-space of objective space in which CA is embedded;
23.     Form CA and map all solutions to its Cells;
24.     Perform CA growing process;
25.     Assign values of each cell to its solutions;
26.     Perform elitist pruning for Population;
27.     num_of_Classes = min(size(Archive), max_num_of_Classes);
28. end-while
```

5.3. VECTLBO Optimization Algorithm

In this section, a new DMOO algorithm is introduced that solves the DMOO problem by considering it as many single-objective problems that cooperate together, in a similar way to the VEDE [60] and VEPSO [12] algorithms. In this technique, referred to as "vector evaluated", one population is defined for each objective, and the algorithm optimizes each population for its corresponding objective. After a certain period of performing iterations, each population sends some of its individuals to other populations and receives some individuals from other populations. Thus, in the subsequent iterations, the transferred individuals are optimized for the objective of their target population. This procedure is repeated and populations cooperate by transferring individuals after each period of iterations. The new algorithm introduced in this section is a combination of the above technique and the CTLBO algorithm; hence, it is called vector evaluated cellular TLBO, or VECTLBO.

In fact, VECTLBO is a version of CTLBO that has a single-objective optimization method and tries to optimize solutions for each objective separately. The main idea of vector evaluated techniques is based on the fact that the result of optimizing a non-dominated solution for an objective is a solution that may not be dominated by its parent. This feature is illustrated in Figure 8.

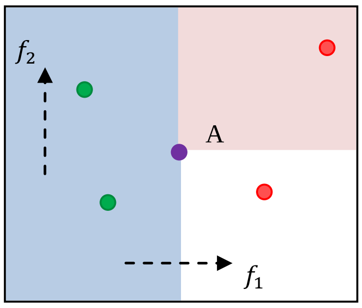


Figure 8: Feasible area of solution A after optimizing it for objective f1. Pink: dominated area. Blue: feasible area.

As shown in Figure 8, the dominated and the feasible areas do not intersect. Therefore, the result of single-objective optimization for a non-dominated solution is a solution that dominates its parent and improves convergence to the POF, or is not dominated by its parent and may help the algorithm find new parts of the Pareto front.

In VECTLBO, the initial population is divided into subpopulations such that the number of subpopulations is equal to number of objectives. Then, for each objective of the problem, the single-objective TLBO is executed and the resulting optimum solution replaces one solution in the corresponding subpopulation. At the beginning of algorithm iteration, a certain number of classes are formed for each objective. The students of each class are optimized during some iterations of the single-objective TLBO for the corresponding objective. Finally, all classes deliver their output solutions to the central unit of the algorithm. The central unit forms the union set of new solutions and the current population. Then, it evaluates all solutions using the CA-based approach. Therefore, the multi-objective quality of each solution is determined compared to all solutions. At the end of each iteration, each subpopulation is pruned greedily and the algorithm prefers to preserve solutions from its non-dominated solutions in the archive, non-dominated solutions, and randomly selected solutions from the union of its new and previous solutions.

As noted above, algorithm iterations are separated into equal periods, and at each period of iterations, the optimization process is performed for each subpopulation independently. At the final iteration of each period, subpopulations begin to exchange some of their non-dominated solutions. Thus, according to a predefined communication topology, each subpopulation sends some of its non-dominated solutions to other subpopulations and receives some solutions from other subpopulations. Once again, a new period of iterations begins and the optimization process is performed independently for each subpopulation. These steps are repeated during the algorithm execution.

The class formation in VECTLBO is different from the CTLBO algorithm in two ways. In VECTLBO, the students are selected randomly from the related subpopulation. There are two approaches to selecting a teacher for a class, and the algorithm adopts one of them randomly. The selection probabilities are not equal, however, and the first approach is more probable. According to the first approach, desirability with respect to the related objective is considered when selecting a teacher among the non-dominated solutions of subpopulations. In the second approach, however, the approximated hypervolume is considered and the non-dominated solutions with larger hypervolume values are selected. Experiments indicate that if the selection probability of the first approach is twice that of the second approach, the desired convergence will be obtained. Thus, in this paper, the probabilities are set to 0.7 and 0.3 respectively.

During optimization in class, and according to single-objective optimization, a new solution will replace its parent if it results in a more desirable value for the optimizing objective; however, because the algorithm cannot judge superiority based on a single objective, all new solutions that are better than their parent for optimizing the objective are sent to the central unit of the algorithm to be evaluated.

In VECTLBO, evaluation of solutions using the CA-based approach and the process of detecting and tracking changes, is performed as in the CTLBO algorithm. It can be implemented according to the pseudocode presented in Algorithm 2.

To establish communication between subpopulations, different topologies can be defined. The topology can be static or dynamic. Depending on the structure and features of the problem, the topologies may obtain different results. As a general approach, forming a random topology in each step can be ideal. It ensures that knowledge is spread gradually in a diverse manner among all subpopulations. If the selected solution for transmission is the best solution of the subpopulation for its relevant objective, a copy of the solution will be created and added to the original subpopulation.

Algorithm 2: VECTLBO Algorithm

```
01.  Initialize sub_Population randomly for each objective;
02.  Execute single-objective TLBO for each objective and add found solutions to Population;
03.  Determine the sub-space of objective space in which CA is embedded;
04.  Form CA and map all solutions to its Cells;
05.  Perform CA growing process;
06.  Assign values of each cell to its solutions;
07.  num_of_Classes = min(size(Archive), max_num_of_Classes);
```

```
08. Choose Sentinel solutions;
09. while algorithm is not terminated do
10.     Detect environmental changes by using Sentinel solutions;
11.     if change occurred then
12.         Perform change adaptation process;
13.     else
14.         Choose new Sentinel solutions;
15.         for each objective do
16.             for each class of objective do
17.                 rnd = a rand number between 0 and 1;
18.                 if rnd< 0.7 then
19.                     Greedy choose one teacher from Archive based on optimality for this objective;
20.                 else
21.                     Greedy choose one teacher from Archive based on approximated value of hypervolume;
22.                 end-if
23.                 Choose students randomly from related subpopulation;
24.                 Execute one-objective TLBO for each Class;
25.                 Add all new desirable Solutions to output_Population;
26.                 Apply polynomial mutation on a randomly selected student and add it to output_Population;
27.             end-for
28.         end-for
29.     end-if
30.     Determine the sub-space of objective space in which CA is embedded;
31.     Form CA and map all solutions to its Cells;
32.     Perform CA growing process;
33.     Assign values of each cell to its solutions;
34.     Perform elitist pruning for Population;
35.     if itr_number is multiple of exchanges_Interval then
36.         Select some non-dominated solutions of each sub_Population randomly;
37.         Send selected solutions to other sub_Population according to defined communication topology;
38.         If one of selections is optimal non-dominated solution of sub_Population for associated objective, one copy of above solution
                will be retained in sub_Population;
39.     end-if
40.     num_of_Classes = min(size(Archive), max_num_of_Classes);
41. end-while
```

6. Experimental study

In this section, the proposed algorithms are evaluated and compared with a number of other DMOO algorithms.

6.1. Experimental Setup

6.1.1. Dynamic Test Functions

In this study, 14 benchmark functions are used to evaluate the performance of the algorithms: FDA1 [16], FDA2$_{\text{Camara}}$ [61], FDA3$_{\text{Zheng}}$ [31], FDA4 [16], dMOP1 [33], dMOP2 [33], dMOP3 [33], F5 [53], F6 [53], F8 [53], ZJZ [62], HE3 [63], HE4 [63] and HE5 [63]. Their definitions and specifications are shown in Table 1.

All these problems are minimization problems, and their dynamic is realized by the following equation:

$$t = \frac{1}{n_t}\left\lfloor \frac{\tau}{\tau_t} \right\rfloor \qquad (16)$$

where τ_t and n_t indicate the frequency and severity of changes in the environment respectively and τ indicates the number of function evaluations or iterations of the algorithm that have been executed up to the current moment. The frequency of change (τ_t) determines how often the environment changes, and the severity of change (n_t) specifies the amount of change in position of Pareto points in search space and/or objective space.

Table 1: Test functions used in experiments in current study

Name	Definition	Search Space	POS & POF	Remarks
FDA1	$f_1(x) = x_1$, $f_2(x,t) = g(\mathrm{x_{II}},t)h(f_1,g)$ $g(x_{\mathrm{II}},t) = 1 + \sum_{x_i \in \mathrm{x_{II}}} (x_i - G(t))^2$ $h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}}$, $G(t) = \sin(0.5\pi t)$	$\mathrm{x_I} = (x_1)$ $\mathrm{x_I} \in [0,1]$ $\mathrm{x_{II}} = (x_2, \dots, x_n)$ $\mathrm{x_{II}} \in [-1,1]^{n-1}$	POS: $\{x \mid \forall x_i \in \mathrm{x_{II}}: x_i = G(t)\}$ POF: $f_2 = 1 - \sqrt{f_1}, 0 \le f_1 \le 1$	type I Two-objective
FDA2 Camara	$f_1(x) = x_1$, $f_2(x,t) = g(\mathrm{x_{II}})h(\mathrm{x_{III}},f_1,g,t)$ $g(x_{\mathrm{II}}) = 1 + \sum_{x_i \in \mathrm{x_{II}}} {x_i}^2$ $h(\mathrm{x_{III}},f_1,g,t) = 1 - \left(\frac{f_1}{g}\right)^{G(t)}$, $H(t) = z^{-\cos(\pi t/4)}$ $G(\mathrm{x_{III}},t) = H(t) + \sum_{x_i \in \mathrm{x_{III}}} (x_i - H(t)/2)^2$	$\mathrm{x_I} = (x_1)$ $\mathrm{x_I} \in [0,1]$ $\mathrm{x_{II}} = (x_2, \dots, x_{r_2})$ $\mathrm{x_{III}} = (x_{r_2+1}, \dots, x_n)$ $[\mathrm{x_{II}}, \mathrm{x_{III}}] \in [-1,1]^{n-1}$	POS: $\{x \mid \forall x_i \in \mathrm{x_{II}}: x_i = 0$, $\forall x_i \in \mathrm{x_{III}}: x_i = \mathrm{c}, G(\mathrm{c}) \le G(d)$, for all $d \in [-1,1]\}$ POF: $f_2 = 1 - f_1^{\min_{\mathrm{x_{III}}} G}$ $, 0 \le f_1 \le 1$	type II Two-objective
FDA3 Zheng	$f_1(x,t) = \frac{1}{\lvert\mathrm{x_I}\rvert} \sum_{x_i \in \mathrm{x_I}} x_i^{F(t)}$, $f_2(x,t) = g(\mathrm{x_{II}},t)h(f_1,g)$ $g(x_{\mathrm{II}},t) = 1 + G(t) + \sum_{x_i \in \mathrm{x_{II}}} (x_i - G(t))^2$, $h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}}$ $G(t) = \lvert sin(0.5\pi t)\rvert$, $F(t) = 10^{2\sin(0.5\pi t)}$	$\mathrm{x_I} = (x_1, \dots, x_{r_1})$ $\mathrm{x_I} \in [0,1]$ $\mathrm{x_{II}} = (x_{r_1+1}, \dots, x_n)$ $\mathrm{x_{II}} \in [-1,1]^{n-1}$	POS: $\{x \mid \forall x_i \in \mathrm{x_{II}}: x_i = G(t)\}$ POF: $\left(1 + G(t)\right) * \left(1 - \sqrt{f_1/(1 + G(t))}\,\right)$	type II Two-objective
FDA4	$f_1(x,t) = (1 + g(\mathrm{x_{II}},t)) \cos(0.5\pi x_1) \cos(0.5\pi x_2)$ $f_2(x,t) = (1 + g(\mathrm{x_{II}},t)) \cos(0.5\pi x_1) \sin(0.5\pi x_2)$ $f_3(x,t) = (1 + g(\mathrm{x_{II}},t)) \sin(0.5\pi x_1)$ $g(\mathrm{x_{II}},t) = \sum_{i=3}^{n} (x_i - G(t))^2$, $G(t) = \lvert\sin(0.5\pi t)\rvert$	$\mathrm{x_I} = (x_1, x_2)$ $\mathrm{x_{II}} = (x_3, \dots, x_n)$ $\mathrm{x}_i \in [0,1]^n$	POS: $\{x \mid 0 \le x_1, x_2 \le 1$, $x_i = G(t)$, for $i = 3, \dots, n\}$ POF: $f_1 + f_2 + f_3 = 1$	type I Three-objective
dMOP1	$f_1(\mathrm{x_I}) = x_1$, $f_2(\mathrm{x_{II}},t) = g(\mathrm{x_{II}})h(f_1,g,t)$ $g(\mathrm{x_{II}}) = 1 + 9\sum_{x_i \in \mathrm{x_{II}}} (x_i)^2$ $h(f_1,g,t) = 1 - \left(\frac{f_1}{g}\right)^{H(t)}$, $H(t) = 0.75\sin(0.5\pi t)$+1.25	$\mathrm{x_I} = (x_1)$ $\mathrm{x_{II}} = (x_2, \dots, x_n)$ $x_i \in [0,1]$	POS: $\{x \mid \forall x_i \in \mathrm{x_{II}}: x_i = 0\}$ POF: $f_2 = 1 - f_1^{H(t)}, 0 \le f_1 \le 1$	type III Two-objective
dMOP2	$f_1(\mathrm{x_I}) = x_1$, $f_2(\mathrm{x_{II}},t) = g(\mathrm{x_{II}},t)h(f_1,g,t)$ $g(\mathrm{x_{II}},t) = 1 + 9\sum_{x_i \in \mathrm{x_{II}}} (x_i - G(t))^2$ $h(f_1,g,t) = 1 - \left(\frac{f_1}{g}\right)^{H(t)}$ $H(t) = 0.75\sin(0.5\pi t)$+1.25 , $G(t) = \sin(0.5\pi t)$	$\mathrm{x_I} = (x_1)$ $\mathrm{x_{II}} = (x_2, \dots, x_n)$ $x_i \in [0,1]$	POS: $\{x \mid \forall x_i \in \mathrm{x_{II}}: x_i = G(t)\}$ POF: $f_2 = 1 - f_1^{H(t)}, 0 \le f_1 \le 1$	type II Two-objective
dMOP3	$f_1(\mathrm{x_I},t) = x_{r(t)}$, $f_2(f_1,g,t) = g(\mathrm{x_{II}},t)h(f_1,g)$ $g(\mathrm{x_{II}},t) = 1 + 9\sum_{x_i \in x_{\mathrm{II}}} (x_i - G(t))^2$ $h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}}$, $G(t) = \sin(0.5\pi t)$, $r(t) = \mathrm{U}(1,2, \dots, n)$	$\mathrm{x_I} = \left(x_{r(t)}\right)$ $\mathrm{x_{II}} = x \backslash x_{r(t)}$ $x_i \in [0,1]$	POS: $\{x \mid \forall x_i \in \mathrm{X}$, $i \ne r(t) : x_i = G(t)\}$ POF: $f_2 = 1 - \sqrt{f_1}, 0 \le f_1 \le 1$	type I Two-objective
F5	$f_1(x,t) = \lvert x_1 - a\rvert^{H(t)} + \sum_{i \in I_1} y_i^2$ $f_2(x,t) = \lvert x_1 - a - 1\rvert^{H(t)} + \sum_{i \in I_2} y_i^2$ $y_i = x_i - b - 1 + \lvert x_1 - a\rvert^{H(t)+\frac{i}{n}}$, $H(t) = 1.25 + 0.75\sin(\pi t/n_t)$ $a = 2\cos(\pi t/n_t) + 2$, $b = 2\sin(2\pi t/n_t) + 2$	$x_i \in [0,5]^n$	POS: $\{x \mid a \le x_1 \le a + 1$, $x_i = b + 1 - \lvert x_1 - a\rvert^{H(t)+i/n}$, for $i = 2, \dots, n\}$ POF: $f_1 = s^{H(t)}$, $f_2 = (1 - s)^{H(t)}$, $0 \le s \le 1$	type II Two-objective nonlinear

	$I_1 = \{i \mid 1 \le i \le n, i \text{ is odd}\}$, $I_2 = \{i \mid 1 \le i \le n, i \text{ is even}\}$			
F6	$f_1(x,t) = \lvert x_1 - a\rvert^{H(t)} + \sum_{i\in I_1} y_i^2$ $f_2(x,t) = \lvert x_1 - a - 1\rvert^{H(t)} + \sum_{i\in I_2} y_i^2$ $y_i = x_i - b - 1 + \lvert x_1 - a\rvert^{H(t)+\frac{i}{n}}$, $H(t) = 1.25 + 0.75\sin(\pi t/n_t)$ $a = 2\cos(1.5\pi t/n_t)\sin(0.5\pi t/n_t) + 2$ $b = 2\cos(1.5\pi t/n_t)\cos(0.5\pi t/n_t) + 2$ $I_1 = \{i \mid 1 \le i \le n, i \text{ is odd}\}$, $I_2 = \{i \mid 1 \le i \le n, i \text{ is even}\}$	$x_i \in [0,5]^n$	POS: $\{x \mid a \le x_1 \le a+1$, $x_i = b + 1 - \lvert x_1 - a\rvert^{H(t)+i/n}$, for $i = 2,\dots,n\}$ POF: $f_1 = s^{H(t)}$, $f_2 = (1-s)^{H(t)}$, $0 \le s \le 1$	type II Two-objective nonlinear
F8	$f_1(x,t) = (1 + g(x,t))\cos(0.5\pi x_2)\cos(0.5\pi x_1)$ $f_2(x,t) = (1 + g(x,t))\cos(0.5\pi x_2)\sin(0.5\pi x_1)$ $f_3(x,t) = (1 + g(x,t))\sin(0.5\pi x_2)$ $g(x,t) = \sum_{i=3}^{n}(x_i - \left(\frac{x_1+x_2}{2}\right)^{H(t)} - G(t))^2$ $H(t) = 1.25 + 0.75\sin(\pi t/n_t)$, $G(t) = \sin(0.5\pi t/n_t)$	$\mathrm{x_I} = (x_1, x_2)$ $\mathrm{x_I} \in [0,1]^2$ $\mathrm{x_{II}} = (x_3,\dots,x_n)$ $\mathrm{x_{II}} \in [-1,1]^{n-2}$	POS: $\{x \mid 0 \le x_1, x_2 \le 1$, $x_i = (\frac{x_1+x_2}{2})^{H(t)} + G(t)$, for $i = 3,\dots,n\}$ POF: $f_1 + f_2 + f_3 = 1$	type I Three-objective nonlinear
ZJZ	$f_1(x) = x_1$, $f_2(x,t) = 1 - (f_1/g)^{H(t)}$ $g(x,t) = 1 + \sum_{i=2}^{n}\left(x_i + G(t) - x_1^{H(t)}\right)^2$ $H(t) = 1.5 + G(t)$, $G(t) = sin(0.5\pi t)$	$\mathrm{x_I} = (x_1)$ $\mathrm{x_I} \in [0,1]$ $\mathrm{x_{II}} = (x_2,\dots,x_n)$ $\mathrm{x_{II}} \in [-1,2]^{\mathrm{n}-1}$	POS: $\{x \mid \forall x_i \in \mathrm{x_{II}}: x_i = G(t) + x_1^{H(t)}\}$ POF: $f_2 = 1 - f_1^{H(t)}$, $0 \le f_1 \le 1$	type II Two-objective nonlinear
HE3	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - x_1^{0.5\left(1+\frac{3(j-2)}{n-2}\right)}\right)^2$ $f_2(x,t) = g(x)h(f_1,g,t)$ $g(x) = 2 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{i\in J_2}\left(x_j - x_1^{0.5\left(1+\frac{3(j-2)}{n-2}\right)}\right)^2$ $h(f_1,g,t) = \left\lvert 1 - (f_1/g)^{H(t)}\right\rvert$, $H(t) = 0.75\sin(0.5\pi t) + 1.25$ $J_1 = \{j \mid 2 \le j \le n, j \text{ is odd}\}$, $J_2 = \{j \mid 2 \le j \le n, j \text{ is even}\}$	$x_j \in [0,1]$	POS: $\left\{x \mid x_j = x_1^{0.5\left(\frac{3(j-2)}{n-2}\right)}, \text{for } j = 2,3,\dots,n\right\}$ POF: $y = \left(2 - \sqrt{x_1}\right)\left[1 - \left(\frac{x_1}{2-\sqrt{x_1}}\right)^{H(t)}\right]$	type III Two-objective nonlinear
HE4	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - \sin(6\pi x_1 + j\pi/n)\right)^2$ $f_2(x,t) = g(x)h(f_1,g,t)$ $g(x) = 2 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{i\in J_2}\left(x_j - \sin(6\pi x_1 + j\pi/n)\right)^2$ $h(f_1,g,t) = \left\lvert 1 - (f_1/g)^{H(t)}\right\rvert$, $H(t) = 0.75\sin(0.5\pi t) + 1.25$ $J_1 = \{j \mid 2 \le j \le n, j \text{ is odd}\}$, $J_2 = \{j \mid 2 \le j \le n, j \text{ is even}\}$	$\mathrm{x_I} = (x_1)$ $\mathrm{x_I} \in [0,1]$ $\mathrm{x_{II}} = (x_2,\dots,x_n)$ $\mathrm{x_{II}} \in [-1,1]^{n-1}$	POS: $\{x \mid x_j = \sin(6\pi x_1 + j\pi/n)$, for $j = 2,3,\dots,n\}$ POF: $y = (2 - \sqrt{x_1})\left[1 - \left(\frac{x_1}{2-\sqrt{x_1}}\right)^{H(t)}\right]$	type III Two-objective nonlinear
HE5	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - 0.8x_1\cos(6\pi x_1 + j\pi/n)\right)^2$ $f_2(x,t) = g(x)h(f_1,g,t)$ $g(x) = 2 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - 0.8\cos(6\pi x_1 + j\pi/n)\right)^2$ $h(f_1,g,t) = \left\lvert 1 - (f_1/g)^{H(t)}\right\rvert$, $H(t) = 0.75\sin(0.5\pi t) + 1.25$ $J_1 = \{j \mid 2 \le j \le n, j \text{ is odd}\}$, $J_2 = \{j \mid 2 \le j \le n, j \text{ is even}\}$	$\mathrm{x_I} = (x_1)$ $\mathrm{x_I} \in [0,1]$ $\mathrm{x_{II}} = (x_2,\dots,x_n)$ $\mathrm{x_{II}} \in [-1,1]^{n-1}$	POS: $\{ x \mid$ $x_j = 0.8x_1\cos(6\pi x_1 + j\pi/n)$, for $j \in J_1 \wedge$ $x_j = 0.8x_1\sin(6\pi x_1 + j\pi/n)$, for $j \in J_2 \}$ POF: $y = \left(2 - \sqrt{x_1}\right)\left[1 - \left(\frac{x_1}{2-\sqrt{x_1}}\right)^{H(t)}\right]$	type III Two-objective nonlinear

6.1.2. Performance Measures

Several measures have been proposed for evaluating DMOO algorithms. Most of them only describe one particular aspect of performance, but some are comprehensive measures. Inverted generational distance (IGD) [64] and hypervolume ratio (HVR) [65] are the comprehensive measures that have attracted the most attention from researchers. Therefore, these measures are used in this paper to describe the performance of the algorithms.

IGD is a performance measure of DMOO algorithms that simultaneously describes the quality of the solution set with respect to convergence, diversity, and spread. IGD is the average of the

Euclidean distances between the sampled points of the POF and the nearest solution to them in the solution set suggested by the algorithm. This measure is calculated using the following equation:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n_{POF'}} d_i^2}}{n_{POF'}} \tag{17}$$

In the above equation, $n_{POF'}$ refers to the number of sampled points, and d_i refers to the Euclidean distance between the *i*th sampled point of the *POF* and the nearest solution to it among the suggested solutions of the algorithm. A lower IGD value indicates a higher quality for the solution set.

