MINISTRY OF EDUCATION & TRAINING

LE QUY DON TECHNICAL UNIVERSITY

VU VAN TRUONG

ENHANCING THE EFFECTIVENESS OF CO-EVOLUTIONARY METHODS IN MULTI-OBJECTIVE OPTIMIZATION AND APPLYING TO DATA CLASSIFICATION PROBLEMS

DOCTORAL THESIS IN MATHEMATICS

HA NOI - 2023

MINISTRY OF EDUCATION & TRAINING

LE QUY DON TECHNICAL UNIVERSITY

VU VAN TRUONG

ENHANCING THE EFFECTIVENESS OF CO-EVOLUTIONARY METHODS IN MULTI-OBJECTIVE OPTIMIZATION AND APPLYING TO DATA CLASSIFICATION PROBLEMS

Specialization: Mathematical Foundation for Informatics
Specialization code: 9 46 01 10

DOCTORAL THESIS IN MATHEMATICS

SUPERVISORS
1. Assoc. Prof. Bui Thu Lam
2. Prof. Nguyen Trung Thanh

HA NOI - 2023

ORIGINALITY STATEMENT

I hereby declare that this thesis is my own work, with my knowledge and belief the thesis has no material previously published or written by others. Any contributions made to the research by colleagues, with people in our research team at Le Quy Don Technical University or elsewhere, during my candidature is clearly acknowledged.

I also declare that the intellectual content in this submission is the research results of my own work, except to the extent that assistance from others in conception or in style, presentation and linguistic expression is acknowledged.

Hanoi, May. 9th, 2023

Author

Vu Van Truong

ACKNOWLEDGEMENTS

This work would not have been possible without the support of my colleagues, friends, and mentors. Specifically, I would like to thank my advisors, Assoc. Prof. Bui Thu Lam and Prof. Nguyen Trung Thanh, for their excellent guidance and generous support throughout my Ph.D. course. I am very grateful to have their trust in my ability, and I have often benefited from their insight and advice.

Additionally, I would like to express my gratitude to the entire research team from the Department of Software Technology, the Department of Survey and Mapping, the Evolutionary Computation group of the Military Technical Academy, and the Operational Research group of Liverpool John Moores University for their insightful discussions and productive teamwork. I would especially like to extend my sincere gratitude to the administrators of the Military Technical Academy's Faculty of Information Technology and Institute of Techniques for Special Engineering for providing me with all the facilities I needed for my research and for their ongoing support. I'm delighted to be a part of a fun and successful research team with amiable, driven, and supportive coworkers who have served as a constant source of inspiration for me.