The hypervolume (HV) measure [5], [66] is a comprehensive measure that indicates what volume of the objective space is dominated by the suggested solutions of the algorithm. For a solution set A, the value of the HV measure with respect to a reference point far away from the POF is the volume of an area in the objective space that is dominated by solutions of A and is limited by the reference point. Its mathematical descriptions are as follows:

$$R(f_i, f_{\text{ref}}) \triangleq \{f_b | f_b \prec f_{\text{ref}} \text{ and } f_i \prec f_b, f_b \in \mathbb{R}^{n_k}\} \tag{18}$$

$$R(A, f_{\text{ref}}) \triangleq \cup_{i=1,\dots,|A|} R(f_i, f_{\text{ref}}) \tag{19}$$

The HV value is the hyper-area or Lebesgue integral of the set $R(A, f_{\text{ref}})$. A greater HV value means that the quality of the solution set is higher with respect to convergence, diversity and spread. The HVR measure is the normalized version of HV, and is calculated as follows:

$$HVR = \frac{HV(POF^*)}{HV(POF)} \tag{20}$$

In the above equation, *POF* indicates the set of sampled points from the Pareto optimal front and *POF** indicates the solution set suggested by the algorithm.

6.1.3. Experimental Settings

In this study, we compare the performance of the proposed algorithms with four well-known algorithms taken from the literature. These algorithms are:

- A GA-based algorithm (DNSGA-II A) [21].
- A GA-based algorithm (DNSGA-II B) [21].
- A clonal selection algorithm (CSADMO) [24].
- A cooperative coevolutionary algorithm with a predictive model (PNSCCDMO) [10].

The reason why the above algorithms have been selected for comparison with the proposed methods can be summarized as follows. DNSGA-II algorithms, which use the well-known NSGA-II algorithm, have attracted the attention of many researchers for similar comparison purposes. CSADMO is selected due to the remarkable results of algorithms inspired by clonal selection theory and PNSCCDMO is representative of algorithms that use prediction-based approaches to track environmental changes.

To compare the algorithms fairly, some considerations are defined for setting the parameters, as described below. Clearly, these considerations mean that some of the parameter settings do not match the settings in the reference papers, but this has a negligible impact on algorithm performance, and in most cases the effects are positive. The parameter settings for the compared algorithms are presented in Table 2.

Table 2: Parameter settings for the algorithms

Algorithm	Parameters
DNSGA-II A	Population size: N = 30 ; SBX probability: $p_\mathrm{c} = 0.9$; PM probability: $p_\mathrm{m} = 1/D$ Distribution indexes: $\eta_\mathrm{c} = 10$, $\eta_\mathrm{m} = 20$; Number of randomly replaced solutions: $\zeta = 20\%$
DNSGA-II B	Population size: N = 30 ; SBX probability: $p_\mathrm{c} = 0.9$; PM probability: $p_\mathrm{m} = 1/D$; Distribution indexes: $\eta_\mathrm{c} = 10$, $\eta_\mathrm{m} = 20$; Number of mutated solutions: $\zeta = 20\%$
CSADMO	Population size: N = 30 ; Clone proportion ≅ N/*Current_pop_size*
PNSCCDMO	subPopulation size: N = $30/n_\mathrm{obj}$; Number of subswarm: M = n_obj ; individuals generated by prediction = 80% Crossover probability: $p_c = 1$; Mutation probability: $p_\mathrm{m} = 1/D$; Distribution indexes: $\eta_\mathrm{c} = 10$, $\eta_\mathrm{m} = 20$
CTLBO	Population size: N = 30 ; Number of class: N_c = 3 ; Number of iterations in class: Itr_class = 1 ; Distribution index for mutation: $\eta_\mathrm{m} = 20$; Nelder-Mead iterations: Itr_NM = 50 ; Number of students: $N_\mathrm{s} = 4$
VECTLBO	subPopulation size: N = $30/n_\mathrm{obj}$; Number of class: N_c = 1 per subswarm ; Number of students: $N_\mathrm{s} = 6$ Number of iterations in class: Itr_class= 3; Distribution index for mutation: $\eta_\mathrm{m} = 20$; Nelder-Mead iterations: Itr_NM= 50 ; Exchanges interval = 6 ;

In the definition of the benchmark problems for dynamic optimization, the frequency of environmental changes was typically determined based on the algorithm iterations (generations); so, after a predefined number of iterations, the environment will be changed. Because each algorithm may perform a different number of solution evaluations in each iteration, many researchers have recently decided to use the "number of function evaluations", instead of "number of iterations", in determining the frequency of changes, to prevent potential unfairness. This decision has also been taken in the field of DMOO. Hence, in this research, the parameter of change frequency (τ_t) is characterized by the number of objective function evaluations. On the other hand, because the increase in iterations causes an increase in the steps of knowledge improvement that may affect performance, the algorithms are configured to have an average of 30 evaluations in each iteration.

Generally, a higher population size increases the exploration ability of the algorithm, but it also delays convergence. In this research, the population number is set to 30 for all algorithms.

In designing experiments for each problem, different combinations of values are considered for "severity of change", "frequency of change" and "problem dimensions" to evaluate the algorithms in different problem states and to investigate the effects of feature changes on algorithm performance. Because problems are of differing complexity, the values for the above features are specified for each problem according to its complexity.

In this research, each algorithm is evaluated 10 times for 50 consecutive environments in each problem, and the results of the experiments are compared using Friedman statistical analysis [67]. The rankings given in the tables are obtained from this analysis. The Friedman test is a nonparametric statistical test, and is used to check if there are significant differences in the distributions of several sets of data (results of different algorithms) by comparing the medians of the distributions.

6.2. Experimental Results

6.2.1. Evaluating and Comparing the Performance of Algorithms

6.2.1.1. Evaluation and Analysis of Algorithms for FDA1 Problem

In this section, the performances of the algorithms are compared for the FDA1 problem, which is a type I problem. Table 3 presents the experimental results for FDA1.

Table 3: Performance of the algorithms for the FDA1 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 15$ $n_t = 10$ $\tau_t = 500$	IGD	mean	2.09e-3	8.00e-4	1.57e-3	6.35e-4	**5.41e-4**	1.09e-3
			std. r.	3.0e-4 (6.0)	6.4e-5 (3.0)	1.0e-4 (5.0)	4.3e-5 (1.9)	**7.3e-5 (1.1)**	1.3e-4 (4.0)
		HVR	mean	64.82%	81.11%	63.64%	85.03%	**86.70%**	75.19%
			std. r.	3.15% (5.1)	1.50% (3.0)	1.70% (5.9)	0.75% (1.9)	**1.63% (1.1)**	2.30% (4.0)
2	$n = 15$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	5.20e-4	2.71e-4	7.75e-4	3.05e-4	**2.29e-4**	3.23e-4
			std. r.	1.9e-4 (4.9)	2.8e-5 (2.2)	3.0e-5 (5.8)	2.1e-5 (3.1)	**9.7e-6 (1.0)**	2.0e-5 (4.0)
		HVR	mean	88.29%	93.55%	80.44%	93.06%	**94.85%**	92.32%
			std. r.	3.45% (4.8)	0.26% (2.1)	0.63% (6.0)	0.40% (3.0)	**0.21% (1.0)**	0.46% (4.1)
3	$n = 15$	IGD	mean	2.52e-4	1.88e-4	3.84e-4	2.57e-4	**1.79e-4**	1.92e-4
			std. r.	5.4e-5 (4.2)	2.7e-6 (2.1)	5.6e-6 (6.0)	1.0e-5 (4.8)	**1.0e-6 (1.0)**	1.8e-6 (2.9)

	$n_t = 10$	HVR	mean	94.61%	95.99%	90.28%	95.19%	**96.61%**	96.33%
	$\tau_t = 2000$		std. r.	1.22% (4.6)	0.10% (3.0)	0.11% (6.0)	0.12% (4.4)	**0.03% (1.0)**	0.04% (2.0)
4	$n = 15$	IGD	mean	9.53e-3	3.11e-2	6.94e-3	**2.77e-3**	4.34e-3	4.74e-3
	$n_t = 2$		std. r.	3.8e-4 (5.0)	1.8e-3 (6.0)	5.2e-4 (4.0)	**1.1e-4 (1.0)**	6.3e-4 (2.4)	5.0e-4 (2.6)
	$\tau_t = 500$	HVR	mean	15.88%	5.54%	21.53%	**48.06%**	44.68%	42.76%
			std. r.	2.32% (5.0)	2.15% (6.0)	0.79% (4.0)	**0.94% (1.2)**	3.64% (2.3)	2.68% (2.6)
5	$n = 15$	IGD	mean	4.13e-3	1.32e-2	1.76e-3	**1.35e-3**	2.05e-3	1.98e-3
	$n_t = 2$		std. r.	7.8e-4 (5.0)	1.3e-3 (6.0)	9.6e-5 (2.5)	**1.1e-4 (1.0)**	2.9e-4 (3.5)	4.4e-4 (3.0)
	$\tau_t = 1000$	HVR	mean	40.34%	22.57%	59.77%	**70.60%**	68.07%	67.83%
			std. r.	5.43% (5.0)	1.63% (6.0)	1.72% (3.9)	**1.97% (1.4)**	3.89% (2.4)	2.77% (2.3)
6	$n = 15$	IGD	mean	1.57e-3	9.93e-4	6.17e-4	6.96e-4	**4.71e-4**	5.21e-4
	$n_t = 2$		std. r.	4.8e-4 (5.8)	2.1e-4 (5.0)	1.4e-5 (2.6)	2.8e-5 (4.0)	**9.2e-5 (1.6)**	1.1e-4 (2.0)
	$\tau_t = 2000$	HVR	mean	74.86%	80.40%	85.04%	85.02%	**90.29%**	88.75%
			std. r.	4.07% (5.9)	2.57% (5.0)	0.33% (3.6)	0.47% (3.5)	**1.44% (1.2)**	1.91% (1.8)
7	$n = 30$	IGD	mean	1.44e-2	9.18e-3	7.55e-3	3.51e-3	**2.72e-3**	3.50e-3
	$n_t = 10$		std. r.	3.0e-3 (6.0)	1.5e-3 (4.9)	4.3e-4 (4.1)	4.0e-4 (2.4)	**3.5e-4 (1.4)**	8.3e-4 (2.1)
	$\tau_t = 500$	HVR	mean	12.13%	23.74%	21.89%	48.45%	51.86%	**52.59%**
			std. r.	5.00% (6.0)	2.38% (4.1)	1.22% (4.9)	2.05% (2.7)	4.43% (1.9)	**1.93% (1.4)**
8	$n = 30$	IGD	mean	6.47e-3	9.52e-4	2.94e-3	9.15e-4	**6.49e-4**	9.41e-4
	$n_t = 10$		std. r.	1.8e-3 (6.0)	1.8e-4 (2.9)	1.1e-4 (5.0)	1.4e-4 (2.9)	**4.8e-5 (1.0)**	1.0e-4 (3.2)
	$\tau_t = 1000$	HVR	mean	45.66%	79.57%	44.86%	80.38%	**83.99%**	78.24%
			std. r.	10.27% (5.4)	2.28% (2.7)	1.45% (5.6)	1.59% (2.5)	**1.05% (1.0)**	1.69% (3.8)
9	$n = 30$	IGD	mean	1.32e-3	3.77e-4	1.27e-3	3.74e-4	**2.87e-4**	3.13e-4
	$n_t = 10$		std. r.	5.0e-4 (5.5)	6.2e-5 (3.5)	3.4e-5 (5.5)	2.7e-5 (3.5)	**2.4e-5 (1.3)**	1.9e-5 (1.7)
	$\tau_t = 2000$	HVR	mean	76.30%	91.42%	69.90%	91.52%	**92.93%**	92.24%
			std. r.	5.55% (5.1)	0.58% (3.4)	0.53% (5.9)	0.31% (3.5)	**0.51% (1.2)**	0.49% (1.9)
10	$n = 30$	IGD	mean	3.87e-2	8.92e-2	5.39e-2	**9.23e-3**	1.35e-2	1.94e-2
	$n_t = 2$		std. r.	3.3e-3 (4.0)	3.2e-3 (6.0)	1.9e-3 (5.0)	**4.8e-4 (1.0)**	4.1e-3 (2.1)	2.4e-3 (2.9)
	$\tau_t = 500$	HVR	mean	0.06%	0.13%	0.88%	**22.58%**	10.45%	19.29%
			std. r.	0.15% (5.7)	0.15% (5.3)	0.54% (4.0)	**0.88% (1.0)**	3.64% (3.0)	1.63% (2.0)
11	$n = 30$	IGD	mean	2.36e-2	7.33e-2	1.76e-2	**5.24e-3**	8.73e-3	1.09e-2
	$n_t = 2$		std. r.	3.0e-3 (5.0)	2.7e-3 (6.0)	1.0e-3 (4.0)	**3.7e-4 (1.0)**	8.4e-4 (2.1)	2.3e-3 (2.9)
	$\tau_t = 1000$	HVR	mean	3.49%	0.88%	11.35%	30.58%	26.85%	**45.39%**
			std. r.	1.84% (5.0)	0.46% (6.0)	0.77% (4.0)	0.78% (2.1)	2.86% (2.9)	**2.86% (1.0)**
12	$n = 30$	IGD	mean	7.65e-3	2.20e-2	4.04e-3	**2.64e-3**	3.92e-3	3.01e-3
	$n_t = 2$		std. r.	7.7e-4 (5.0)	4.5e-3 (6.0)	1.4e-4 (3.7)	**1.3e-4 (1.2)**	3.8e-4 (3.3)	5.2e-4 (1.8)
	$\tau_t = 2000$	HVR	mean	16.51%	15.98%	28.36%	51.75%	52.49%	**72.21%**
			std. r.	3.53% (5.5)	2.28% (5.5)	0.84% (4.0)	1.81% (2.6)	1.81% (2.4)	**3.05% (1.0)**

As shown in Table 3, for experiments on FDA1, the CTLBO, VECTLBO and PNSCCDMO algorithms are the most successful in achieving first rank. According to the results obtained in experiments 1, 2, 3, 7, 8 and 9 in comparison with the results of the other experiments, it can be concluded that the CTLBO algorithm operates more successfully when the severity of change is lower. The VECTLBO algorithm is ranked first when the dimension number is greater and the change severity is higher (experiments 6, 7 and 8). In experiments with high severity of changes, PNSCCDMO was more successful and approached the first rank in cases where the frequency of change was higher, but by decreasing the frequency, CTLBO and VECTLBO both obtained results closer to those of PNSCCDMO, and surpassed the latter algorithm in some cases. By analyzing the presented results, it appears that in solving the FDA1 problem, the VECTLBO algorithm has good potential for leading the solutions to the POF and spreading them uniformly. Thus, if the frequency of changes is lower, this algorithm performs better compared to other algorithms.

6.2.1.2. Evaluation and Analysis of Algorithms for FDA2$_{\text{Camara}}$ Problem

FDA2$_{\text{Camara}}$ is a type II problem and therefore, its POS and POF change over time. Table 4 shows the evaluation findings for the FDA2$_{\text{Camara}}$ problem.

Table 4: Performance of the algorithms for FDA2$_{\text{Camara}}$ problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 15$ $n_t = 10$ $\tau_t = 500$	IGD	mean	4.91e-4	**4.82e-4**	4.62e-4	1.02e-3	7.03e-4	8.18e-4
			std. r.	1.6e-4 (2.1)	**1.5e-4 (1.9)**	5.1e-5 (2.6)	2.1e-4 (5.7)	2.0e-5 (3.7)	2.3e-5 (5.0)
		HVR	mean	91.33%	**91.18%**	89.50%	78.38%	90.59%	88.91%
			std. r.	2.29% (2.1)	**2.17% (1.9)**	0.56% (4.0)	3.85% (6.0)	0.52% (2.6)	0.90% (4.4)
2	$n = 15$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	**2.52e-4**	2.96e-4	3.24e-4	5.55e-4	6.29e-4	7.18e-4
			std. r.	**2.1e-5 (1.3)**	4.7e-5 (1.9)	5.0e-5 (2.8)	1.3e-4 (4.4)	1.9e-5 (4.8)	2.2e-5 (5.8)
		HVR	mean	**95.30%**	94.66%	93.71%	91.09%	93.83%	92.95%
			std. r.	**0.17% (1.0)**	0.75% (2.3)	0.29% (3.6)	2.09% (5.9)	0.16% (3.2)	0.44% (5.0)
3	$n = 15$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	**2.40e-4**	2.17e-4	2.71e-4	3.52e-4	6.21e-4	6.86e-4
			std. r.	**9.7e-5 (1.5)**	1.7e-5 (1.9)	1.9e-5 (2.8)	1.9e-5 (3.9)	1.2e-5 (5.0)	1.0e-5 (6.0)
		HVR	mean	95.20%	**95.57%**	95.29%	94.25%	94.74%	94.58%
			std. r.	1.60% (1.9)	**0.23% (1.8)**	0.11% (2.8)	0.19% (5.8)	0.22% (4.0)	0.08% (4.9)
4	$n = 15$ $n_t = 2$ $\tau_t = 500$	IGD	mean	1.68e-3	8.48e-4	9.51e-4	1.74e-3	**7.00e-4**	8.43e-4
			std. r.	1.9e-3 (4.0)	1.5e-4 (3.0)	5.1e-5 (4.3)	5.8e-4 (5.9)	**5.3e-5 (1.1)**	4.2e-5 (2.7)
		HVR	mean	70.67%	81.10%	75.29%	61.75%	**83.66%**	82.06%
			std. r.	21.00% (3.9)	2.59% (2.6)	0.83% (4.7)	8.88% (5.9)	**2.32% (1.7)**	1.67% (2.3)
5	$n = 15$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	1.14e-3	7.85e-4	**5.09e-4**	8.95e-4	5.86e-4	6.76e-4
			std. r.	4.1e-4 (5.4)	2.2e-4 (4.0)	**3.5e-5 (1.0)**	1.2e-4 (4.9)	3.6e-5 (2.3)	2.4e-5 (3.4)
		HVR	mean	76.02%	83.48%	87.47%	81.36%	88.78%	**88.76%**
			std. r.	10.04% (5.3)	4.86% (4.0)	0.47% (3.2)	2.32% (5.0)	1.44% (1.9)	**0.81% (1.6)**
6	$n = 15$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	1.17e-3	7.49e-4	**3.30e-4**	5.67e-4	5.04e-4	5.88e-4
			std. r.	3.8e-4 (5.4)	1.8e-4 (4.8)	**2.4e-5 (1.0)**	7.6e-5 (3.5)	3.0e-5 (2.3)	1.5e-5 (4.0)
		HVR	mean	75.05%	84.22%	93.20%	88.45%	**93.68%**	93.16%
			std. r.	8.69% (5.6)	3.97% (5.0)	0.16% (2.3)	1.70% (4.4)	**0.75% (1.3)**	0.31% (2.4)
7	$n = 31$ $n_t = 10$ $\tau_t = 500$	IGD	mean	5.06e-3	1.89e-3	1.64e-3	1.06e-2	**1.07e-3**	1.28e-3
			std. r.	2.6e-3 (4.9)	2.8e-4 (3.9)	1.6e-4 (3.1)	2.7e-4 (6.0)	**9.5e-5 (1.1)**	1.5e-4 (2.0)
		HVR	mean	38.13%	67.34%	69.03%	0.02%	79.25%	**78.23%**
			std. r.	24.38% (4.7)	5.86% (3.9)	1.60% (3.4)	0.05% (6.0)	1.39% (1.6)	**3.53% (1.4)**
8	$n = 31$ $n_t = 10$	IGD	mean	2.75e-3	1.13e-3	9.38e-4	3.91e-3	**8.17e-4**	9.50e-4
			std. r.	2.3e-3 (4.8)	3.3e-4 (3.4)	6.0e-5 (2.8)	2.5e-3 (5.8)	**5.8e-5 (1.3)**	5.7e-5 (2.9)
		HVR	mean	62.69%	82.14%	83.88%	38.71%	**89.90%**	87.51%

	$\tau_t = 1000$		std. r.	23.99% (4.8)	7.10% (3.4)	0.75% (3.7)	25.69% (5.9)	**0.80% (1.1)**	1.97% (2.1)
9	$n = 31$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	5.12e-3	7.99e-4	**6.15e-4**	1.36e-3	7.16e-4	8.34e-4
			std. r.	3.0e-3 (5.2)	1.5e-4 (3.4)	**6.3e-5 (1.1)**	4.8e-4 (5.0)	3.6e-5 (2.4)	4.1e-5 (3.9)
		HVR	mean	40.33%	88.52%	91.77%	77.60%	**93.20%**	92.22%
			std. r.	33.75% (5.5)	3.39% (4.4)	0.38% (2.8)	8.75% (5.1)	**0.16% (1.0)**	0.98% (2.2)
10	$n = 31$ $n_t = 2$ $\tau_t = 500$	IGD	mean	4.71e-3	2.09e-3	2.57e-3	1.06e-2	**1.07e-3**	1.32e-3
			std. r.	2.2e-3 (5.0)	3.9e-4 (3.1)	1.5e-4 (3.9)	2.1e-4 (6.0)	**4.6e-5 (1.0)**	4.4e-5 (2.0)
		HVR	mean	37.15%	56.47%	44.82%	0.01%	**73.74%**	71.63%
			std. r.	21.17% (4.1)	6.40% (3.3)	1.67% (4.6)	0.02% (6.0)	**0.94% (1.1)**	1.01% (1.9)
11	$n = 31$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	3.98e-3	1.37e-3	6.15e-4	6.70e-3	**8.73e-4**	1.04e-3
			std. r.	2.5e-3 (4.9)	2.0e-4 (3.3)	6.3e-5 (3.9)	9.6e-4 (5.9)	**5.0e-5 (1.0)**	5.5e-5 (2.0)
		HVR	mean	42.54%	71.94%	62.96%	7.39%	**79.05%**	78.74%
			std. r.	24.42% (4.8)	3.83% (3.1)	2.21% (4.2)	6.28% (5.9)	**1.14% (1.3)**	1.07% (1.7)
12	$n = 31$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	5.88e-3	1.19e-3	9.44e-4	2.84e-3	**6.81e-4**	8.56e-4
			std. r.	2.2e-3 (5.8)	2.4e-4 (3.8)	4.2e-5 (3.2)	6.7e-4 (5.2)	**4.6e-5 (1.0)**	3.6e-5 (2.0)
		HVR	mean	24.42%	76.76%	79.60%	46.88%	**86.69%**	85.54%
			std. r.	20.67% (5.8)	5.47% (3.8)	0.74% (3.2)	10.37% (5.2)	**1.40% (1.3)**	1.00% (1.7)

It can be observed that the DNSGA-II A and DNSGA-II B algorithms lead the competition when the number of search space dimensions is smaller and the severity of change is lower (Experiments 1, 2 and 3); however, the VECTLBO, CSADMO, and CTLBO algorithms surpass them as the severity of change increases (Experiments 4, 5 and 6). When the number of dimensions is increased to 31, the CLTBO and VECTLBO algorithms maintain their efficiency, and although results are close, they are ranked first and second respectively. Additionally, the CSADMO algorithm achieved desirable efficiency in some cases (Experiments 5, 6 and 9) based on the IGD criterion, and can therefore be regarded as competitive with the above two algorithms.

6.2.1.3. Evaluation and Analysis of Algorithms for FDA3$_{\text{Zheng}}$ Problem

In this step, the performance of the algorithms in solving the FDA3$_{\text{Zheng}}$ problem is assessed. This is a type II problem in which both the POS and POF are subject to change. Table 5 shows the results of the experiments for this problem.

Table 5: Performance of the algorithms for FDA3$_{\text{Zheng}}$ problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 15$ $n_t = 10$ $\tau_t = 500$	IGD	mean	5.19e-3	2.64e-3	3.28e-3	1.39e-3	**7.60e-4**	9.67e-4
			std. r.	1.2e-3 (6.0)	8.0e-4 (4.1)	2.4e-4 (4.9)	2.0e-4 (3.0)	**4.3e-5 (1.0)**	1.2e-4 (2.0)
		HVR	mean	42.65%	69.09%	57.07%	79.76%	**85.97%**	83.03%
			std. r.	10.90% (6.0)	8.25% (4.0)	2.24% (4.9)	1.65% (3.1)	**0.64% (1.1)**	1.39% (1.9)

2	$n = 15$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	3.42e-3	1.52e-3	2.08e-3	8.90e-4	**4.36e-4**	7.34e-4
			std. r.	7.8e-4 (5.9)	4.2e-4 (3.9)	1.5e-4 (5.1)	1.2e-4 (2.7)	**5.4e-5 (1.0)**	1.7e-4 (2.4)
		HVR	mean	64.63%	85.85%	73.65%	88.70%	**93.04%**	89.08%
			std. r.	9.79% (6.0)	3.01% (4.1)	0.94% (4.8)	0.73% (3.0)	**1.07% (1.0)**	1.24% (2.1)
3	$n = 15$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	1.91e-3	9.56e-4	1.39e-3	7.81e-4	4.73e-4	**4.13e-4**
			std. r.	2.7e-4 (5.9)	3.2e-4 (3.9)	6.4e-5 (5.0)	9.0e-5 (3.3)	3.9e-5 (1.9)	**3.8e-5 (1.1)**
		HVR	mean	80.93%	91.47%	83.97%	91.46%	93.05%	**94.13%**
			std. r.	3.47% (5.8)	2.42% (2.9)	0.46% (5.3)	0.69% (3.8)	0.67% (2.1)	**0.37% (1.2)**
4	$n = 15$ $n_t = 2$ $\tau_t = 500$	IGD	mean	7.74e-3	8.60e-3	6.29e-3	5.18e-3	3.73e-3	**2.55e-3**
			std. r.	4.5e-4 (5.0)	4.8e-4 (6.0)	2.8e-4 (4.0)	6.3e-4 (3.0)	3.6e-4 (2.0)	**1.1e-4 (1.0)**
		HVR	mean	19.95%	41.04%	32.94%	36.27%	53.60%	**62.71%**
			std. r.	1.43% (6.0)	4.50% (3.3)	1.56% (4.7)	4.04% (4.0)	3.74% (2.0)	**1.59% (1.0)**
5	$n = 15$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	5.89e-3	5.49e-3	3.50e-3	3.79e-3	2.76e-3	**1.49e-3**
			std. r.	3.0e-4 (5.9)	6.1e-4 (5.1)	1.7e-4 (3.3)	4.0e-4 (3.7)	3.5e-4 (2.0)	**1.6e-4 (1.0)**
		HVR	mean	32.72%	48.13%	53.97%	52.07%	68.24%	**78.51%**
			std. r.	3.05% (6.0)	3.10% (4.8)	1.52% (3.3)	3.08% (3.9)	3.72% (2.0)	**1.73% (1.0)**
6	$n = 15$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	4.71e-3	3.27e-3	2.67e-3	2.70e-3	1.31e-3	**1.04e-3**
			std. r.	3.1e-4 (6.0)	3.3e-4 (5.0)	1.2e-4 (3.6)	2.7e-4 (3.4)	2.5e-4 (1.8)	**2.1e-4 (1.2)**
		HVR	mean	51.23%	64.95%	70.87%	68.65%	85.50%	**86.49%**
			std. r.	2.79% (6.0)	3.67% (4.9)	0.95% (3.2)	2.83% (3.9)	2.65% (1.6)	**1.64% (1.4)**
7	$n = 30$ $n_t = 10$ $\tau_t = 500$	IGD	mean	1.44e-2	1.07e-2	9.16e-3	7.34e-3	4.16e-3	**3.99e-3**
			std. r.	1.0e-3 (6.0)	1.1e-3 (5.0)	3.1e-4 (4.0)	6.2e-4 (3.0)	4.6e-4 (1.6)	**4.1e-4 (1.4)**
		HVR	mean	8.75%	19.91%	16.36%	38.96%	45.33%	**50.05%**
			std. r.	1.64% (6.0)	2.20% (4.0)	1.54% (5.0)	1.53% (2.9)	3.24% (2.1)	**2.24% (1.0)**
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	8.60e-3	4.00e-3	5.16e-3	3.25e-3	2.35e-3	**1.70e-3**
			std. r.	1.7e-3 (6.0)	6.0e-4 (4.0)	2.4e-4 (4.9)	4.8e-4 (3.0)	3.0e-4 (2.1)	**4.1e-4 (1.0)**
		HVR	mean	27.65%	54.02%	33.73%	58.39%	66.45%	**73.08%**
			std. r.	8.77% (5.8)	7.03% (3.5)	1.46% (5.2)	4.74% (3.3)	2.84% (2.2)	**3.69% (1.0)**
9	$n = 30$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	4.30e-3	2.22e-3	3.50e-3	1.53e-3	1.19e-3	**8.56e-4**
			std. r.	6.2e-4 (5.9)	4.1e-4 (4.0)	2.3e-4 (5.1)	2.4e-4 (2.8)	2.3e-4 (2.0)	**2.6e-4 (1.2)**
		HVR	mean	56.54%	77.45%	54.32%	80.56%	81.42%	**86.28%**
			std. r.	5.03% (5.3)	3.65% (3.4)	1.80% (5.7)	2.10% (2.9)	2.01% (2.7)	**1.52% (1.0)**
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	3.13e-2	2.68e-2	2.31e-2	1.72e-2	2.34e-2	**8.48e-3**
			std. r.	4.3e-3 (5.9)	1.5e-3 (5.0)	4.0e-4 (3.3)	1.9e-3 (2.0)	2.9e-3 (3.9)	**1.4e-3 (1.0)**
		HVR	mean	3.49%	19.93%	8.58%	6.33%	3.80%	**28.69%**
			std. r.	1.57% (5.6)	2.50% (2.0)	0.90% (3.1)	2.22% (4.1)	0.60% (5.1)	**1.92% (1.0)**
11	$n = 30$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	2.50e-2	1.99e-2	1.33e-2	1.34e-2	1.04e-2	**4.50e-3**
			std. r.	1.5e-3 (6.0)	9.3e-4 (5.0)	6.1e-4 (3.4)	1.5e-3 (3.6)	9.5e-4 (2.0)	**5.5e-4 (1.0)**
		HVR	mean	3.73%	24.83%	13.95%	13.15%	20.14%	**51.31%**
			std. r.	0.94% (6.0)	1.58% (2.2)	1.20% (4.1)	1.91% (4.9)	3.84% (2.8)	**1.55% (1.0)**
12	$n = 30$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	1.06e-2	9.82e-3	5.88e-3	8.66e-3	5.24e-3	**2.11e-3**
			std. r.	3.8e-4 (5.7)	7.7e-4 (5.0)	2.7e-4 (2.7)	1.1e-3 (4.3)	6.1e-4 (2.3)	**1.5e-4 (1.0)**
		HVR	mean	11.60%	34.34%	26.70%	23.78%	42.17%	**71.49%**
			std. r.	1.46% (6.0)	3.02% (2.8)	1.19% (4.0)	3.14% (4.9)	8.98% (2.3)	**1.43% (1.0)**

The results presented in Table 5 suggest that the VECTLBO algorithm surpassed the other algorithms in solving the FDA3$_{\text{Zheng}}$ problem in the majority of cases. Its main rival is the CTLBO algorithm, which ranked first in cases where the dimension number and the severity of changes were both low (Experiments 1 and 2).

6.2.1.4. Evaluation and Analysis of Algorithms for FDA4 Problem

FDA4 is another problem in the FDA suite which has three objectives and is of type I. The results obtained for experiments on this problem are presented in Table 6.

Table 6: Performance of the algorithms for FDA4 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 15$ $n_t = 10$ $\tau_t = 500$	IGD	mean	6.55e-4	1.34e-3	9.53e-4	6.41e-4	**3.86e-4**	5.42e-4
			std. r.	5.6e-5 (3.5)	2.5e-4 (6.0)	2.8e-5 (5.0)	3.1e-5 (3.5)	**3.5e-5 (1.0)**	6.6e-5 (2.0)
		HVR	mean	49.05%	29.08%	33.26%	52.24%	**75.55%**	61.08%
			std. r.	5.85% (3.8)	3.20% (5.9)	3.89% (5.1)	2.73% (3.1)	**1.85% (1.0)**	5.57% (2.1)
2	$n = 15$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	4.51e-4	5.40e-4	5.94e-4	4.57e-4	**2.16e-4**	2.95e-4
			std. r.	3.2e-5 (3.4)	5.3e-5 (5.0)	3.2e-5 (5.9)	1.9e-5 (3.8)	**2.4e-5 (1.0)**	2.5e-5 (2.0)
		HVR	mean	67.38%	62.25%	54.90%	71.54%	**88.81%**	81.71%
			std. r.	2.88% (4.1)	4.07% (4.6)	2.02% (6.0)	3.15% (3.2)	**1.97% (1.0)**	2.11% (2.0)
3	$n = 15$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	3.43e-4	2.78e-4	5.30e-4	3.51e-4	**1.31e-4**	1.53e-4
			std. r.	3.2e-5 (4.3)	1.1e-5 (3.0)	2.4e-5 (6.0)	7.7e-6 (4.7)	**2.8e-6 (1.0)**	9.5e-6 (2.0)
		HVR	mean	79.06%	85.10%	60.65%	84.25%	**94.65%**	93.51%
			std. r.	3.28% (5.0)	1.92% (3.3)	1.24% (6.0)	1.27% (3.7)	**0.40% (1.0)**	0.66% (2.0)
4	$n = 15$ $n_t = 2$ $\tau_t = 500$	IGD	mean	1.09e-3	2.72e-3	2.09e-3	1.06e-3	**6.59e-4**	1.16e-3
			std. r.	5.9e-5 (2.8)	2.7e-4 (6.0)	1.3e-4 (5.0)	4.3e-5 (2.8)	**2.9e-5 (1.0)**	1.6e-4 (3.5)
		HVR	mean	35.84%	22.66%	19.18%	27.62%	**60.02%**	34.27%
			std. r.	3.48% (2.3)	6.60% (5.0)	2.23% (5.6)	1.17% (4.3)	**3.31% (1.0)**	3.97% (2.9)
5	$n = 15$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	7.62e-4	2.41e-3	1.67e-3	8.26e-4	**2.96e-4**	5.89e-4
			std. r.	8.5e-5 (3.1)	2.1e-4 (6.0)	9.9e-5 (5.0)	3.5e-5 (3.9)	**6.7e-5 (1.0)**	3.6e-5 (2.0)
		HVR	mean	44.63%	33.21%	27.58%	40.13%	**83.57%**	59.67%
			std. r.	6.46% (3.4)	3.59% (4.9)	2.52% (5.9)	2.05% (3.9)	**2.25% (1.0)**	2.49% (2.0)
6	$n = 15$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	4.59e-4	1.71e-3	1.61e-3	5.56e-4	**1.74e-4**	2.39e-4
			std. r.	5.9e-5 (3.0)	1.5e-4 (5.9)	6.1e-5 (5.2)	4.8e-5 (4.0)	**2.4e-5 (1.0)**	1.9e-5 (2.0)
		HVR	mean	66.37%	48.73%	30.21%	61.36%	**93.05%**	86.83%
			std. r.	5.89% (3.2)	5.85% (5.0)	1.64% (6.0)	4.66% (3.9)	**1.13% (1.0)**	1.70% (2.0)
7	$n = 30$ $n_t = 10$ $\tau_t = 500$	IGD	mean	1.54e-3	4.26e-3	3.67e-3	1.28e-3	**8.92e-4**	1.80e-3
			std. r.	3.6e-4 (3.0)	9.9e-4 (5.9)	2.6e-4 (5.1)	6.1e-5 (2.1)	**1.3e-4 (1.0)**	2.9e-4 (3.9)
		HVR	mean	12.99%	6.51%	2.64%	31.74%	**44.56%**	16.55%
			std. r.	5.40% (4.0)	1.45% (4.8)	1.41% (6.0)	1.47% (2.0)	**3.64% (1.0)**	4.05% (3.2)
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	1.23e-3	2.22e-3	2.42e-3	8.23e-4	**5.25e-4**	8.70e-4
			std. r.	1.1e-4 (4.0)	3.5e-4 (5.4)	2.1e-4 (5.6)	4.7e-5 (2.2)	**6.9e-5 (1.0)**	6.5e-5 (2.7)
		HVR	mean	25.29%	21.45%	7.70%	43.14%	**63.61%**	45.96%
			std. r.	3.83% (4.1)	2.70% (4.9)	1.17% (6.0)	1.76% (2.9)	**3.39% (1.0)**	3.28% (2.1)
9	$n = 30$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	8.05e-4	6.06e-4	2.18e-3	5.20e-4	**2.85e-4**	3.99e-4
			std. r.	8.3e-5 (5.0)	5.7e-5 (3.9)	9.4e-5 (6.0)	3.1e-5 (3.1)	**3.2e-5 (1.0)**	6.3e-5 (2.0)
		HVR	mean	49.38%	57.56%	11.21%	64.53%	**82.88%**	72.90%
			std. r.	3.71% (4.9)	4.57% (4.1)	1.05% (6.0)	2.39% (3.0)	**3.51% (1.0)**	3.84% (2.0)
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	2.48e-3	6.25e-3	5.93e-3	2.01e-3	**1.24e-3**	2.58e-3
			std. r.	2.3e-4 (3.3)	5.8e-4 (5.8)	2.7e-4 (5.2)	8.2e-5 (2.0)	**1.3e-4 (1.0)**	2.8e-4 (3.8)
		HVR	mean	5.79%	5.68%	1.41%	14.06%	**26.74%**	14.54%
			std. r.	3.79% (4.6)	2.39% (4.5)	1.13% (5.9)	1.52% (2.4)	**4.46% (1.0)**	0.89% (2.6)
11	$n = 30$ $n_t = 2$	IGD	mean	1.80e-3	6.79e-3	5.10e-3	1.42e-3	**7.49e-4**	1.78e-3
			std. r.	1.9e-4 (3.6)	8.4e-4 (6.0)	9.9e-5 (5.0)	9.3e-5 (2.0)	**9.5e-5 (1.0)**	5.3e-5 (3.4)
		HVR	mean	7.04%	9.32%	5.51%	23.46%	**53.83%**	25.15%

	$\tau_t = 1000$		std. r.	2.34% (5.0)	4.78% (4.5)	1.93% (5.5)	1.85% (2.6)	**5.20% (1.0)**	2.46% (2.4)
12	$n = 30$	IGD	mean	1.24e-3	5.67e-3	4.89e-3	1.12e-3	**3.49e-4**	8.67e-4
	$n_t = 2$		std. r.	7.7e-5 (3.8)	3.4e-4 (6.0)	1.8e-4 (5.0)	7.0e-5 (3.0)	**2.5e-5 (1.0)**	1.9e-4 (2.2)
	$\tau_t = 2000$	HVR	mean	16.08%	17.82%	8.47%	32.32%	**78.21%**	47.51%
			std. r.	4.55% (4.5)	3.30% (4.5)	2.19% (6.0)	2.48% (3.0)	**1.36% (1.0)**	7.92% (2.0)

It can be concluded from Table 6 that the CTLBO algorithm outperforms the other algorithms significantly in solving the FDA4 problem. Among the other algorithms, VECTLBO ranks second in many cases, but by increasing the complexity of the problem, PNSCCDMO becomes more successful in preserving its efficiency and obtains results close to those of VECTLBO, surpassing it in some cases (Experiments 7 and 10).

6.2.1.5. Evaluation and Analysis of Algorithms for dMOP1 Problem

The dMOP1 problem is a type III problem in which only the POF changes. The results of experiments for this problem are presented in Table 7.

Table 7: Performance of the algorithms for dMOP1 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$	IGD	mean	1.30e-3	**3.07e-4**	4.41e-4	6.59e-4	5.37e-4	4.81e-4
	$n_t = 10$		std. r.	2.5e-2 (2.6)	**5.3e-5 (1.4)**	2.7e-5 (3.0)	1.8e-4 (5.0)	7.2e-5 (5.0)	2.5e-5 (4.0)
	$\tau_t = 500$	HVR	mean	80.40%	**93.01%**	87.86%	81.28%	86.37%	88.82%
			std. r.	31.29% (2.3)	**0.42% (1.3)**	0.61% (3.9)	2.53% (5.9)	1.63% (4.6)	0.69% (3.1)
2	$n = 10$	IGD	mean	1.57e-3	2.90e-4	4.09e-4	3.56e-4	2.12e-4	**2.01e-4**
	$n_t = 10$		std. r.	2.8e-3 (3.2)	5.7e-5 (3.6)	1.4e-5 (5.7)	7.8e-5 (4.4)	1.1e-5 (2.4)	**1.2e-5 (1.7)**
	$\tau_t = 1000$	HVR	mean	77.59%	93.78%	90.22%	90.29%	94.83%	**95.39%**
			std. r.	35.80% (3.7)	0.60% (3.5)	0.24% (5.5)	0.88% (5.1)	0.26% (2.2)	**0.20% (1.0)**
3	$n = 10$	IGD	mean	9.05e-4	2.37e-4	5.03e-4	2.75e-4	**1.77 e-4**	1.77e-4
	$n_t = 10$		std. r.	2.1e-3 (3.8)	5.9e-5 (3.7)	1.2e-5 (5.9)	6.1e-5 (4.1)	**5.9e-6 (1.6)**	3.7e-6 (1.9)
	$\tau_t = 2000$	HVR	mean	86.20%	94.68%	88.30%	93.10%	96.01%	**96.21%**
			std. r.	26.8% (3.7)	0.76% (3.5)	0.17% (5.9)	0.74% (4.9)	0.17% (1.9)	**0.06% (1.1)**
4	$n = 10$	IGD	mean	1.99e-3	**2.84e-4**	4.84e-4	8.13e-4	5.22e-4	4.57e-4
	$n_t = 2$		std. r.	2.9e-3 (3.0)	**6.3e-5 (1.1)**	6.6e-5 (3.7)	1.8e-4 (5.7)	3.9e-5 (4.3)	4.2e-5 (3.1)
	$\tau_t = 500$	HVR	mean	71.67%	**93.19%**	88.06%	79.40%	86.51%	89.41%
			std. r.	36.84% (3.0)	**0.64% (1.1)**	0.77% (3.9)	2.07% (5.7)	0.93% (4.6)	0.73% (2.7)
5	$n = 10$	IGD	mean	1.60e-3	2.46e-4	4.19e-4	3.62e-4	2.11e-4	**1.97e-4**
	$n_t = 2$		std. r.	2.8e-3 (3.9)	5.0e-5 (3.3)	1.7e-5 (5.6)	1.2e-4 (4.6)	1.2e-5 (2.5)	**8.0e-6 (1.1)**
	$\tau_t = 1000$	HVR	mean	77.25%	94.44%	90.18%	90.57%	94.91%	**95.49%**
			std. r.	35.81% (3.8)	0.60% (3.2)	0.26% (5.6)	1.57% (4.9)	0.34% (2.4)	**0.17% (1.1)**
6	$n = 10$	IGD	mean	1.53e-3	2.24e-4	5.08e-4	2.67e-4	**1.76e-4**	**1.72e-4**
	$n_t = 2$		std. r.	2.8e-3 (3.9)	6.3e-5 (3.4)	1.9e-5 (5.8)	6.0e-5 (4.5)	**9.4e-6 (1.7)**	**3.5e-6 (1.7)**
	$\tau_t = 2000$	HVR	mean	78.24%	94.93%	88.34%	93.39%	96.08%	**96.32%**
			std. r.	36.2% (4.0)	0.79% (3.5)	0.32% (5.8)	0.77% (4.7)	0.21% (2.0)	**0.06% (1.0)**
7	$n = 30$	IGD	mean	5.46e-3	2.56e-3	2.68e-3	4.97e-2	**2.57e-3**	2.62e-2
			std. r.	2.7e-3 (4.6)	4.8e-4 (3.0)	3.0e-4 (3.1)	2.7e-3 (6.0)	**7.3e-4 (1.9)**	1.0e-3 (2.4)

	$n_t = 10$ $\tau_t = 500$	HVR	mean	46.24%	**82.23%**	67.44%	0%	41.99%	60.19%
			std. r.	36.09% (3.3)	**1.33% (1.0)**	2.76% (2.6)	0% (6.0)	14.13% (4.6)	5.02% (3.6)
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	5.02e-3	9.47e-4	1.10e-3	7.41e-3	1.47e-3	**8.39e-4**
			std. r.	3.0e-3 (5.0)	2.1e-4 (2.3)	3.1e-4 (3.0)	5.9e-3 (5.6)	9.1e-4 (3.4)	**6.5e-4 (1.7)**
		HVR	mean	39.69%	**89.61%**	82.47%	44.66%	68.27%	86.25%
			std. r.	39.68% (4.6)	**0.88% (1.1)**	1.18% (3.4)	12.90% (5.3)	17.11% (4.3)	5.79% (2.3)
9	$n = 30$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	3.78e-3	4.13e-4	7.10e-4	1.03e-3	1.28e-3	**3.32e-4**
			std. r.	3.4e-3 (4.7)	4.4e-5 (2.4)	1.2e-4 (4.1)	3.3e-4 (5.0)	1.4e-3 (3.2)	**1.9e-4 (1.6)**
		HVR	mean	50.83%	92.91%	86.60%	81.33%	71.48%	**93.38%**
			std. r.	43.61% (4.4)	0.13% (1.9)	0.48% (4.2)	2.49% (5.2)	30.23% (4.0)	**1.87% (1.3)**
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	5.65e-3	2.37e-3	2.59e-3	5.10e-2	2.75e-3	**2.26e-3**
			std. r.	3.2e-3 (4.6)	4.8e-4 (2.4)	7.5e-4 (3.0)	2.5e-3 (6.0)	6.7e-4 (2.7)	**5.1e-4 (2.3)**
		HVR	mean	46.49%	**83.38%**	69.85%	0%	42.66%	62.44%
			std. r.	36.41% (3.4)	**0.82% (1.0)**	2.68% (2.4)	0% (6.0)	15.45% (4.6)	4.31% (3.6)
11	$n = 30$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	2.90e-3	**9.65e-4**	9.99e-4	3.70e-3	1.20e-3	**1.02e-3**
			std. r.	2.9e-3 (4.6)	**2.4e-4 (2.4)**	2.4e-4 (2.6)	1.4e-3 (5.8)	5.2e-4 (3.2)	**4.1e-4 (2.4)**
		HVR	mean	66.59%	**89.85%**	82.86%	53.70%	72.71%	86.03%
			std. r.	35.82% (3.5)	**0.46% (1.0)**	0.58% (3.8)	6.41% (5.8)	11.51% (4.8)	1.76% (2.2)
12	$n = 30$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	3.18e-3	3.92e-4	7.80e-4	9.16e-4	7.14e-4	**3.18e-4**
			std. r.	3.3e-3 (4.5)	4.2e-5 (2.3)	9.4e-5 (5.2)	2.0e-4 (4.8)	8.9 e-4 (2.9)	**1.3e-4 (1.3)**
		HVR	mean	58.28%	92.98%	86.56%	82.48%	83.79%	**93.40%**
			std. r.	42.27% (4.3)	0.09% (1.8)	0.34% (4.5)	1.61% (5.4)	19.95% (3.7)	**1.24% (1.3)**

As shown in Table 7, for the dMOP1 problem the main competition is between the VECTLBO, CTLBO and DNSGA-II B algorithms. In experiments with a low dimension number, CSADMO outperforms the other algorithms in cases where the frequency is high (Experiments 1 and 4), but as the frequency decreases, VECTLBO and CTLBO both surpass it and achieve first and second rank respectively. In cases with a high dimension number, VECTLBO and DSNAG-II B are the superior algorithms; each of them achieves first rank in some of the cases or according to one of the evaluation criteria.

6.2.1.6. Evaluation and Analysis of Algorithms for dMOP2 Problem

The efficiency of the algorithms is evaluated using the dMOP2 problem, which is a type II problem. The evaluation results are presented in Table 8.

Table 8: Performance of the algorithms for dMOP2 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$ $n_t = 10$ $\tau_t = 500$	IGD	mean	1.08e-2	**2.40e-3**	3.60e-3	2.48e-3	2.86e-3	2.62e-3
			std. r.	3.3e-3 (6.0)	**2.0e-4 (1.9)**	3.1e-4 (4.7)	1.0e-3 (2.7)	1.0e-4 (3.4)	2.5e-4 (2.3)
		HVR	mean	10.33%	50.54%	35.07%	**63.05%**	37.10%	47.92%
			std. r.	13.34% (5.7)	2.42% (2.3)	0.82% (5.0)	**2.43% (1.0)**	1.51% (4.3)	1.79% (2.7)
2	$n = 10$	IGD	mean	1.77e-2	7.62e-4	1.46e-3	7.12e-4	1.86e-3	**3.48e-4**

	$n_t = 10$ $\tau_t = 1000$		std. r.	1.3e-2 (5.6)	1.4e-4 (2.5)	5.4e-5 (4.3)	1.1e-4 (2.5)	2.3e-4 (5.1)	**1.6e-5 (1.0)**
		HVR	mean	23.09%	81.70%	59.14%	83.91%	59.13%	**89.71%**
			std. r.	30.19% (5.6)	1.53% (3.0)	1.04% (4.7)	1.01% (2.0)	3.89% (4.7)	**0.46% (1.0)**
3	$n = 10$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	1.96e-3	3.31e-4	5.64e-4	5.59e-4	1.43e-3	**3.00e-4**
			std. r.	4.9e-3 (3.6)	4.3e-5 (2.4)	2.0e-5 (4.1)	2.5e-4 (3.6)	5.1e-5 (5.9)	**3.4e-4 (1.4)**
		HVR	mean	81.48%	91.46%	81.43%	88.27%	67.45%	**93.77%**
			std. r.	26.2% (3.4)	0.52% (2.5)	0.49% (4.8)	2.94% (3.0)	1.13% (5.9)	**4.23% (1.4)**
4	$n = 10$ $n_t = 2$ $\tau_t = 500$	IGD	mean	4.18e-2	1.63e-1	1.50e-2	1.46e-2	**1.02e-2**	1.08e-2
			std. r.	4.2e-3 (5.0)	9.7e-3 (6.0)	1.1e-3 (3.6)	1.7e-3 (3.3)	**6.6e-4 (1.1)**	1.1e-3 (2.0)
		HVR	mean	2.48%	0.83%	5.26%	**18.35%**	11.87%	14.20%
			std. r.	1.84% (5.3)	0.86% (5.7)	1.80% (4.0)	**1.88% (1.0)**	2.00% (2.7)	1.30% (2.3)
5	$n = 10$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	1.42e-2	2.44e-2	**2.67e-3**	5.98e-3	4.93e-3	**2.80e-3**
			std. r.	1.1e-3 (5.0)	4.0e-3 (6.0)	**1.4e-4 (1.5)**	3.7e-4 (4.0)	2.1e-4 (3.0)	**5.0e-4 (1.5)**
		HVR	mean	4.99%	16.10%	40.57%	38.16%	32.74%	**56.76%**
			std. r.	4.98% (5.9)	1.83% (5.1)	2.06% (2.4)	3.20% (2.6)	2.31% (4.0)	**5.82% (1.0)**
6	$n = 10$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	1.60e-2	5.08e-3	**6.21e-4**	2.37e-3	3.16e-3	2.17e-3
			std. r.	3.2e-3 (6.0)	4.8e-3 (3.6)	**4.3e-5 (1.0)**	6.5e-4 (3.0)	2.2e-4 (4.4)	3.5e-4 (2.8)
		HVR	mean	16.60%	68.32%	**81.06%**	72.50%	48.46%	70.03%
			std. r.	13.9% (5.9)	5.80% (3.5)	**0.97% (1.0)**	3.62% (2.4)	1.57% (5.1)	4.46% (3.1)
7	$n = 20$ $n_t = 10$ $\tau_t = 500$	IGD	mean	6.15e-2	1.72e-2	2.22e-2	3.05e-2	**1.46e-2**	2.29e-2
			std. r.	2.3e-2 (6.0)	9.2e-4 (2.0)	1.4e-3 (3.7)	8.9e-3 (4.6)	**1.2e-3 (1.0)**	2.4e-3 (3.7)
		HVR	mean	4.21%	16.73%	11.58%	**18.19%**	8.44%	2.82%
			std. r.	5.83% (5.1)	1.14% (1.7)	0.64% (3.1)	**3.60% (1.4)**	1.47% (4.3)	0.22% (5.3)
8	$n = 20$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	4.92e-2	**3.40e-3**	7.38e-3	4.16e-3	6.30e-3	8.21e-3
			std. r.	3.5e-2 (6.0)	**4.1e-4 (1.5)**	3.0e-4 (4.0)	1.6e-3 (1.6)	6.4e-4 (3.0)	5.7e-4 (4.9)
		HVR	mean	6.92%	43.05%	22.88%	**56.37%**	25.15%	12.87%
			std. r.	12.00% (5.5)	2.39% (2.0)	0.45% (4.1)	**3.61% (1.0)**	2.44% (3.2)	2.87% (5.2)
9	$n = 20$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	5.00e-2	1.01e-3	2.34e-3	**8.48e-4**	3.44e-3	5.58e-3
			std. r.	4.2e-2 (5.3)	1.2e-4 (1.7)	9.9e-5 (3.1)	**2.2e-4 (1.3)**	6.0e-4 (4.3)	2.6e-4 (5.3)
		HVR	mean	26.83%	75.57%	45.30%	**80.78%**	41.88%	24.01%
			std. r.	26.4% (5.1)	1.58% (2.0)	1.22% (3.7)	**1.90% (1.0)**	4.74% (3.9)	3.08% (5.3)
10	$n = 20$ $n_t = 2$ $\tau_t = 500$	IGD	mean	2.15e-1	5.34e-1	1.78e-1	6.12e-2	**3.56e-2**	7.98e-2
			std. r.	1.4e-2 (5.0)	1.6e-2 (6.0)	1.1e-2 (4.0)	3.6e-3 (2.0)	**4.4e-3 (1.0)**	9.1e-3 (3.0)
		HVR	mean	0%	0%	0%	**10.86%**	6.32%	1.06%
			std. r.	0% (5.1)	0% (5.1)	0.02% (4.7)	**1.77% (1.0)**	0.84% (2.0)	0.28% (3.0)
11	$n = 20$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	8.18e-2	3.29e-1	3.36e-2	3.13e-2	**2.19e-2**	2.80e-2
			std. r.	8.6e-3 (5.0)	2.8e-2 (6.0)	2.7e-3 (3.9)	3.2e-3 (3.0)	**9.0e-4 (1.0)**	3.3e-3 (2.1)
		HVR	mean	0.22%	0.10%	0.24%	**17.64%**	12.94%	4.26%
			std. r.	0.35% (4.8)	0.28% (5.4)	0.46% (4.8)	**2.12% (1.0)**	1.33% (2.0)	1.86% (3.0)
12	$n = 20$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	2.48e-2	2.75e-2	**4.70e-3**	1.16e-2	1.14e-2	1.30e-2
			std. r.	4.5e-3 (5.5)	4.2e-3 (5.5)	**2.7e-4 (1.0)**	1.2e-3 (2.7)	1.1e-3 (2.7)	1.5e-3 (3.6)
		HVR	mean	9.73%	18.19%	19.23%	**26.83%**	20.34%	17.40%
			std. r.	9.13% (4.8)	1.78% (4.1)	1.65% (3.8)	**1.82% (1.0)**	1.27% (3.1)	4.12% (4.2)

According to Table 8, in the majority of cases where the frequency is high, either the CTLBO or the PNSCCDMO algorithm is the superior algorithm, according to one of the evaluation criteria. In experiments where the dimension number is 10, the VECTLBO algorithm is either the superior algorithm or one of the two best algorithms in many cases (Experiments 1 to 6). When the severity of change is increased, the CSADMO algorithm is less affected and becomes the major rival of the VECTLBO algorithm in terms of efficiency. By reducing the frequency of changes, it benefits from opportunities for convergence and surpasses the VECTLBO algorithm (Experiments 5 and 6).

On increasing the number of dimensions to 20, the PNSCCDMO algorithm achieved superior performance in most cases, and the DNSGA-II B and CTLBO algorithms were its major rivals when the severity of changes was lower (Experiments 7 to 9) and higher (Experiments 10 and 11) respectively.

6.2.1.7. Evaluation and Analysis of Algorithms for dMOP3 Problem

Although there are similarities between the dMOP3 and FDA1 problems, dMOP3 is of greater complexity due to changes in the variable that controls the spread of the POF solutions. The results of experiments for dMOP3 are presented in Table 9.

Table 9: Performance of the algorithms for dMOP3 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$ $n_t = 10$ $\tau_t = 500$	IGD	mean	3.83e-3	2.23e-3	**1.68e-3**	2.08e-3	2.09e-3	2.03e-3
			std. r.	6.6e-4 (6.0)	2.5e-4 (4.3)	**1.2e-4 (1.1)**	2.0e-4 (3.6)	1.5e-4 (3.4)	1.1e-4 (2.6)
		HVR	mean	45.05%	59.47%	**62.27%**	60.98%	57.22%	56.90%
			std. r.	4.87% (6.0)	1.95% (3.0)	**2.00% (1.7)**	3.77% (2.0)	2.76% (4.1)	2.36% (4.1)
2	$n = 10$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	1.21e-3	**4.93e-4**	6.97e-4	7.97e-4	1.21e-3	5.52e-4
			std. r.	2.9e-4 (5.3)	**6.1e-5 (1.2)**	2.7e-5 (3.2)	1.0e-4 (3.8)	1.3e-4 (5.7)	3.3e-5 (1.8)
		HVR	mean	81.23%	**90.47%**	82.69%	87.22%	74.55%	87.34%
			std. r.	3.23% (4.6)	**0.93% (1.0)**	0.63% (4.4)	1.06% (2.6)	2.40% (6.0)	0.66% (2.4)
3	$n = 10$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	5.81e-4	2.98e-4	3.06e-4	4.91e-4	8.34e-4	**2.12e-4**
			std. r.	1.6e-4 (4.8)	4.7e-5 (2.6)	9.1e-6 (2.4)	7.7e-5 (4.3)	1.6e-4 (5.9)	**3.9e-6 (1.0)**
		HVR	mean	89.41%	94.81%	92.69%	92.59%	82.04%	**95.87%**
			std. r.	1.79% (4.9)	0.65% (1.9)	0.17% (3.3)	0.84% (3.8)	3.84% (6.0)	**0.13% (1.1)**
4	$n = 10$ $n_t = 2$ $\tau_t = 500$	IGD	mean	1.28e-2	5.59e-2	7.40e-3	4.70e-3	5.04e-3	**3.43e-3**
			std. r.	1.4e-3 (5.0)	2.21e-3 (6.0)	7.1e-4 (4.0)	2.9e-4 (2.1)	4.2e-4 (2.8)	**3.7e-4 (1.0)**
		HVR	mean	10.41%	29.25%	17.38%	33.38%	35.79%	**44.77%**
			std. r.	2.34% (6.0)	1.47% (4.0)	1.72% (5.0)	2.18% (2.9)	3.04% (2.1)	**2.71% (1.0)**
5	$n = 10$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	4.35e-3	2.80e-2	1.19e-3	1.93e-3	3.11e-3	**7.59e-4**
			std. r.	1.2e-3 (4.9)	1.6e-3 (6.0)	6.7e-5 (2.0)	2.4e-4 (3.0)	4.6e-4 (4.1)	**7.7e-5 (1.0)**
		HVR	mean	42.41%	34.81%	72.24%	69.71%	52.14%	**83.89%**
			std. r.	5.22% (5.1)	2.26% (5.9)	1.27% (2.2)	2.32% (2.8)	3.53% (4.0)	**1.43% (1.0)**
6	$n = 10$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	1.87e-3	4.51e-3	3.65e-4	9.42e-4	1.73e-3	**2.10e-4**
			std. r.	6.5e-4 (4.7)	9.4e-4 (6.0)	1.8e-5 (2.0)	2.1e-4 (3.1)	2.9e-4 (4.2)	**8.1e-6 (1.0)**
		HVR	mean	78.76%	53.95%	91.53%	87.04%	64.16%	**96.13%**
			std. r.	6.34% (4.0)	3.60% (5.9)	0.39% (2.0)	2.43% (3.1)	5.24% (5.0)	**0.20% (1.0)**
7	$n = 30$ $n_t = 10$	IGD	mean	5.85e-2	8.37e-2	2.99e-2	2.29e-2	**2.05e-2**	2.35e-2
			std. r.	8.1e-3 (5.0)	9.3e-3 (6.0)	1.6e-3 (4.0)	1.1e-3 (2.4)	**8.3e-4 (1.3)**	3.3e-3 (2.3)
		HVR	mean	0.98%	1.27%	6.15%	**13.24%**	7.35%	6.36%

	$\tau_t = 500$		std. r.	0.47% (5.6)	0.92% (5.4)	1.61% (3.6)	**2.83% (1.0)**	0.76% (2.3)	0.73% (3.1)
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	2.31e-2	1.10e-2	9.46e-3	9.74e-3	1.09e-2	**7.81e-3**
			std. r.	3.0e-3 (6.0)	8.2e-4 (4.7)	5.2e-4 (2.5)	9.0e-4 (2.9)	1.2e-3 (3.9)	**8.1e-4 (1.0)**
		HVR	mean	12.52%	24.70%	19.41%	**27.04%**	13.56%	23.79%
			std. r.	4.65% (5.5)	1.66% (2.3)	0.81% (3.9)	**0.99% (1.1)**	2.14% (5.4)	3.10% (2.8)
9	$n = 30$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	1.42e-2	1.73e-3	3.17e-3	2.46e-3	3.54e-3	**1.44e-3**
			std. r.	2.6e-3 (6.0)	4.2e-4 (2.0)	2.3e-4 (4.1)	2.7e-4 (2.9)	4.2e-4 (4.9)	**1.4e-4 (1.1)**
		HVR	mean	30.31%	**68.44%**	44.43%	64.37%	37.03%	67.82%
			std. r.	11.1% (5.7)	**3.93% (1.4)**	1.29% (4.0)	1.98% (2.8)	3.56% (5.3)	1.99% (1.8)
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	8.20e-2	2.76e-1	1.83e-1	**3.27e-2**	3.74e-2	5.04e-2
			std. r.	5.2e-3 (4.0)	6.8e-3 (6.0)	4.1e-3 (5.0)	**1.5e-3 (1.0)**	4.5e-3 (2.0)	4.4e-3 (3.0)
		HVR	mean	0%	2.87%	0.07%	8.77%	**15.18%**	6.06%
			std. r.	0% (5.6)	1.94% (3.9)	0.19% (5.4)	2.10% (2.1)	**3.73% (1.0)**	1.56% (3.0)
11	$n = 30$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	5.05e-2	2.16e-1	9.07e-2	**1.86e-2**	3.00e-2	2.01e-2
			std. r.	3.7e-3 (4.0)	4.2e-3 (6.0)	2.6e-3 (5.0)	**9.6e-4 (1.5)**	3.1e-3 (2.9)	3.8e-3 (1.6)
		HVR	mean	0.00%	0.87%	0.10%	19.50%	**29.17%**	15.43%
			std. r.	0.00% (5.8)	0.99% (4.3)	0.10% (4.9)	1.29% (2.1)	**2.17% (1.0)**	3.15% (2.9)
12	$n = 30$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	2.11e-2	1.24e-1	2.54e-2	7.48e-3	1.58e-2	**3.78e-3**
			std. r.	2.3e-3 (4.0)	6.6e-3 (6.0)	1.2e-3 (5.0)	5.1e-4 (2.0)	1.5e-3 (3.0)	**7.0e-4 (1.0)**
		HVR	mean	1.98%	0.79%	3.53%	27.88%	38.54%	**47.49%**
			std. r.	1.19% (5.3)	0.40% (5.7)	0.91% (4.0)	1.48% (3.0)	1.06% (1.9)	**5.65% (1.1)**

The results presented in Table 9 indicate that the VECTLBO algorithm gave the best results for the dMOP3 problem in many cases where the dimension number was low. Both CSADMO and DNSGA-II B algorithms outperformed it in some cases where the severity was low and the frequency was high (Experiments 1 and 2). When the dimension number was increased, CTLBO and PNSCCDMO were more successful in cases where the frequency of changes was higher, but as the frequency decreased, VECTLBO gave closer results and finally achieved the best results.

6.2.1.8. Evaluation and Analysis of Algorithms for F5 Problem

In this section, the algorithms are evaluated using the F5 problem, which is a type II problem with nonlinear correlation between variables. The results of the experiments are presented in Table 10.

Table 10: Performance of the algorithms for F5 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$ $n_t = 10$ $\tau_t = 500$	IGD	mean	4.09e-3	**3.95e-3**	4.72e-3	4.17e-3	7.96e-3	9.67e-3
			std. r.	3.1e-4 (1.9)	**9.6e-5 (1.4)**	2.1e-4 (3.9)	1.5e-4 (2.8)	1.3e-3 (5.2)	1.6e-3 (5.8)
		HVR	mean	78.78%	**79.47%**	76.95%	77.44%	63.64%	62.11%
			std. r.	1.22% (2.0)	**0.44% (1.4)**	0.84% (3.6)	0.74% (3.0)	3.87% (5.4)	3.83% (5.6)
2	$n = 10$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	3.87e-3	**3.87e-3**	4.25e-3	3.91e-3	5.08e-3	5.93e-2
			std. r.	1.6e-4 (1.9)	**5.1e-5 (1.6)**	6.0e-5 (3.9)	1.0e-4 (2.6)	5.0e-4 (5.1)	1.1e-3 (5.9)
		HVR	mean	80.54%	**80.61%**	80.02%	80.11%	73.67%	73.05%
			std. r.	0.39% (1.7)	**0.16% (1.6)**	0.21% (3.7)	0.36% (3.0)	1.83% (5.3)	3.18% (5.7)

3	$n = 10$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	3.89e-3	**3.88e-3**	4.19e-3	3.89e-3	4.26e-3	4.54e-3
			std. r.	3.9e-5 (2.3)	**4.8e-5 (1.7)**	2.4e-5 (4.4)	5.8e-5 (2.0)	1.9e-4 (4.7)	1.6e-4 (5.9)
		HVR	mean	81.09%	**81.12%**	80.82%	80.87%	78.23%	78.56%
			std. r.	0.08% (1.7)	**0.06% (1.4)**	0.09% (3.4)	0.17% (3.4)	1.23% (5.4)	0.50% (5.6)
4	$n = 10$ $n_t = 2$ $\tau_t = 500$	IGD	mean	2.68e-2	8.82e-2	2.25e-2	**1.90e-2**	2.54e-2	3.47e-2
			std. r.	1.2e-3 (3.8)	7.2e-2 (6.0)	1.6e-3 (2.0)	**1.0e-3 (1.1)**	8.6e-4 (3.1)	1.9e-3 (5.0)
		HVR	mean	19.02%	18.76%	25.30%	**29.50%**	22.62%	10.50%
			std. r.	1.35% (4.4)	2.08% (4.5)	1.73% (2.1)	**2.28% (1.1)**	1.84% (2.9)	1.77% (6.0)
5	$n = 10$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	1.63e-2	2.02e-2	1.69e-2	**1.30e-2**	1.98e-2	2.24e-2
			std. r.	9.8e-4 (2.3)	5.3e-4 (4.8)	1.20% (2.8)	**4.9e-4 (1.0)**	1.3e-3 (4.1)	1.1e-3 (6.0)
		HVR	mean	38.10%	31.86%	40.14%	**47.34%**	33.88%	26.91%
			std. r.	1.90% (2.8)	1.38% (4.9)	1.86% (2.2)	**1.50% (1.0)**	1.71% (4.1)	2.48% (6.0)
6	$n = 10$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	1.25e-2	1.71e-2	1.43e-2	**1.15e-2**	1.66e-2	1.71e-2
			std. r.	5.9e-4 (1.9)	4.9e-4 (5.0)	1.0e-3 (3.1)	**5.0e-4 (1.1)**	7.6e-4 (4.6)	8.8e-4 (5.3)
		HVR	mean	49.68%	41.56%	46.33%	**53.22%**	41.99%	38.19%
			std. r.	1.40% (2.0)	1.00% (4.7)	2.29% (3.0)	**1.25% (1.0)**	1.04% (4.6)	2.18% (5.7)
7	$n = 20$ $n_t = 10$ $\tau_t = 500$	IGD	mean	**5.98e-3**	7.83e-3	7.99e-3	1.45e-2	3.53e-2	4.53e-2
			std. r.	**9.6e-4 (1.7)**	5.1e-3 (2.1)	3.2e-3 (2.4)	3.9e-3 (3.8)	1.2e-2 (5.1)	9.3e-3 (5.9)
		HVR	mean	**70.77%**	66.28%	61.89%	40.34%	16.87%	8.48%
			std. r.	**2.48% (1.5)**	1.39% (1.9)	12.03% (2.8)	5.09% (3.8)	8.84% (5.1)	8.09% (5.9)
8	$n = 20$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	5.18e-3	**6.70e-3**	5.59e-3	5.01e-3	7.13e-3	2.12e-2
			std. r.	2.0e-3 (2.3)	**6.9e-3 (2.1)**	1.1e-3 (3.7)	1.4e-3 (2.5)	2.1e-3 (4.4)	7.9e-3 (6.0)
		HVR	mean	**73.84%**	70.73%	72.75%	74.15%	65.76%	33.06%
			std. r.	**8.59% (2.0)**	18.07% (2.2)	4.32% (3.3)	4.37% (2.9)	8.05% (4.6)	13.11% (6.0)
9	$n = 20$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	3.95e-3	**3.89e-3**	4.57e-3	4.07e-3	5.02e-3	1.22e-2
			std. r.	1.9e-4 (1.9)	**9.5e-5 (1.4)**	3.7e-4 (4.3)	1.2e-4 (2.9)	1.1e-3 (4.5)	6.4e-3 (6.0)
		HVR	mean	79.37%	**79.58%**	78.68%	79.07%	75.54%	54.84%
			std. r.	0.87% (1.9)	**0.28% (1.7)**	1.40% (3.3)	0.38% (3.4)	4.17% (4.7)	15.00% (6.0)
10	$n = 20$ $n_t = 2$ $\tau_t = 500$	IGD	mean	7.04e-2	5.67e-1	6.02e-2	5.25e-2	**5.02e-2**	7.52e-2
			std. r.	5.1e-3 (4.2)	2.5e-1 (6.0)	5.1e-3 (2.9)	2.4e-3 (1.6)	**2.5e-3 (1.5)**	7.5e-3 (4.8)
		HVR	mean	1.28%	**7.36%**	6.12%	2.21%	5.19%	0.46%
			std. r.	1.00% (5.0)	**2.00% (1.4)**	1.35% (1.8)	0.77% (4.3)	1.13% (2.8)	0.55% (5.7)
11	$n = 20$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	3.74e-2	4.35e-2	3.14e-2	**3.00e-2**	3.87e-2	5.01e-2
			std. r.	2.2e-3 (3.7)	5.7e-3 (4.8)	1.3e-3 (1.6)	**3.2e-3 (1.4)**	1.8e-3 (3.7)	4.4e-3 (5.8)
		HVR	mean	9.69%	17.20%	**16.71%**	12.99%	13.82%	3.79%
			std. r.	2.14% (4.8)	2.60% (2.0)	**1.18% (1.6)**	1.98% (3.4)	0.90% (3.2)	1.32% (6.0)
12	$n = 20$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	2.07e-2	2.34e-2	2.43e-2	**1.82e-2**	3.14e-2	3.21e-2
			std. r.	1.4e-3 (1.8)	8.9e-4 (3.3)	8.3e-4 (3.7)	**1.4e-3 (1.2)**	9.8e-4 (5.4)	4.0e-3 (5.6)
		HVR	mean	29.36%	25.80%	25.64%	**35.32%**	20.57%	13.03%
			std. r.	1.69% (2.1)	1.74% (3.3)	0.87% (3.5)	**2.75% (1.1)**	1.39% (5.0)	2.95% (6.0)

As can be seen from Table 10, in cases where the severity of changes is lower (Experiments 1 to 3 and 7 to 9), DNSGA-II B and DNSGA-II A outperform the other algorithms. In cases where the severity of changes is high and the number of dimensions is low (Experiments 4 to 6), PNSCCDMO is the best algorithm, but when the number of dimensions is increased, each of the CTLBO, DNSGA-II B and CSADMO algorithms, in certain cases, obtain results close to or superior to those of PNSCCDMO. On increasing the severity of changes, CTLBO is less affected and its rank is improved, but it is less successful in improving convergence as the frequency decreases.

6.2.1.9. Evaluation and Analysis of Algorithms for F6 Problem

F6 is another type II problem that, similarly to F5, has nonlinear correlation between variables. The experimental results for this problem are reported in Table 11.

Table 11: Performance of the algorithms for F6 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$ $n_t = 10$ $\tau_t = 500$	IGD	mean	5.00e-3	5.07e-3	5.62e-3	**4.93e-3**	6.65e-3	8.23e-3
			std. r.	1.1e-4 (2.4)	1.7e-4 (1.9)	1.5e-4 (4.0)	**2.3e-4 (1.7)**	7.8e-4 (5.0)	1.1e-3 (6.0)
		HVR	mean	74.20%	74.01%	73.05%	**74.26%**	63.71%	67.59%
			std. r.	0.52% (2.3)	0.59% (2.2)	0.97% (3.6)	**1.05% (1.9)**	2.66% (5.8)	2.33% (5.2)
2	$n = 10$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	5.08e-3	4.97e-3	5.42e-3	**4.78e-3**	5.40e-3	6.01e-3
			std. r.	9.6e-5 (3.0)	2.3e-5 (1.9)	1.6e-4 (4.5)	**1.0e-4 (1.2)**	2.4e-4 (4.7)	4.0e-4 (5.7)
		HVR	mean	75.08%	75.43%	74.92%	**76.18%**	72.80%	73.04%
			std. r.	0.38% (3.7)	0.11% (2.1)	0.83% (3.2)	**0.44% (1.2)**	1.49% (5.1)	0.71% (5.7)
3	$n = 10$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	4.99e-3	5.03e-3	5.24e-3	**4.75e-3**	4.94e-3	5.36e-3
			std. r.	6.0e-5 (2.9)	2.9e-5 (3.6)	4.7e-5 (5.1)	**6.4e-5 (1.0)**	1.3e-4 (2.6)	1.4e-4 (5.9)
		HVR	mean	76.00%	75.97%	76.37%	**76.95%**	75.88%	75.42%
			std. r.	0.20% (4.3)	0.19% (4.0)	0.30% (2.9)	**0.17% (1.0)**	0.88% (3.6)	0.74% (5.3)
4	$n = 10$ $n_t = 2$ $\tau_t = 500$	IGD	mean	1.32e-2	2.02e-2	**1.23e-2**	1.47e-2	1.33e-2	2.29e-2
			std. r.	9.9e-4 (2.1)	4.8e-3 (5.4)	**9.5e-4 (1.7)**	1.1e-3 (3.9)	1.0e-3 (2.3)	1.3e-3 (5.6)
		HVR	mean	35.30%	31.04%	**38.01%**	30.25%	34.76%	14.79%
			std. r.	2.06% (2.4)	1.96% (4.4)	**2.88% (1.5)**	3.21% (4.1)	3.60% (2.6)	1.74% (6.0)
5	$n = 10$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	9.25e-3	9.56e-3	**8.12e-3**	8.36e-3	9.43e-3	1.42e-2
			std. r.	7.0e-4 (3.4)	5.2e-4 (4.2)	**3.0e-4 (1.4)**	9.1e-4 (2.1)	6.4e-4 (3.9)	1.3e-3 (6.0)
		HVR	mean	49.19%	47.46%	**54.58%**	52.96%	48.87%	32.79%
			std. r.	1.95% (3.4)	1.36% (4.3)	**1.22% (1.4)**	2.91% (2.1)	2.64% (3.8)	4.13% (6.0)
6	$n = 10$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	8.08e-3	8.27e-3	**6.76e-3**	7.38e-3	7.35e-3	9.40e-3
			std. r.	4.6e-4 (3.9)	5.7e-5 (4.4)	**1.5e-4 (1.1)**	5.4e-4 (2.9)	5.1e-4 (2.7)	6.3e-4 (6.0)
		HVR	mean	55.70%	54.70%	**62.45%**	59.15%	58.91%	50.39%
			std. r.	1.71% (4.0)	0.50% (4.9)	**0.72% (1.1)**	1.84% (2.6)	1.94% (2.4)	2.19% (6.0)
7	$n = 20$ $n_t = 10$ $\tau_t = 500$	IGD	mean	7.21e-3	**7.54e-3**	1.16e-2	3.18e-2	1.43e-2	2.11e-2
			std. r.	2.0e-3 (1.9)	**1.6e-3 (1.7)**	7.0e-3 (2.9)	3.7e-3 (5.9)	4.7e-3 (3.7)	6.8e-3 (4.9)
		HVR	mean	**66.31%**	64.82%	51.70%	7.34%	38.41%	43.42%
			std. r.	**7.43% (1.7)**	6.05% (2.0)	17.67% (3.1)	3.71% (6.0)	12.97% (4.4)	11.68% (3.8)
8	$n = 20$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	5.94e-3	5.33e-3	6.56e-3	**5.56e-3**	8.82e-3	1.30e-2
			std. r.	8.5e-4 (2.8)	2.1e-4 (2.0)	8.6e-4 (3.6)	**8.7e-4 (1.7)**	2.2e-3 (5.0)	1.1e-2 (5.9)
		HVR	mean	69.64%	**72.44%**	69.90%	71.25%	61.77%	58.52%
			std. r.	3.87% (2.8)	**0.56% (2.1)**	3.07% (3.2)	2.95% (2.2)	3.93% (5.8)	17.83% (4.9)
9	$n = 20$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	5.17e-3	5.29e-3	5.86e-3	**4.95e-3**	6.49e-3	7.70e-3
			std. r.	1.3e-4 (2.5)	3.5e-4 (3.0)	1.0e-3 (4.2)	**1.8e-4 (1.3)**	1.8e-3 (4.4)	1.4e-3 (5.6)
		HVR	mean	73.78%	73.49%	73.91%	**74.91%**	69.14%	67.21%
			std. r.	0.78% (3.1)	1.53% (3.5)	1.62% (2.5)	**0.93% (1.6)**	5.38% (4.8)	2.96% (5.5)
10	$n = 20$	IGD	mean	3.07e-2	1.35e-1	**2.53e-2**	4.69e-2	3.10e-2	5.06e-2
			std. r.	4.8e-3 (2.3)	4.5e-2 (6.0)	**3.2e-3 (1.1)**	1.9e-3 (4.2)	2.9e-3 (2.6)	5.0e-3 (4.8)

	$n_t = 2$ $\tau_t = 500$	HVR	mean	7.75%	4.48%	**14.68%**	0.31%	8.99%	0.81%
			std. r.	1.58% (2.9)	3.98% (4.0)	**2.42% (1.1)**	0.22% (5.8)	2.71% (2.4)	0.45% (4.8)
11	$n = 20$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	1.66e-2	1.79e-2	**1.58e-2**	2.53e-2	2.13e-2	3.33e-2
			std. r.	1.4e-3 (2.0)	2.0e-3 (2.7)	**9.2e-4 (1.4)**	1.9e-3 (5.0)	1.9e-3 (3.9)	4.1e-3 (6.0)
		HVR	mean	26.06%	25.23%	**29.69%**	11.03%	21.64%	4.98%
			std. r.	1.65% (2.1)	2.43% (3.0)	**1.90% (1.1)**	1.78% (5.1)	1.39% (3.8)	1.91% (6.0)
12	$n = 20$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	**1.14e-2**	1.15e-2	1.23e-2	1.29e-2	1.54e-2	2.11e-2
			std. r.	**9.4e-4 (1.5)**	5.4e-4 (2.1)	1.6e-3 (2.7)	7.0e-4 (3.9)	1.6e-3 (4.9)	3.3e-3 (5.9)
		HVR	mean	**40.82%**	39.74%	39.88%	37.18%	32.25%	19.00%
			std. r.	**3.03% (1.8)**	1.86% (2.4)	2.75% (2.3)	2.75% (3.7)	3.08% (4.8)	5.24% (6.0)

For the F6 problem, the results in Table 11 indicate that in cases with a smaller dimension number (Experiments 1 to 6), the PNSCCDMO and CSADMO algorithms obtain the best results in environments where the severity of changes is low and high respectively. However, when the dimension number increases, the above algorithms cannot preserve their absolute superiority, and the DNSGA-II A and DNSGA-II B algorithms surpass PNSCCDMO and CSADMO as the frequency increases and decreases respectively. The results obtained for CTLBO indicate that its rank improves as the severity of changes increases, which means that it is less affected by increases in severity.

6.2.1.10. Evaluation and Analysis of Algorithms for F8 Problem

In this section, the algorithms are evaluated using the F8 problem, which is a type I problem. It is a problem with three objectives, like FDA4, with the difference that there is a nonlinear correlation between its variables. Table 12 presents the results of the experiments for this problem.

Table 12: Performance of the algorithms for F8 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 15$ $n_t = 10$ $\tau_t = 500$	IGD	mean	7.69e-4	7.49e-4	8.79e-4	3.54e-3	**5.66e-4**	6.22e-4
			std. r.	5.8e-5 (3.4)	1.0e-4 (3.6)	1.1e-4 (4.6)	2.6e-4 (6.0)	**1.2e-4 (1.4)**	4.4e-5 (2.0)
		HVR	mean	48.67%	52.37%	52.81%	0.16%	**72.17%**	65.97%
			std. r.	5.33% (4.3)	7.36% (4.0)	2.67% (3.8)	0.20% (6.0)	**2.36% (1.0)**	2.18% (2.0)
2	$n = 15$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	5.62e-4	5.22e-4	5.22e-4	1.78e-3	**3.99e-4**	4.07e-4
			std. r.	3.4e-5 (4.5)	6.0e-5 (3.6)	6.9e-5 (3.6)	1.7e-4 (6.0)	**5.1e-5 (1.5)**	4.1e-5 (1.8)
		HVR	mean	63.97%	66.02%	72.44%	3.15%	**82.02%**	77.71%
			std. r.	2.75% (4.6)	4.11% (4.1)	2.76% (3.1)	1.52% (6.0)	**1.50% (1.1)**	2.72% (2.0)
3	$n = 15$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	4.28e-4	4.33e-4	4.17e-4	1.01e-3	3.24e-4	**2.63e-4**
			std. r.	1.8e-5 (4.2)	4.2e-5 (4.2)	3.3e-5 (3.7)	8.6e-5 (6.0)	4.1e-5 (2.0)	**1.3e-5 (1.0)**
		HVR	mean	71.41%	74.54%	79.24%	21.29%	87.26%	**87.68%**
			std. r.	3.37% (4.7)	2.85% (4.2)	2.17% (3.2)	5.08% (6.0)	0.97% (1.7)	**0.57% (1.3)**

4	$n = 15$ $n_t = 2$ $\tau_t = 500$	IGD	mean	2.55e-3	1.70e-3	2.16e-3	2.63e-3	**1.32e-3**	1.84e-3
			std. r.	5.4e-4 (5.4)	1.5e-4 (2.5)	2.8e-4 (4.1)	2.0e-4 (5.4)	**9.5e-5 (1.1)**	2.3e-4 (2.5)
		HVR	mean	4.06%	10.26%	5.75%	5.78%	**24.28%**	18.84%
			std. r.	2.59% (5.4)	4.87% (3.4)	3.05% (4.8)	1.86% (4.4)	**4.51% (1.1)**	1.87% (2.0)
5	$n = 15$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	1.11e-3	9.73e-4	1.07e-3	1.73e-3	**7.19e-4**	8.05e-4
			std. r.	7.0e-5 (4.6)	6.6e-5 (3.1)	1.1e-4 (4.0)	1.2e-4 (6.0)	**1.0e-4 (1.3)**	1.2e-4 (2.0)
		HVR	mean	31.64%	38.29%	27.50%	6.34%	**56.93%**	48.88%
			std. r.	3.98% (4.1)	4.94% (3.1)	4.49% (4.8)	2.67% (6.0)	**6.85% (1.1)**	7.34% (1.9)
6	$n = 15$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	6.28e-4	5.95e-4	7.90e-4	9.67e-4	4.61e-4	**4.58e-4**
			std. r.	2.2e-5 (3.8)	2.5e-5 (3.2)	5.6e-5 (5.0)	7.7e-5 (6.0)	2.3e-5 (1.7)	**1.9e-5 (1.3)**
		HVR	mean	59.75%	60.69%	43.39%	24.59%	**78.26%**	76.48%
			std. r.	1.90% (3.8)	2.24% (3.2)	2.93% (5.0)	5.85% (6.0)	**1.21% (1.3)**	1.47% (1.7)
7	$n = 30$ $n_t = 10$ $\tau_t = 500$	IGD	mean	2.62e-3	2.50e-3	3.80e-3	1.20e-2	**1.31e-3**	1.47e-3
			std. r.	4.6e-4 (3.6)	3.9e-4 (3.4)	2.9e-4 (5.0)	6.3e-4 (6.0)	**1.3e-4 (1.3)**	2.1e-4 (1.8)
		HVR	mean	8.46%	14.34%	4.68%	0.00%	25.17%	**31.02%**
			std. r.	3.89% (4.1)	7.07% (2.9)	2.13% (4.8)	0.00% (6.0)	9.34% (2.0)	**4.64% (1.3)**
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	1.48e-3	1.49e-3	1.78e-3	6.32e-3	**8.71e-4**	**8.20e-4**
			std. r.	1.5e-4 (4.0)	2.2e-4 (3.6)	3.8e-4 (4.4)	4.3e-4 (6.0)	**1.9e-4 (1.5)**	**4.4e-5 (1.5)**
		HVR	mean	29.28%	34.47%	41.98%	0.00%	55.15%	**58.52%**
			std. r.	4.46% (4.8)	9.29% (3.9)	5.77% (3.4)	0.00% (6.0)	3.19% (1.9)	**2.35% (1.1)**
9	$n = 30$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	8.64e-4	7.94e-4	1.56e-3	3.23e-3	6.82e-4	**5.13e-4**
			std. r.	1.1e-4 (3.6)	1.1e-4 (3.0)	2.2e-4 (5.0)	3.1e-4 (6.0)	3.4e-3 (2.0)	**3.7e-5 (1.4)**
		HVR	mean	51.88%	57.04%	57.61%	0.08%	**71.02%**	**71.26%**
			std. r.	4.97% (4.8)	5.39% (3.9)	3.96% (3.4)	0.18% (6.0)	**3.48% (1.5)**	**2.37% (1.5)**
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	8.10e-3	7.02e-3	1.30e-2	7.15e-3	**4.37e-3**	6.22e-3
			std. r.	1.6e-3 (4.3)	1.4e-3 (3.3)	1.5e-3 (6.0)	2.6e-4 (3.6)	**6.1e-4 (1.0)**	7.7e-4 (2.9)
		HVR	mean	0.00%	0.00%	0.00%	2.75%	**4.26%**	3.37%
			std. r.	0.00% (4.9)	0.00% (4.9)	0.00% (4.9)	1.66% (2.5)	**0.80% (1.4)**	1.41% (2.4)
11	$n = 30$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	4.21e-3	3.02e-3	5.59e-3	5.18e-3	**2.40e-3**	3.32e-3
			std. r.	6.3e-4 (4.0)	4.6e-4 (2.3)	8.3e-4 (5.6)	3.9e-4 (5.1)	**3.5e-4 (1.2)**	3.9e-4 (2.8)
		HVR	mean	0.08%	0.62%	0.26%	0.95%	8.26%	**10.16%**
			std. r.	0.11% (5.3)	0.61% (4.2)	0.47% (5.1)	0.67% (3.4)	1.93% (1.6)	**2.16% (1.4)**
12	$n = 30$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	1.84e-3	1.55e-3	4.78e-3	3.37e-3	**1.16e-3**	1.26e-3
			std. r.	8.0e-5 (4.0)	1.3e-4 (2.7)	7.2e-4 (6.0)	3.3e-4 (5.0)	**9.3e-5 (1.3)**	3.7e-4 (2.0)
		HVR	mean	9.46%	16.01%	1.73%	1.53%	23.52%	**32.13%**
			std. r.	2.11% (4.0)	2.50% (3.0)	1.54% (5.3)	1.17% (5.7)	1.87% (2.0)	**3.22% (1.0)**

The results presented in Table 12 indicate that the CTLBO and VECTLBO algorithms outperform the other algorithms in all experiments for the F8 problem. When the dimension number is low, CTLBO is the best algorithm in most cases and VECTLBO can surpass it only in cases where the frequency is low enough. From the results, it is obvious that the performance of VECTLBO is less affected by the dimensionality of the problem.

6.2.1.11. Evaluation and Analysis of Algorithms for ZJZ Problem

In this section, experiments are designed to evaluate the algorithms using the ZJZ problem, which is a type II problem with nonlinear linkages between its decision variables. Table 13 presents the results of the above experiments.

Table 13: Performance of the algorithms for ZJZ problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$ $n_t = 10$ $\tau_t = 500$	IGD	mean	1.68e-3	1.98e-3	1.59e-3	1.92e-3	**5.18e-4**	7.27e-4
			std. r.	1.6e-4 (4.0)	2.2e-4 (5.3)	2.8e-4 (3.8)	2.3e-4 (4.9)	**1.3e-5 (1.0)**	4.5e-5 (2.0)
		HVR	mean	72.31%	71.08%	68.14%	62.52%	**83.61%**	79.62%
			std. r.	1.85% (3.5)	2.08% (3.9)	2.51% (4.7)	2.58% (5.9)	**0.38% (1.0)**	0.97% (2.0)
2	$n = 10$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	9.25e-4	1.24e-3	1.05e-3	8.31e-4	**3.54e-4**	5.57e-4
			std. r.	9.0e-5 (4.1)	1.3e-4 (5.7)	2.1e-4 (4.6)	1.2e-4 (3.6)	**7.9e-6 (1.0)**	5.8e-5 (2.0)
		HVR	mean	83.10%	83.29%	78.89%	79.36%	**88.87%**	84.62%
			std. r.	0.82% (3.6)	1.18% (3.3)	2.09% (5.5)	0.99% (5.4)	**0.28% (1.0)**	1.29% (2.2)
3	$n = 10$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	5.52e-4	9.84e-4	6.75e-4	4.75e-4	**2.65e-4**	4.16e-4
			std. r.	3.8e-5 (3.7)	9.5e-5 (6.0)	4.0e-5 (5.0)	6.7e-5 (2.9)	**5.1e-6 (1.0)**	7.5e-5 (2.4)
		HVR	mean	88.73%	88.39%	83.93%	86.39%	**92.01%**	88.24%
			std. r.	0.42% (2.9)	1.00% (3.1)	0.47% (6.0)	0.50% (4.9)	**0.18% (1.0)**	1.55% (3.1)
4	$n = 10$ $n_t = 2$ $\tau_t = 500$	IGD	mean	3.08e-3	3.44e-3	2.88e-3	2.81e-3	1.20e-3	**1.07e-3**
			std. r.	1.7e-4 (4.9)	4.2e-4 (5.9)	1.8e-4 (3.8)	7.1e-5 (3.5)	9.1e-5 (1.9)	**6.3e-5 (1.1)**
		HVR	mean	46.68%	41.72%	46.57%	48.51%	**71.99%**	71.37%
			std. r.	1.64% (4.6)	5.15% (5.5)	1.75% (4.5)	0.70% (3.4)	**1.19% (1.4)**	1.13% (1.6)
5	$n = 10$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	2.14e-3	2.74e-3	1.90e-3	2.16e-3	**6.05e-4**	7.51e-4
			std. r.	1.5e-4 (4.7)	4.2e-4 (5.6)	1.2e-4 (3.2)	9.0e-5 (4.5)	**3.3e-5 (1.0)**	2.6e-5 (2.0)
		HVR	mean	64.63%	59.73%	64.55%	60.45%	**82.86%**	79.49%
			std. r.	2.00% (3.4)	4.41% (5.0)	1.41% (4.0)	1.13% (5.6)	**0.73% (1.0)**	0.49% (2.0)
6	$n = 10$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	1.28e-3	1.97e-3	1.38e-3	1.44e-3	**3.37e-4**	5.88e-4
			std. r.	9.8e-5 (3.4)	2.4e-4 (6.0)	1.4e-4 (4.1)	1.1e-4 (4.4)	**9.5e-6 (1.0)**	3.9e-5 (2.0)
		HVR	mean	79.48%	74.78%	74.64%	73.86%	**89.62%**	84.55%
			std. r.	1.03% (3.1)	2.73% (4.7)	1.13% (4.6)	0.95% (5.6)	**0.15% (1.0)**	0.88% (2.0)
7	$n = 30$ $n_t = 10$ $\tau_t = 500$	IGD	mean	3.95e-3	4.25e-3	4.00e-3	3.85e-3	2.09e-3	**2.10e-3**
			std. r.	3.2e-4 (4.4)	2.6e-4 (5.5)	8.2e-5 (4.7)	3.8e-5 (3.4)	1.2e-4 (1.6)	**8.2e-5 (1.4)**
		HVR	mean	36.07%	34.21%	30.26%	33.50%	54.09%	**55.87%**
			std. r.	2.49% (3.4)	1.94% (4.0)	0.75% (6.0)	0.41% (4.6)	1.55% (1.8)	**0.94% (1.2)**
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	3.37e-3	3.43e-3	3.59e-3	3.55e-3	**1.53e-3**	1.66e-3
			std. r.	2.7e-4 (4.3)	2.1e-4 (3.9)	2.2e-4 (5.1)	6.4e-5 (4.7)	**7.2e-5 (1.1)**	8.2e-5 (1.9)
		HVR	mean	49.33%	49.17%	40.69%	38.63%	**64.39%**	63.75%
			std. r.	2.00% (3.6)	1.65% (3.4)	1.79% (5.1)	0.75% (5.9)	**0.98% (1.3)**	1.10% (1.7)
9	$n = 30$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	2.58e-3	2.43e-3	3.01e-3	2.71e-3	**9.55e-4**	1.27e-3
			std. r.	3.1e-4 (4.2)	1.9e-4 (3.4)	2.3e-4 (5.9)	8.7e-5 (4.5)	**5.3e-5 (1.0)**	6.7e-5 (2.0)
		HVR	mean	60.84%	62.67%	51.01%	49.87%	**73.65%**	70.71%
			std. r.	2.77% (3.7)	1.90% (3.3)	2.27% (5.5)	0.92% (5.5)	**0.82% (1.0)**	0.79% (2.0)
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	4.27e-3	4.42e-3	4.45e-3	3.94e-3	3.90e-3	**3.36e-3**
			std. r.	5.0e-5 (4.0)	1.1e-4 (5.4)	3.0e-5 (5.6)	4.0e-5 (2.5)	1.1e-4 (2.5)	**2.0e-4 (1.0)**
		HVR	mean	25.81%	20.74%	21.48%	31.54%	32.03%	**38.53%**
			std. r.	0.62% (4.0)	0.48% (5.9)	0.23% (5.1)	0.21% (2.6)	1.93% (2.4)	**1.64% (1.0)**
11	$n = 30$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	4.10e-3	4.40e-3	4.31e-3	3.82e-3	3.26e-3	**2.67e-3**
			std. r.	4.7e-5 (4.0)	8.4e-5 (5.8)	3.0e-5 (5.2)	2.6e-5 (3.0)	1.3e-4 (2.0)	**1.1e-4 (1.0)**
		HVR	mean	29.32%	24.29%	24.08%	33.58%	40.24%	**47.99%**
			std. r.	0.70% (4.0)	1.41% (5.6)	0.34% (5.4)	0.12% (3.0)	2.11% (2.0)	**0.97% (1.0)**
12	$n = 30$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	3.64e-3	4.23e-3	4.12e-3	3.57e-3	2.49e-3	**2.03e-3**
			std. r.	1.0e-4 (3.8)	1.8e-4 (5.7)	5.7e-5 (5.3)	5.7e-5 (3.2)	4.8e-5 (2.0)	**1.5e-4 (1.0)**
		HVR	mean	40.04%	34.91%	27.68%	37.35%	52.28%	**58.57%**
			std. r.	1.38% (3.0)	2.14% (4.8)	0.57% (6.0)	0.34% (4.2)	1.23% (2.0)	**1.08% (1.0)**

According to Table 13, it is clear that the CTLBO and VECTLBO algorithms achieved the best results for the ZJZ problem as they are ranked first or second. In cases where the problem is less complex, CTLBO had superior performance, but as the complexity increased, VECTLBO was more successful in preserving its performance, and therefore surpassed CTLBO.

6.2.1.12. Evaluation and Analysis of Algorithms for HE3 Problem

HE3 is a nonlinear two-objective problem of type III. Table 14 presents the results obtained for experiments on this problem.

Table 14: Performance of the algorithms for HE3 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$ $n_t = 10$ $\tau_t = 500$	IGD	mean	2.30e-4	3.94e-4	5.13e-4	6.98e-4	**1.98e-4**	2.21e-4
			std. r.	1.1e-5 (2.7)	2.4e-5 (4.0)	1.7e-5 (5.0)	2.2e-5 (6.0)	**8.7e-6 (1.0)**	6.6e-6 (2.3)
		HVR	mean	91.81%	92.11%	81.96%	85.86%	**94.02%**	93.25%
			std. r.	0.48% (3.7)	0.39% (3.3)	0.55% (6.0)	0.40% (5.0)	**0.36% (1.1)**	0.32% (1.9)
2	$n = 10$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	2.26e-4	3.86e-4	5.00e-4	3.45e-4	**1.78e-4**	2.01e-4
			std. r.	9.7e-6 (3.0)	6.8e-6 (5.0)	2.5e-5 (6.0)	5.8e-6 (4.0)	**3.7e-6 (1.0)**	5.1e-6 (2.0)
		HVR	mean	92.08%	92.39%	82.70%	87.80%	**94.81%**	94.14%
			std. r.	0.35% (3.8)	0.40% (3.2)	0.94% (6.0)	0.20% (5.0)	**0.15% (1.0)**	0.15% (2.0)
3	$n = 10$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	3.69e-4	3.68e-4	5.73e-4	5.56e-4	**1.70e-4**	1.85e-4
			std. r.	9.6e-6 (3.8)	2.2e-6 (3.3)	2.7e-5 (5.8)	8.8e-6 (5.3)	**2.7e-6 (1.0)**	3.3e-6 (2.0)
		HVR	mean	92.78%	92.43%	81.04%	88.98%	**95.20%**	94.74%
			std. r.	0.22% (3.1)	0.33% (3.9)	0.68% (6.0)	0.15% (5.0)	**0.09% (1.0)**	0.07% (2.0)
4	$n = 10$ $n_t = 2$ $\tau_t = 500$	IGD	mean	4.07e-4	3.99e-4	5.15e-4	3.9e-4	**1.96e-4**	2.22e-4
			std. r.	2.3e-5 (4.4)	4.8e-6 (4.3)	1.8e-5 (6.0)	8.7e-6 (3.3)	**8.2e-6 (1.0)**	9.0e-6 (2.0)
		HVR	mean	91.92%	92.08%	82.16%	86.18%	**94.10%**	93.27%
			std. r.	0.63% (3.6)	0.36% (3.4)	0.83% (6.0)	0.35% (5.0)	**0.39% (1.0)**	0.31% (2.0)
5	$n = 10$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	4.05e-4	3.91e-4	5.03e-4	3.44e-4	**1.79e-4**	2.04e-4
			std. r.	2.9e-5 (4.6)	4.7e-6 (4.4)	2.1e-5 (6.0)	6.9e-6 (3.0)	**2.9e-6 (1.0)**	6.6e-6 (2.0)
		HVR	mean	91.93%	92.47%	82.79%	87.95%	**94.83%**	93.99%
			std. r.	0.66% (3.8)	0.50% (3.2)	0.65% (6.0)	0.22% (5.0)	**0.13% (1.0)**	0.26% (2.0)
6	$n = 10$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	3.88e-4	3.98e-4	5.80e-4	5.60e-4	**1.70e-4**	1.87e-4
			std. r.	1.7e-5 (3.3)	2.9e-6 (3.7)	3.7e-5 (5.9)	1.0e-5 (5.1)	**1.9e-6 (1.0)**	2.8e-6 (2.0)
		HVR	mean	92.43%	92.23%	81.11%	88.96%	**95.21%**	94.65%
			std. r.	0.45% (3.3)	0.23% (3.7)	1.03% (6.0)	0.21% (5.1)	**0.07% (1.0)**	0.10% (2.0)
7	$n = 30$ $n_t = 10$ $\tau_t = 500$	IGD	mean	5.05e-4	4.36e-4	7.13e-4	8.97e-4	**2.61e-4**	**2.69e-4**
			std. r.	2.8e-5 (3.9)	2.2e-5 (3.1)	1.9e-5 (5.0)	1.7e-5 (6.0)	**2.3e-5 (1.5)**	**1.8e-5 (1.5)**
		HVR	mean	89.26%	89.33%	73.49%	81.57%	90.65%	**90.47%**
			std. r.	0.69% (3.4)	0.11% (2.8)	0.79% (6.0)	0.42% (5.0)	1.07% (2.1)	**0.88% (1.7)**
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	4.97e-4	4.30e-4	6.59e-4	4.49e-4	**2.19e-4**	2.35e-4
			std. r.	2.4e-5 (5.0)	1.4e-5 (3.2)	3.2e-5 (6.0)	9.9e-6 (3.8)	**3.0e-5 (1.1)**	5.9e-6 (1.9)
		HVR	mean	89.43%	89.57%	75.89%	83.72%	**92.53%**	91.96%
			std. r.	0.58% (3.5)	0.10% (3.4)	1.06% (6.0)	0.32% (5.0)	**1.17% (1.4)**	0.27% (1.7)
9	$n = 30$	IGD	mean	2.78e-4	2.87e-4	6.93e-4	4.06e-4	**1.99e-4**	2.15e-4

	$n_t = 10$ $\tau_t = 2000$		std. r.	2.6e-5 (3.4)	1.4e-5 (3.6)	3.8e-5 (6.0)	7.5e-6 (5.0)	**1.2e-5 (1.1)**	3.7e-6 (1.9)
		HVR	mean	89.71%	89.29%	75.33%	85.13%	**93.40%**	92.88%
			std. r.	1.02% (3.4)	0.55% (3.6)	1.33% (6.0)	0.30% (5.0)	**0.58% (1.3)**	0.16% (1.8)
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	2.90e-4	4.23e-4	7.20e-4	5.07e-4	2.68e-4	**2.57e-4**
			std. r.	2.8e-5 (2.4)	1.3e-5 (4.0)	2.9e-5 (6.0)	1.24e-5 (5.0)	2.4e-5 (1.9)	**5.6e-6 (1.7)**
		HVR	mean	89.72%	89.09%	73.95%	81.87%	90.57%	**91.27%**
			std. r.	0.71% (3.3)	1.14% (3.3)	0.95% (6.0)	0.47% (5.0)	1.03% (2.0)	**0.18% (1.4)**
11	$n = 30$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	5.05e-4	4.29e-4	6.64e-4	4.45e-4	**2.18e-4**	2.34e-4
			std. r.	4.9e-5 (4.9)	1.4e-5 (3.2)	2.9e-5 (6.0)	9.3e-6 (3.9)	**1.8e-5 (1.1)**	8.2e-6 (1.9)
		HVR	mean	89.36%	89.68%	75.92%	84.02%	**92.58%**	92.17%
			std. r.	1.01% (3.7)	0.86% (3.3)	1.30% (6.0)	0.33% (5.0)	**0.87% (1.2)**	0.31% (1.8)
12	$n = 30$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	5.16e-4	4.16e-4	6.95e-4	4.06e-4	**2.01e-4**	2.19e-4
			std. r.	5.8e-5 (5.0)	1.9e-5 (3.7)	3.5e-5 (6.0)	7.9e-6 (3.3)	**1.5e-5 (1.3)**	7.9e-6 (1.7)
		HVR	mean	89.13%	90.40%	75.69%	85.29%	**93.35%**	92.76%
			std. r.	1.18% (3.8)	0.12% (3.2)	1.19% (6.0)	0.34% (5.0)	**0.64% (1.3)**	0.34% (1.7)

The results presented in Table 14 show that the CTLBO and VECTLBO algorithms outperform the other algorithms in solving the HE3 problem. In most cases, CTLBO is the best algorithm and VECTLBO obtains the second rank, but in cases where the dimension number and the frequency were higher (Experiments 7 and 10), VECTLBO showed better convergence and achieved the first rank.

6.2.1.13. Evaluation and Analysis of Algorithms for HE4 Problem

In this section, algorithms are evaluated using the HE4 problem which is a type III problem with nonlinear linkages between its variables. The results of experiments for this problem are presented in Table 15.

Table 15: Performance of the algorithms for HE4 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$ $n_t = 10$ $\tau_t = 500$	IGD	mean	8.46e-4	9.87e-4	**7.56e-4**	1.55e-3	8.37e-4	9.81e-4
			std. r.	2.5e-4 (3.3)	5.5e-4 (2.9)	**1.9e-4 (2.4)**	7.2e-5 (5.9)	6.6e-5 (2.6)	8.5e-5 (4.0)
		HVR	mean	77.74%	77.99%	**79.43%**	70.05%	73.67%	69.15%
			std. r.	4.43% (2.4)	5.47% (2.5)	**1.50% (1.8)**	1.33% (5.4)	1.48% (3.5)	2.57% (5.5)
2	$n = 10$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	**5.45e-4**	7.41e-4	6.76e-4	1.30e-3	7.75e-4	8.58e-4
			std. r.	**1.2e-4 (1.3)**	2.8e-4 (2.9)	4.1e-5 (2.5)	1.4e-4 (6.0)	4.0e-5 (3.9)	5.8e-5 (4.5)
		HVR	mean	**84.34%**	81.47%	79.39%	76.03%	76.08%	73.55%
			std. r.	**3.40% (1.1)**	5.54% (2.4)	2.13% (3.0)	1.24% (4.6)	2.30% (4.4)	2.03% (5.5)
3	$n = 10$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	**5.79e-4**	1.25e-3	7.35e-4	6.81e-4	7.60e-4	8.34e-4
			std. r.	**1.1e-4 (1.4)**	7.6e-4 (5.7)	2.9e-5 (3.3)	4.1e-5 (2.0)	5.4e-5 (3.4)	5.9e-5 (5.1)
		HVR	mean	83.75%	**83.55%**	77.50%	78.48%	77.63%	74.70%
			std. r.	2.48% (1.9)	**4.12% (1.6)**	1.61% (4.1)	0.77% (3.6)	2.28% (4.0)	1.97% (5.9)
4	$n = 10$	IGD	mean	7.07e-4	**6.25e-4**	8.86e-4	9.25e-4	8.51e-4	1.01e-3
			std. r.	1.7e-4 (2.3)	**1.5e-4 (1.7)**	3.2e-4 (3.2)	1.2e-4 (4.7)	4.8e-5 (3.6)	9.1e-5 (5.5)

	$n_t = 2$ $\tau_t = 500$	HVR	mean	80.82%	**83.34%**	79.72%	71.02%	73.94%	69.24%
			std. r.	3.74% (2.0)	**2.55% (1.6)**	1.33% (2.4)	1.83% (5.3)	2.39% (4.3)	3.49% (5.4)
5	$n = 10$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	**5.75e-4**	6.84e-4	7.18e-4	7.43e-4	7.94e-4	9.24e-4
			std. r.	**1.4e-4 (2.0)**	3.6e-4 (2.3)	9.2e-5 (2.9)	8.3e-5 (4.0)	4.0e-5 (4.4)	1.3e-4 (5.5)
		HVR	mean	83.41%	**82.95%**	81.06%	75.87%	76.26%	72.73%
			std. r.	3.18% (1.9)	**3.69% (1.8)**	0.81% (2.4)	1.61% (4.6)	1.62% (4.5)	2.53% (5.9)
6	$n = 10$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	**5.14e-4**	1.21e-3	8.47e-4	6.53e-4	7.48e-4	8.12e-4
			std. r.	**9.5e-5 (1.1)**	4.3e-4 (5.6)	1.5e-4 (4.0)	9.4e-5 (2.3)	3.4e-5 (3.3)	8.1e-5 (4.7)
		HVR	mean	**85.96%**	84.67%	76.78%	79.02%	78.34%	74.61%
			std. r.	**1.63% (1.4)**	3.56% (1.7)	2.47% (5.0)	1.70% (3.3)	1.58% (3.7)	2.58% (5.9)
7	$n = 30$ $n_t = 10$ $\tau_t = 500$	IGD	mean	1.23e-3	**1.08e-3**	1.12e-3	1.42e-3	**1.08e-3**	1.62e-3
			std. r.	6.8e-4 (2.7)	**2.5e-4 (2.6)**	1.8e-4 (2.9)	6.2e-5 (4.8)	**4.4e-5 (2.6)**	1.3e-4 (5.5)
		HVR	mean	**72.68%**	**72.21%**	71.23%	50.23%	63.69%	46.78%
			std. r.	**5.80% (2.0)**	**6.91% (2.0)**	1.57% (2.3)	2.06% (5.5)	3.04% (3.7)	4.59% (5.5)
8	$n = 30$ $n_t = 10$ $\tau_t = 1000$	IGD	mean	9.24e-4	**8.67e-4**	9.43e-4	1.12e-3	9.30e-4	1.32e-3
			std. r.	2.0e-4 (2.6)	**1.2e-4 (2.3)**	1.4e-4 (2.9)	8.0e-5 (4.8)	8.3e-5 (2.6)	1.8e-4 (5.9)
		HVR	mean	**78.26%**	**77.92%**	74.04%	61.12%	69.99%	55.14%
			std. r.	**2.39% (1.6)**	**3.69% (1.6)**	1.97% (2.9)	2.33% (5.1)	3.21% (3.9)	5.69% (5.9)
9	$n = 30$ $n_t = 10$ $\tau_t = 2000$	IGD	mean	1.04e-3	9.65e-4	1.26e-3	1.15e-3	**8.32e-4**	1.10e-3
			std. r.	5.6e-4 (3.2)	6.0e-4 (3.4)	2.1e-4 (4.8)	1.6e-4 (4.4)	**7.4e-5 (2.0)**	1.8e-4 (4.3)
		HVR	mean	77.01%	**76.98%**	73.29%	67.21%	74.97%	62.96%
			std. r.	4.32% (2.1)	**5.88% (1.8)**	1.21% (3.6)	2.01% (5.1)	4.08% (2.8)	6.55% (5.6)
10	$n = 30$ $n_t = 2$ $\tau_t = 500$	IGD	mean	1.17e-3	**9.10e-4**	1.13e-3	1.45e-3	1.10e-3	1.62e-3
			std. r.	3.4e-4 (3.3)	**1.3e-4 (1.4)**	2.1e-4 (2.8)	1.0e-4 (4.8)	9.6e-5 (3.3)	2.2e-4 (5.6)
		HVR	mean	73.01%	**77.58%**	71.13%	51.17%	63.75%	46.58%
			std. r.	3.31% (2.1)	**1.82% (1.1)**	4.22% (2.9)	2.29% (5.1)	4.25% (3.9)	6.70% (5.9)
11	$n = 30$ $n_t = 2$ $\tau_t = 1000$	IGD	mean	1.01e-3	1.50e-3	1.04e-3	1.16e-3	**9.07e-4**	1.29e-3
			std. r.	2.2e-4 (2.6)	1.3e-3 (3.6)	2.6e-4 (2.9)	7.1e-5 (4.4)	**3.4e-5 (2.0)**	6.6e-5 (5.5)
		HVR	mean	**76.76%**	70.24%	75.08%	60.72%	72.56%	58.00%
			std. r.	**3.53% (1.9)**	14.38% (2.5)	1.25% (2.1)	2.19% (5.0)	1.32% (3.8)	4.37% (5.8)
12	$n = 30$ $n_t = 2$ $\tau_t = 2000$	IGD	mean	9.40e-4	9.71e-4	1.02e-3	1.17e-3	**8.25e-4**	1.11e-3
			std. r.	3.3e-4 (3.0)	6.5e-4 (2.8)	3.1e-4 (3.4)	3.0e-4 (5.0)	**2.7e-5 (2.3)**	1.6e-4 (4.6)
		HVR	mean	78.10%	**78.00%**	72.79%	69.16%	76.07%	64.92%
			std. r.	5.22% (2.1)	**5.35% (1.9)**	2.79% (4.0)	2.68% (4.9)	1.03% (2.4)	5.27% (5.8)

The results obtained for the HE4 problem indicate that, in most cases, DNSGA-II B and DNSGA-II A are the superior algorithms, with each of them achieving the first or second rank. In cases where the frequency of changes is high, CSADMO achieves results close to those of the best algorithm, and in cases where the dimension number is lower and the severity of changes is smaller, it becomes the best algorithm. Moreover, when the number of dimensions is increased, CTLBO is less affected and, according to the IGD criterion, it achieves the best results (Experiments 7, 9, 11 and 12) or results close to the best results.

6.2.1.14. Evaluation and Analysis of Algorithms for HE5 Problem

HE5 is another type III problem used for evaluating the algorithms in the current research. The results of the experiments for this problem are presented in Table 16.

Table 16: Performance of the algorithms for HE5 problem and Friedman test ranking

#	Configuration	Measures		DNSGA-II A	DNSGA-II B	CSADMO	PNSCCDMO	CTLBO	VECTLBO
1	$n = 10$, $n_t = 10$, $\tau_t = 500$	IGD	mean	**4.79e-4**	5.43e-4	6.37e-4	9.05e-4	5.71e-4	7.11e-4
			std. r.	**6.9e-5 (1.5)**	6.4e-5 (2.4)	3.1e-5 (4.0)	2.8e-5 (5.9)	3.5e-5 (2.8)	1.1e-4 (4.4)
		HVR	mean	83.48%	**84.33%**	78.08%	67.05%	80.86%	76.81%
			std. r.	1.62% (1.9)	**1.77% (1.6)**	1.11% (4.4)	1.03% (6.0)	1.21% (2.9)	3.06% (4.2)
2	$n = 10$, $n_t = 10$, $\tau_t = 1000$	IGD	mean	**4.55e-4**	5.38e-4	6.19e-4	6.74e-4	5.59e-4	6.18e-4
			std. r.	**1.0e-4 (2.1)**	8.0e-5 (2.4)	2.2e-5 (4.3)	3.1e-5 (5.4)	6.1e-5 (3.0)	1.8e-5 (3.8)
		HVR	mean	84.19%	**84.97%**	78.52%	75.27%	82.11%	80.04%
			std. r.	2.80% (2.1)	**2.42% (2.0)**	0.82% (4.9)	1.03% (6.0)	1.34% (2.4)	1.06% (3.6)
3	$n = 10$, $n_t = 10$, $\tau_t = 2000$	IGD	mean	**3.80e-4**	4.40e-4	6.97e-4	5.86e-4	5.86e-4	5.85e-4
			std. r.	**6.8e-5 (1.3)**	4.7e-5 (1.9)	2.0e-5 (6.0)	2.3e-5 (4.3)	6.7e-5 (3.9)	5.7e-5 (3.7)
		HVR	mean	**86.63%**	86.72%	75.80%	78.78%	81.83%	81.89%
			std. r.	**1.95% (1.4)**	1.65% (1.7)	0.65% (6.0)	0.77% (4.9)	1.83% (3.6)	2.29% (3.4)
4	$n = 10$, $n_t = 2$, $\tau_t = 500$	IGD	mean	**4.49e-4**	5.13e-4	6.30e-4	8.88e-4	6.16e-4	6.71e-4
			std. r.	**8.3e-5 (1.5)**	3.0e-5 (1.7)	2.7e-5 (4.1)	2.1e-5 (6.0)	5.1e-5 (3.2)	3.7e-5 (4.5)
		HVR	mean	84.35%	**85.10%**	78.26%	67.91%	80.24%	78.28%
			std. r.	2.43% (1.9)	**1.04% (1.3)**	0.94% (4.5)	0.46% (6.0)	1.31% (3.0)	1.53% (4.3)
5	$n = 10$, $n_t = 2$, $\tau_t = 1000$	IGD	mean	**4.00e-4**	5.23e-4	6.13e-4	6.89e-4	5.26e-4	5.87e-4
			std. r.	**5.8e-5 (1.1)**	5.0e-5 (2.9)	3.0e-5 (4.6)	2.5e-5 (6.0)	5.1e-5 (2.6)	5.1e-5 (3.8)
		HVR	mean	**86.11%**	84.77%	78.96%	75.14%	83.01%	81.15%
			std. r.	**1.52% (1.6)**	2.18% (2.1)	1.12% (4.9)	0.81% (6.0)	1.06% (2.7)	1.27% (3.7)
6	$n = 10$, $n_t = 2$, $\tau_t = 2000$	IGD	mean	**3.70e-4**	4.78e-4	6.87e-4	5.68e-4	5.37e-4	5.98e-4
			std. r.	**4.8e-5 (1.1)**	6.8e-5 (2.3)	4.6e-5 (6.0)	2.3e-5 (3.4)	6.4e-5 (3.3)	3.4e-5 (4.9)
		HVR	mean	**86.97%**	86.07%	76.48%	79.50%	83.47%	81.63%
			std. r.	**1.34% (1.3)**	2.39% (2.0)	1.42% (6.0)	0.69% (5.0)	1.52% (2.7)	1.38% (4.0)
7	$n = 30$, $n_t = 10$, $\tau_t = 500$	IGD	mean	**6.60e-4**	6.85e-4	9.48e-4	1.73e-3	8.69e-4	9.01e-4
			std. r.	**1.0e-4 (1.3)**	5.7e-5 (1.7)	6.0e-5 (4.7)	3.8e-5 (6.0)	8.0e-5 (3.5)	4.0e-5 (3.8)
		HVR	mean	75.46%	**76.34%**	66.85%	40.42%	68.11%	66.55%
			std. r.	2.94% (1.6)	**2.47% (1.4)**	2.24% (4.2)	1.06% (6.0)	2.55% (3.7)	1.45% (4.1)
8	$n = 30$, $n_t = 10$, $\tau_t = 1000$	IGD	mean	6.05e-4	**6.17e-4**	8.70e-4	1.28e-3	7.74e-4	7.94e-4
			std. r.	7.3e-5 (1.7)	**4.9e-5 (1.3)**	5.2e-5 (4.8)	4.2e-5 (6.0)	4.0e-5 (3.6)	4.5e-5 (3.6)
		HVR	mean	77.44%	**79.28%**	69.32%	53.65%	71.77%	70.60%
			std. r.	2.36% (1.8)	**2.05% (1.2)**	1.83% (4.6)	1.44% (6.0)	1.43% (3.5)	1.50% (3.9)
9	$n = 30$, $n_t = 10$, $\tau_t = 2000$	IGD	mean	**5.28e-4**	5.64e-4	8.78e-4	9.82e-4	6.75e-4	7.36e-4
			std. r.	**5.0e-5 (1.4)**	9.4e-5 (1.9)	5.7e-5 (5.1)	3.1e-5 (5.9)	3.7e-5 (3.0)	5.2e-5 (3.8)
		HVR	mean	**80.29%**	79.37%	68.58%	63.63%	75.88%	73.80%
			std. r.	**1.69% (1.4)**	2.14% (1.9)	2.15% (4.9)	1.00% (6.0)	1.15% (3.0)	1.80% (3.9)
10	$n = 30$, $n_t = 2$, $\tau_t = 500$	IGD	mean	**6.04e-4**	6.65e-4	9.34e-4	1.67e-3	8.67e-4	8.97e-4
			std. r.	**5.4e-5 (1.2)**	6.7e-5 (1.8)	2.6e-5 (4.5)	4.8e-5 (6.0)	7.6e-5 (3.5)	7.5e-5 (4.0)
		HVR	mean	**77.48%**	76.79%	67.30%	43.15%	68.08%	67.37%
			std. r.	**1.78% (1.3)**	2.28% (1.7)	1.21% (4.3)	1.12% (6.0)	2.67% (3.8)	2.75% (3.9)
11	$n = 30$, $n_t = 2$, $\tau_t = 1000$	IGD	mean	5.58e-4	**5.57e-4**	8.66e-4	1.25e-3	7.67e-4	7.81e-4
			std. r.	4.8e-5 (1.6)	**5.0e-5 (1.4)**	4.2e-5 (5.0)	4.6e-5 (6.0)	2.7e-5 (3.3)	4.7e-5 (3.7)
		HVR	mean	**79.07%**	78.37%	69.64%	55.12%	72.61%	71.35%
			std. r.	**1.58% (1.4)**	1.94% (1.6)	2.03% (4.8)	1.58% (6.0)	1.05% (3.5)	2.02% (3.7)
12	$n = 30$, $n_t = 2$, $\tau_t = 2000$	IGD	mean	**4.98e-4**	5.23e-4	8.97e-4	9.79e-4	7.10e-4	7.45e-4
			std. r.	**2.8e-5 (1.3)**	4.8e-5 (1.7)	6.3e-5 (5.1)	2.6e-5 (5.9)	3.9e-5 (3.3)	6.1e-5 (3.7)
		HVR	mean	**81.36%**	79.91%	67.97%	63.92%	75.47%	73.84%
			std. r.	**0.96% (1.2)**	1.66% (1.8)	2.24% (5.1)	0.91% (5.9)	0.85% (3.3)	2.01% (3.7)

It can be seen from Table 16 that the DNSGA-II A and DNSGA-II B algorithms achieved first and second ranks in all the experiments for the HE5 problem. Among the other algorithms, CTLBO had the best performance and was given third rank in all the experiments. Moreover, in most of the experiments, VECTLBO outperformed the CSADMO and PNSCCDMO algorithms and achieved fourth place in the performance ranking.

6.2.2. Investigation of Convergence Trends and the Obtained Pareto Fronts

In this section, the convergence trends of the algorithms and the quality of the obtained Pareto front are illustrated by using IGD curves and Pareto front shapes for a number of problems, each representing one problem type, as shown in Figures 9 to 16. The IGD curves are the averages of IGD values over 20 runs on the problem, and the Pareto front in each environment is the Pareto front which had the nearest IGD value to the median IGD value of all Pareto fronts obtained in that environment over the 20 runs.

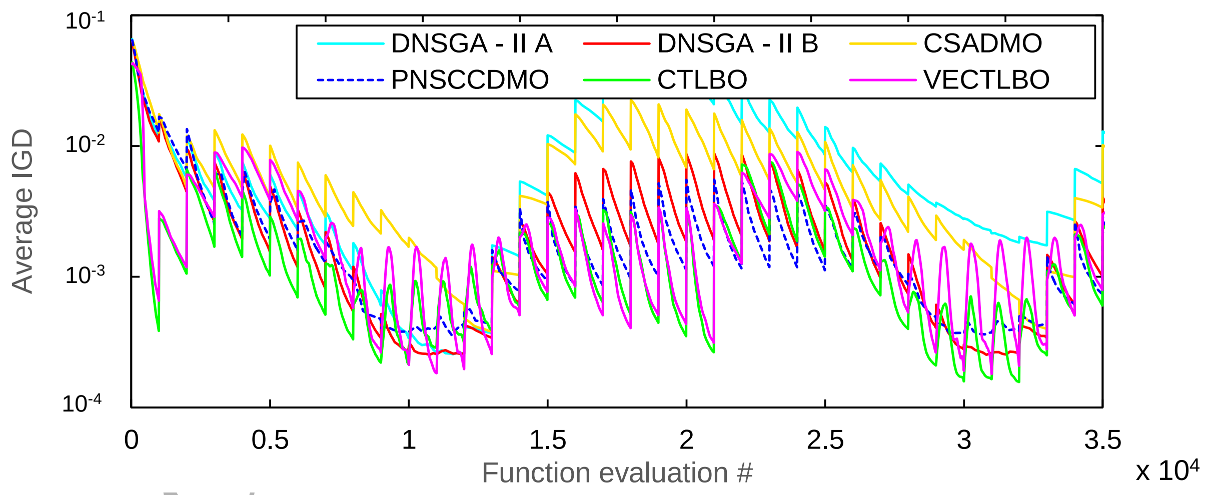


Figure 9: Average IGD values over 20 runs on FDA1(30-10-1000)

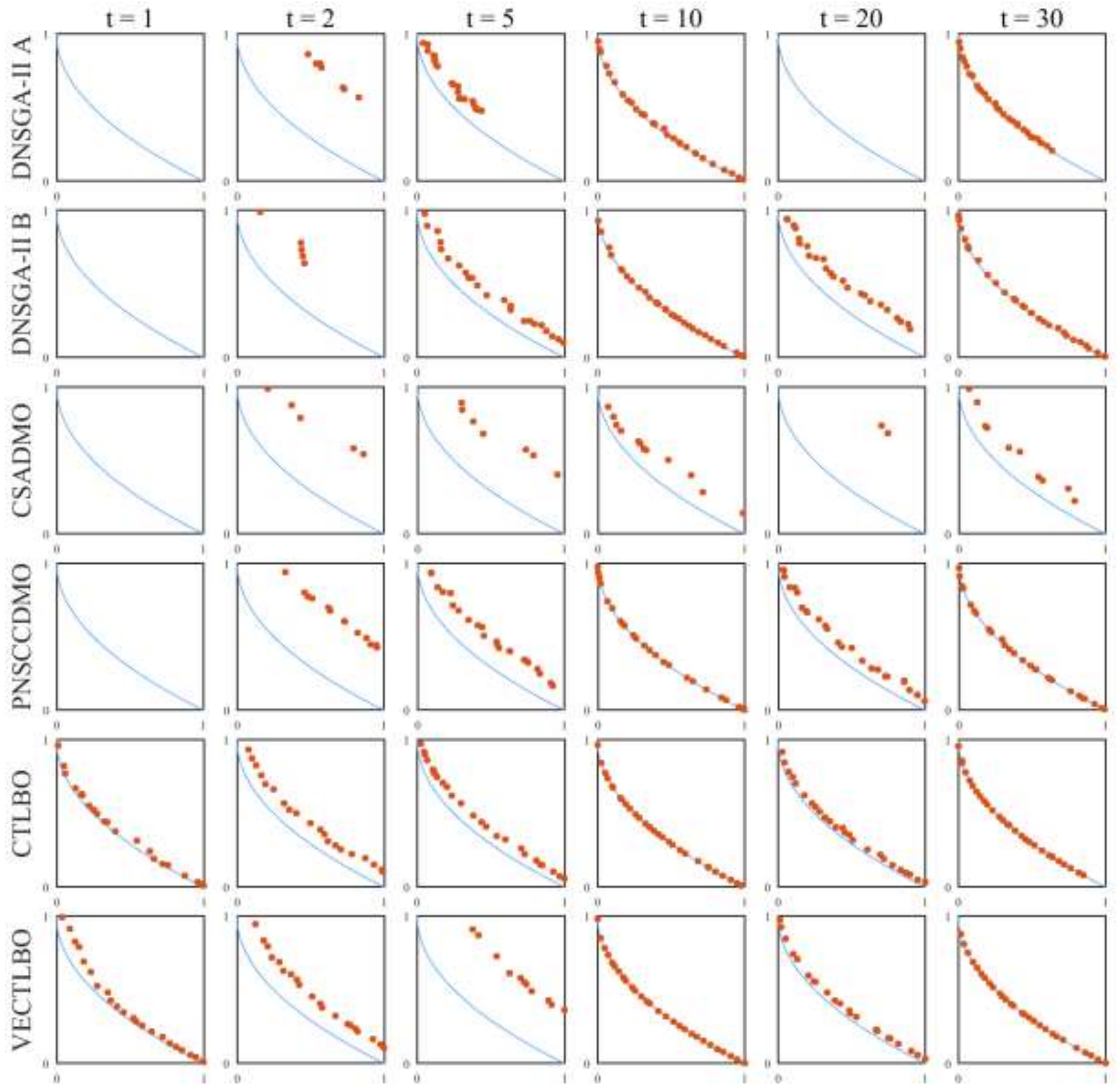


Figure 10: Obtained Pareto fronts for eight environments of FDA1(30-10-1000)

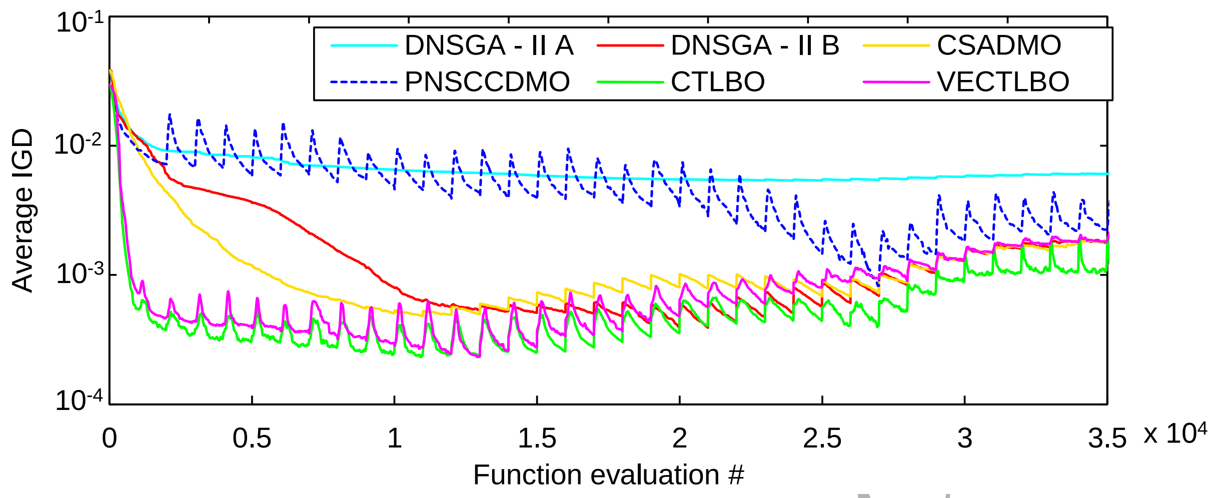


Figure 11: Average IGD values over 20 runs on FDA2$_{\text{Camara}}$(31-10-1000)

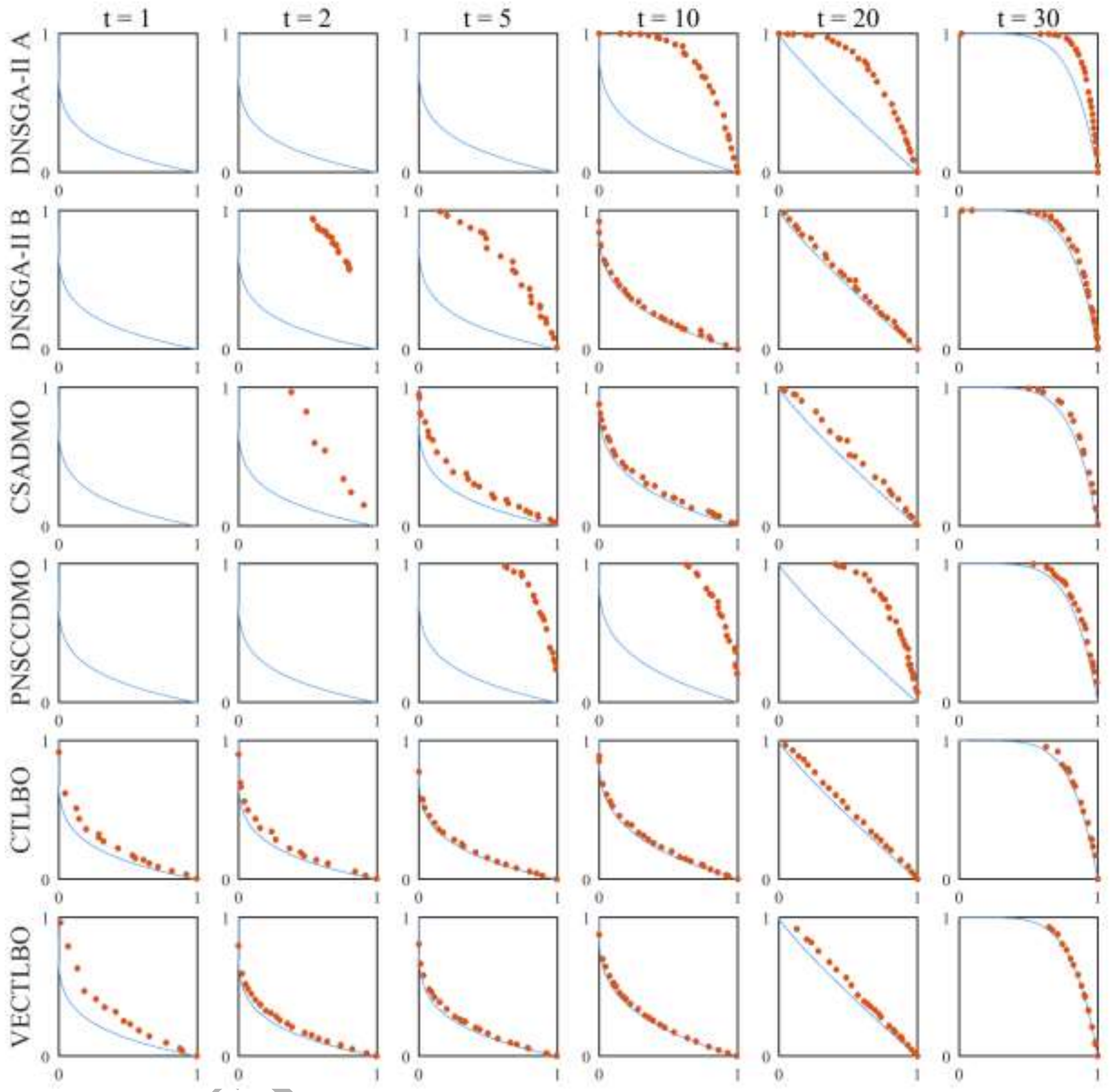


Figure 12: Obtained Pareto fronts for eight environments of FDA2$_{\text{Camara}}$(31-10-1000)

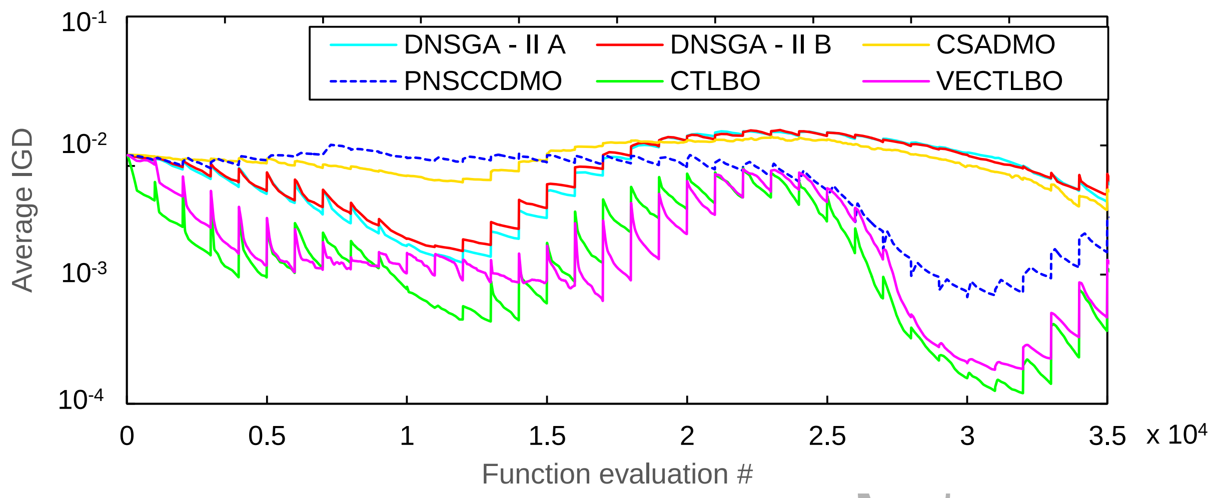


Figure 13: Average IGD values over 20 runs on ZJZ(30-10-1000)

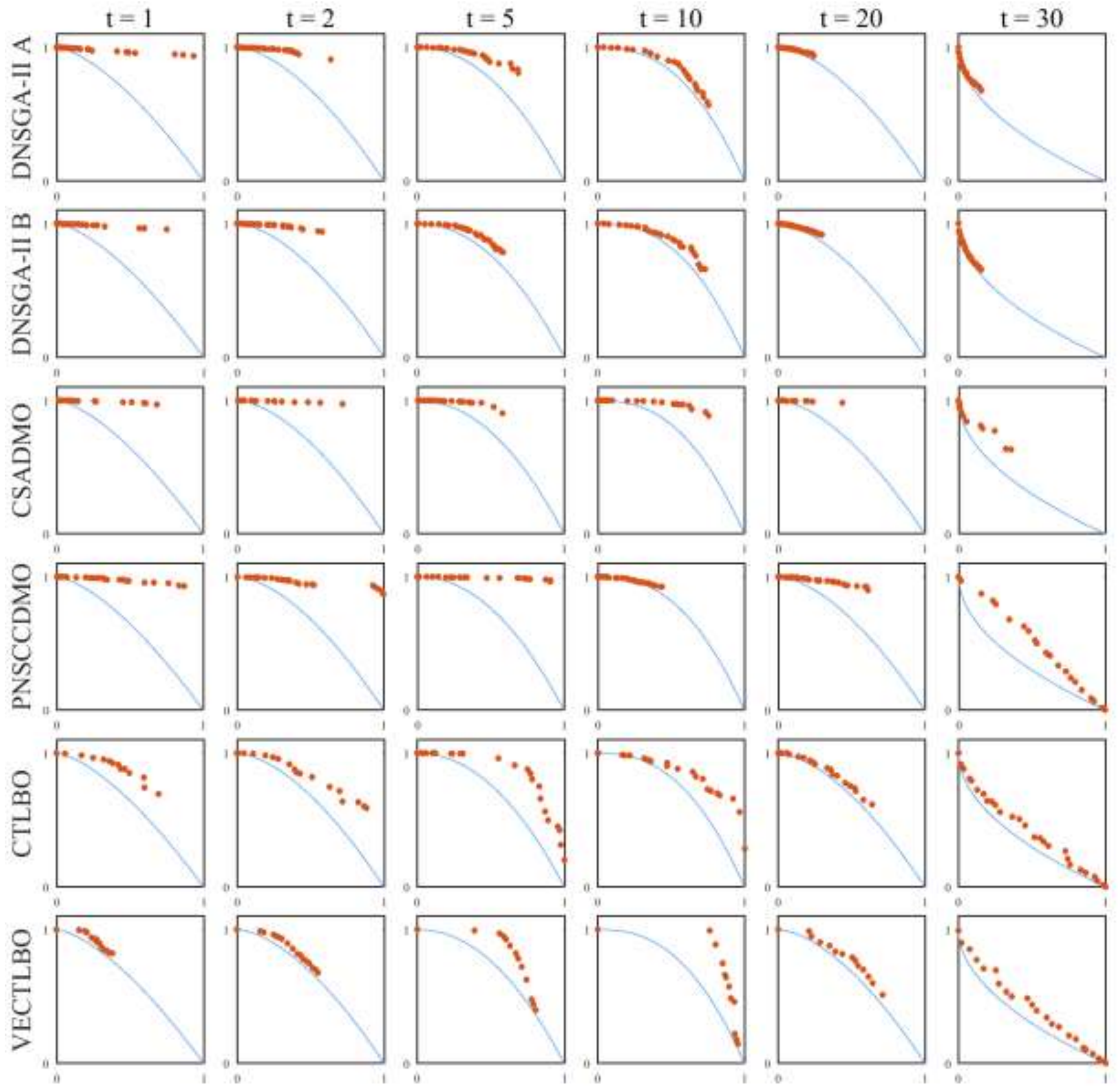


Figure 14: Obtained Pareto fronts for eight environments of ZJZ(30-10-1000)

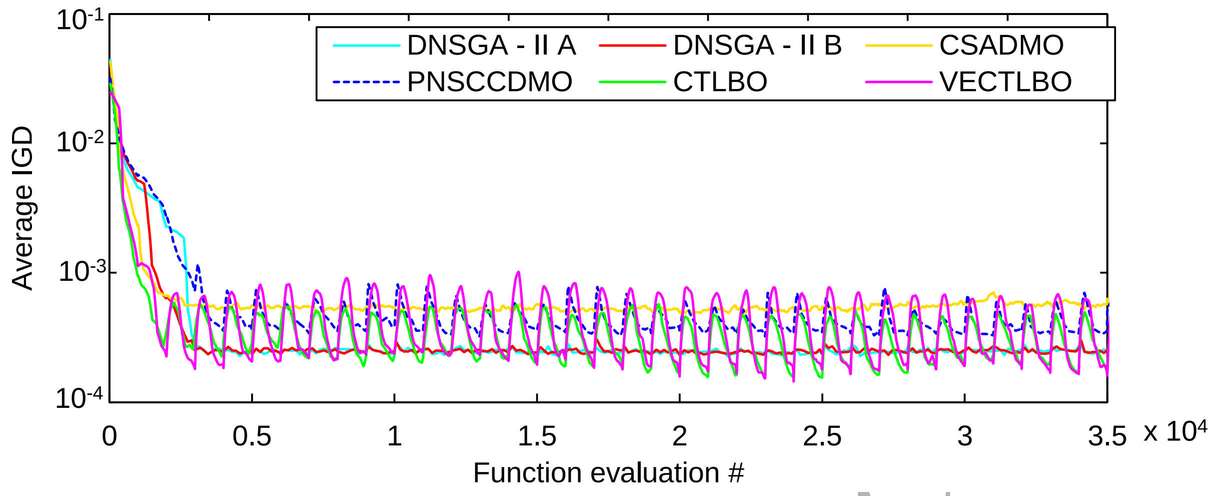


Figure 15: Average IGD values over 20 runs on dMOP1(10-10-1000)

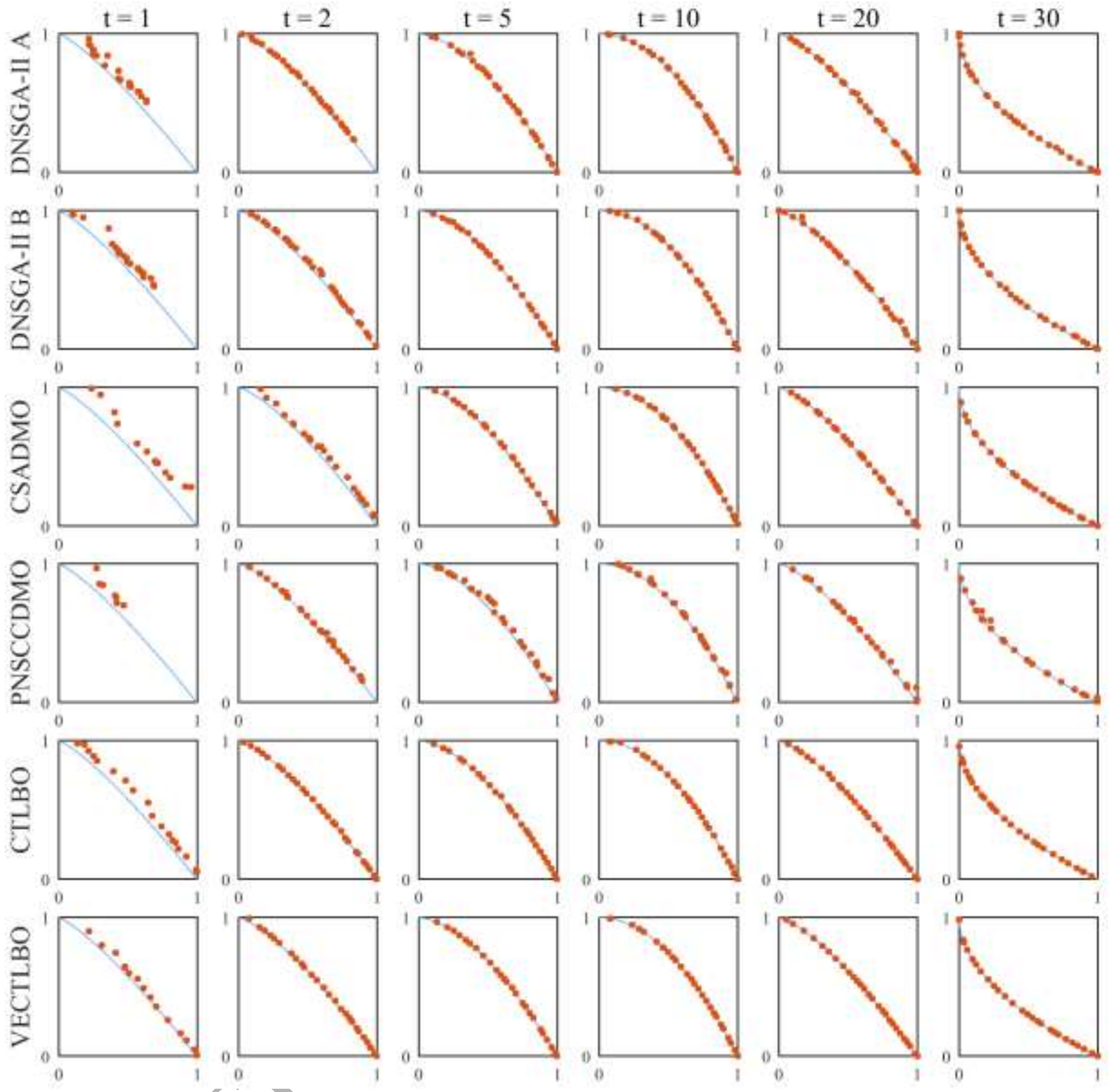


Figure 16: Obtained Pareto fronts for eight environments of dMOP1(10-10-1000)

As shown in Figure 9, the CTLBO and VECTLBO algorithms, in comparison with the other algorithms, converged faster to the POF in the first environments of the problem. In the subsequent environments, each of them converged faster in many cases and/or gave the best results according to the IGD criterion. This can also be deduced from Figure 10. As observed, the solutions obtained from these two algorithms are more favorable than those of the other algorithms in terms of convergence and uniformity of distribution.

It can be concluded from Figure 11, which displays the IGD curves for FDA2$_{\text{Camara}}$ problem, that the two algorithms CTLBO and VECTLBO converged faster to the POF in the first environments. In the subsequent environments, the CTLBO algorithm maintained its absolute superiority in most environments, and VECTLBO performed better than DNSGA-II B and CSADMO in some environments, though results were close. As also shown in Figure 12, the convergence and solution distributions of CTLBO and VECTLBO algorithms are more favorable than those of the other algorithms in most cases.

It is apparent from Figure 13 that in the experiment on the ZJZ problem, the CTLBO and VECTLBO algorithms outperform the other algorithms in all environments. Figure 14 also confirms this. These two algorithms converged better to the POF and distributed the solutions more uniformly.

The IGD curves for the dMOP1 problem presented in Figure 15, suggest a close competition between CTLBO and VECTLBO on the one hand and the DNSGA-II algorithms on the other. The DNSGA-II algorithms and the CSADMO algorithm transfer many of their solutions with no changes to the next environment. Thus, in solving problems of type III, they stay near their previous solution set, and have equal values in all environments. However, as can be seen from Figures 15 and 16, the two algorithms CTLBO and VECTLBO converged relatively more successfully in most environments, and achieved more uniform sets of solutions.

6.2.3. Investigation of Computational Complexity of Algorithms

From the computational complexity point of view, the most expensive components of the CTLBO and VECTLBO algorithms are “forming the cellular automata”, “growth of cellular automata”, “producing new solutions by TLBO algorithm” and “sorting the population based on closeness to the found Pareto front”. In the worst case, these have complexities of O(MN), O(MN^M), O$((M+D)N)$ and O$(N\log N)$ respectively, where M signifies the number of objectives, N the number of solutions in the population, and D the number of search space dimensions. Therefore, the complexities of these algorithms in the worst-case scenario are influenced by the growth stage of the automata and are of the O(MN^M) order of complexity, where N^M refers to the number of cells and M refers to $(M+1)$ neighboring cells whose information is used in cell processing. Although this complexity seems large compared to the complexities of the DNSGA-II A, DNSGA-II B, CSADMO and PNSCCDMO algorithms, which are of the order of O(MN^2) due to the use of “non-dominated sorting” or “extracting non-dominated solutions” stages, it should be noted that in calculating this complexity it is assumed that all cells are processed. However, in practice, using the technique referred to in section 5.1.4, automata growth stops when a certain condition is reached, and hence, many of the cells are not processed. Analysis of all the experiments performed on the studied benchmark problems in this article suggests that on average about 58% of the cells are activated and processed in each iteration. Moreover, it is not necessary for the number of cells in each dimension of the automata to be equal to the number in the population. The user can consider a smaller number for cells according to the expected number and distribution of the final non-dominated solutions. This results in a reduction in the complexity of the algorithms.

7. Conclusions

In the present study, a cellular automata-based approach was proposed for the evaluation and management of solutions in the solving procedure for dynamic multi-objective optimization problems. In addition to helping the algorithm identify non-dominated solutions and describing their closeness to the Pareto front, this approach provides an estimation of the hypervolume metric for non-dominant solutions. Moreover, in this approach, the density of candidate solutions in the objective space is managed in each iteration of the algorithm, and a diverse set of candidate solutions is extracted. Two dynamic multi-objective optimization algorithms were also introduced. In these algorithms, the TLBO optimization algorithm is used to optimize the solutions, and the cellular automata-based method is employed to manage and evaluate the solutions. The first algorithm, CTLBO, carries out optimization in a multi-objective manner, while the second algorithm, VECTLBO, utilizes a single-objective manner. To evaluate the proposed algorithms, these, along with some of the dynamic multi-objective optimization algorithms in the literature, were applied to some well-known benchmark problems and the results compared. Evaluation results are briefly summarized as follows:

- By investigating the results of all the experiments, it can be concluded that there is no clear relationship between the performance of the proposed algorithms and the problem type. However, their performance is more dependent on the definition of the problem. For problems of type I, CTLBO and VECTLBO achieve the best results for all experiments on FDA4 and F8 problems, but for FDA1 and dMOP3 problems, these algorithms achieved different ranks in different cases, though they remain superior algorithms in many of them. For problems of type II, both algorithms outperform the other algorithms in most experiments on $FDA2_{Camara}$, $FDA3_{Zheng}$ and ZJZ problems. They also gave the best results in some cases of the dMOP2 problem, but, for F5 and F6 problems, the results are poor and in some cases CTLBO converged better and achieved first rank. Regarding type III problems, it can be concluded that both algorithms outperform the other algorithms in solving the HE3 problem. Furthermore, each of them ranked first or second in most cases of the dMOP1 problem, but, for the HE4 and HE5 problems, VECTLBO obtained poor results and CTLBO, despite its poor results in some cases, showed some results close to those of the best algorithm, and ranked first in some cases.

- Regarding the FDA4 and F8 problems, which are problems with three objectives, it seems that CTLBO and VECTLBO are more successful on three-objective problems. The efficiency and superiority of CTLBO is remarkable in solving both problems. Although the efficiency of VECTLBO is not as good as CTLBO, and it loses second rank in few cases of the FDA4 problem, it still outperforms the other algorithms in many cases with remarkable differences.
- The results for linear and nonlinear problems indicate that CTLBO and VECTLBO algorithms had successes and failures in solving problems in both groups. Among the nonlinear problems, F8, HE3 and ZJZ are the problems for which CTLBO and VECTLBO outperform the other algorithms, and F6, F7, HE4 and HE5 are the problems for which these algorithms are less successful. Hence, there is no clear relationship between the nonlinearity feature and the efficiency of these algorithms.

Analysis indicated that the time complexities of the CTLBO and VECTLBO algorithms are higher than for the other algorithms, in theory due to the use of cellular automata. However, by using the proposed techniques for implementations, the processing of many cells is eliminated and the complexity is reduced significantly in practice. In addition, it is not necessary to choose the number of cells equal to the population size. The number of cells can be lower according to expectations for the number and distribution of solutions. Hence, the time complexity of such algorithms could be reduced by employing some techniques like the time complexity reduction techniques of cellular automata. On the other hand, these complexities may be neglected given the superiority of performance in such algorithms as well as growing trends in the development of processors.

There are some aspects of CTLBO and VECTLBO that could be further improved, and that could be discussed in future work. First, the experimental results of dMOP1, F4 and F5 show that the performance of the proposed algorithms degrades in terms of IGD when a change occurs in the environment. This issue can be alleviated by using memory. Furthermore, the time complexity of the proposed methods could be enormously improved using parallel processing for the CA cells. Finally, we would like to apply CTLBO and VECTLBO to real-world problems.

References

[1] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, “Multiobjective evolutionary algorithms: A survey of the state of the art,” *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32 – 49, 2011.

[2] Q. Zhang and H. Li, “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition,” *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[3] L. C. Jiao, H. Wang, R. H. Shang, and F. Liu, “A Co-evolutionary Multi-objective Optimization Algorithm Based on Direction Vectors,” *Inf Sci*, vol. 228, pp. 90–112, Apr. 2013.

[4] K. Deb and A. Kumar, “Interactive Evolutionary Multi-objective Optimization and Decision-making Using Reference Direction Method,” in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2007, pp. 781–788.

[5] E. Zitzler and L. Thiele, “Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study,” in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, London, UK, UK, 1998, pp. 292–304.

[6] E. Zitzler and S. Künzli, “Indicator-Based Selection in Multiobjective Search,” in *Parallel Problem Solving from Nature - PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings*, X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 832–842.

[7] A. Elhossini, S. Areibi, and R. Dony, “Strength Pareto Particle Swarm Optimization and Hybrid EA-PSO for Multi-Objective Optimization,” *Evol. Comput.*, vol. 18, no. 1, pp. 127–156, Mar. 2010.

[8] Q. Jiang *et al.*, “Multi-objective differential evolution with dynamic covariance matrix learning for multi-objective optimization problems with variable linkages,” *Knowl.-Based Syst.*, vol. 121, pp. 111 – 128, 2017.

[9] J. D. Knowles and D. W. Corne, “M-PAES: a memetic algorithm for multiobjective optimization,” in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, 2000, vol. 1, pp. 325–332 vol.1.

[10] R. Liu, Y. Chen, W. Ma, C. Mu, and L. Jiao, “A novel cooperative coevolutionary dynamic multi-objective optimization algorithm using a new predictive model,” *Soft Comput.*, vol. 18, no. 10, pp. 1913–1929, 2014.
[11] R. Shang, Y. Wang, J. Wang, L. Jiao, S. Wang, and L. Qi, “A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem,” *Inf. Sci.*, vol. 277, pp. 609 – 642, 2014.
[12] M. Greeff and A. Engelbrecht, “Solving dynamic multi-objective problems with vector evaluated particle swarm optimisation,” in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, 2008, pp. 2917–2924.
[13] X. Li, “Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function,” in *Genetic and Evolutionary Computation - GECCO 2004, Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004, Proceedings, Part I*, 2004, pp. 117–128.
[14] R. Shang, L. Jiao, F. Liu, and W. Ma, “A Novel Immune Clonal Algorithm for MO Problems,” *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 35–50, Feb. 2012.
[15] P. Amato and M. Farina, “An ALife-Inspired Evolutionary Algorithm for Dynamic Multiobjective Optimization Problems,” in *Soft Computing: Methodologies and Applications*, vol. 32, F. Hoffmann, M. Köppen, F. Klawonn, and R. Roy, Eds. Springer Berlin Heidelberg, 2005, pp. 113–125.
[16] M. Farina, K. Deb, and P. Amato, “Dynamic multiobjective optimization problems: test cases, approximations, and applications,” *Evol. Comput. IEEE Trans. On*, vol. 8, no. 5, pp. 425–442, Oct. 2004.
[17] M. Helbig and A. P. Engelbrecht, “Population-based metaheuristics for continuous boundary-constrained dynamic multi-objective optimisation problems,” *Swarm Evol. Comput.*, vol. 14, pp. 31 – 47, 2014.
[18] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization,” *Comput. J.*, vol. 7, pp. 308–313, 1965.
[19] Z. Avdagić, S. Konjicija, and S. Omanović, “Evolutionary Approach to Solving Non-stationary Dynamic Multi-Objective Problems,” in *Foundations of Computational Intelligence Volume 3*, vol. 203, A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, Eds. Springer Berlin Heidelberg, 2009, pp. 267–289.
[20] C. M. Fonseca and P. J. Fleming, “Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization,” in *Proceedings of the 5th International Conference on Genetic Algorithms*, San Francisco, CA, USA, 1993, pp. 416–423.
[21] K. Deb, U. Rao N., and S. Karthik, “Dynamic Multi-objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-thermal Power Scheduling,” in *Evolutionary Multi-Criterion Optimization*, vol. 4403, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds. Springer Berlin Heidelberg, 2007, pp. 803–817.
[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[23] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II,” in *Parallel Problem Solving from Nature PPSN VI*, vol. 1917, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 849–858.

[24] R. Shang, L. Jiao, M. Gong, and B. Lu, “Clonal Selection Algorithm for Dynamic Multiobjective Optimization,” in *Computational Intelligence and Security*, vol. 3801, Y. Hao, J. Liu, Y. Wang, Y. Cheung, H. Yin, L. Jiao, J. Ma, and Y.-C. Jiao, Eds. Springer Berlin Heidelberg, 2005, pp. 846–851.

[25] F. M. Burnet, *The clonal selection theory of acquired immunity*. Nashville,Vanderbilt University Press, 1959.

[26] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996.

[27] Y. Li, R. Xiang, L. Jiao, and R. Liu, “An improved cooperative quantum-behaved particle swarm optimization,” *Soft Comput.*, vol. 16, no. 6, pp. 1061–1069, 2012.

[28] Z. Zhang, “Multiobjective optimization immune algorithm in dynamic environments and its application to greenhouse control,” *Appl. Soft Comput.*, vol. 8, no. 2, pp. 959 – 971, 2008.

[29] S. Zeng *et al.*, “A Dynamic Multi-Objective Evolutionary Algorithm Based on an Orthogonal Design,” in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 573–580.

[30] A. H. Wright, “Genetic Algorithms for Real Parameter Optimization,” in *Foundations of Genetic Algorithms*, 1991, pp. 205–218.

[31] B. Zheng, “A New Dynamic Multi-objective Optimization Evolutionary Algorithm,” in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, 2007, vol. 5, pp. 565–570.

[32] K. C. Tan, Y. H. Chew, T. H. Lee, and Y. J. Yang, “A cooperative coevolutionary algorithm for multiobjective optimization.,” in *SMC*, 2003, pp. 390–395.

[33] C.-K. Goh and K. Chen Tan, “A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization,” *Evol. Comput. IEEE Trans. On*, vol. 13, no. 1, pp. 103–127, Feb. 2009.

[34] Y. Wang and B. Li, “Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization,” *Memetic Comput.*, vol. 2, no. 1, pp. 3–24, 2010.

[35] H. Chen, M. Li, and X. Chen, “Using Diversity as an Additional-objective in Dynamic Multi-objective Optimization Algorithms,” in *2009 Second International Symposium on Electronic Commerce and Security*, 2009, vol. 1, pp. 484–487.

[36] M. Liu and W. Zeng, “A fast evolutionary algorithm for dynamic bi-objective optimization problems,” in *Computer Science Education (ICCSE), 2012 7th International Conference on*, 2012, pp. 130–134.

[37] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[38] K. Li, S. Kwong, J. Cao, M. Li, J. Zheng, and R. Shen, “Achieving balance between proximity and diversity in multi-objective evolutionary algorithm,” *Inf. Sci.*, vol. 182, no. 1, pp. 220 – 242, 2012.

[39] C. A. Coello Coello and M. S. Lechuga, “MOPSO: a proposal for multiple objective particle swarm optimization,” in *Evolutionary Computation, 2002. CEC ’02. Proceedings of the 2002 Congress on*, 2002, vol. 2, pp. 1051–1056.

[40] M. S. Lechuga, “Multi-objective optimisation using sharing in swarm optimisation algorithms,” Jul-2009.

[41] K. E. Parsopoulos and M. N. Vrahatis, “Recent approaches to global optimization problems through Particle Swarm Optimization,” *Nat. Comput.*, vol. 1, no. 2–3, pp. 235–306, 2002.

[42] L. Huang, I. H. Suh, and A. Abraham, “Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants,” *Inf. Sci.*, vol. 181, no. 11, pp. 2370 – 2391, 2011.

[43] Y. Wang and C. Dang, “An evolutionary algorithm for dynamic multi-objective optimization,” *Appl. Math. Comput.*, vol. 205, no. 1, pp. 6 – 18, 2008.

[44] A. Isaacs, V. Puttige, T. Ray, W. Smith, and S. Anavatti, “Development of a memetic algorithm for Dynamic Multi-Objective Optimization and its applications for online neural network modeling of UAVs,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 548–554.

[45] C. Liu and Y. Wang, “New Evolutionary Algorithm for Dynamic Multiobjective Optimization Problems,” in *Advances in Natural Computation*, vol. 4221, L. Jiao, L. Wang, X. Gao, J. Liu, and F. Wu, Eds. Springer Berlin Heidelberg, 2006, pp. 889–892.

[46] C. Liu and Y. Wang, “Dynamic Multi-objective Optimization Evolutionary Algorithm,” in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, 2007, vol. 4, pp. 456–459.

[47] I. Hatzakis and D. Wallace, “Dynamic Multi-objective Optimization with Evolutionary Algorithms: A Forward-looking Approach,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2006, pp. 1201–1208.

[48] C. Liu, “New Dynamic Multiobjective Evolutionary Algorithm with Core Estimation of Distribution,” in *Electrical and Control Engineering (ICECE), 2010 International Conference on*, 2010, pp. 1345–1348.

[49] A. K. M. K. A. Talukder, M. Kirley, and R. Buyya, “A Pareto Following Variation Operator for Fast-converging Multiobjective Evolutionary Algorithms,” in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2008, pp. 721–728.

[50] A. K. M. Khaled, A. Talukder, and M. Kirley, “A Pareto following variation operator for evolutionary dynamic multi-objective optimization,” in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp. 2270–2277.

[51] C. K. Goh, Y. S. Ong, K. C. Tan, and E. J. Teoh, “An investigation on evolutionary gradient search for multi-objective optimization,” in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, 2008, pp. 3741–3746.

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

[53] A. Zhou, Y. Jin, and Q. Zhang, “A Population Prediction Strategy for Evolutionary Dynamic Multiobjective Optimization,” *Cybern. IEEE Trans. On*, vol. 44, no. 1, pp. 40–53, Jan. 2014.

[54] K. Deb and R. B. Agrawal, “Simulated Binary Crossover for Continuous Search Space,” *Complex Syst.*, vol. 9, pp. 115–148, 1995.

[55] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[56] R. V. Rao, V. J. Savsani, and D. P. Vakharia, “Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems,” *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303 – 315, 2011.

[57] S. C. Satapathy and A. Naik, “A Modified Teaching-Learning-Based Optimization (mTLBO) for Global Search,” *Recent Pat. Comput. Sci.*, vol. 6, no. 1, pp. 60–72, Apr. 2013.
[58] S. Wolfram, *A New Kind of Science*. Wolfram Media, 2002.
[59] K. Deb, M. Mohan, and S. Mishra, “Evaluating the &Epsi;-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions,” *Evol Comput*, vol. 13, no. 4, pp. 501–525, Dec. 2005.
[60] K.E. Parsopoulos, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, and M.N. Vrahatis, “Vector Evaluated Differential Evolution for Multiobjective Optimization,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2004)*, Portland, Oregon, 2004.
[61] M. Cámara, J. Ortega, and F. de Toro, “Approaching Dynamic Multi-Objective Optimization Problems by Using Parallel Evolutionary Algorithms,” in *Advances in Multi-Objective Nature Inspired Computing*, vol. 272, C. Coello Coello, C. Dhaenens, and L. Jourdan, Eds. Springer Berlin Heidelberg, 2010, pp. 63–86.
[62] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, “Prediction-Based Population Re-initialization for Evolutionary Dynamic Multi-objective Optimization,” in *Evolutionary Multi-Criterion Optimization: 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007. Proceedings*, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 832–846.
[63] M. Helbig and A. P. Engelbrecht, “Benchmarks for Dynamic Multi-objective Optimisation Algorithms,” *ACM Comput Surv*, vol. 46, no. 3, pp. 37:1–37:39, Jan. 2014.
[64] M. Sierra and C. Coello Coello, “Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ∈-Dominance,” in *Evolutionary Multi-Criterion Optimization*, vol. 3410, C. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds. Springer Berlin Heidelberg, 2005, pp. 505–519.
[65] D. A. Van Veldhuizen, “Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations,” Air Force Institute of Technology, Wright Patterson AFB, OH, USA, 1999.
[66] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[67] M. Friedman, “The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance,” *J. Am. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, 1937.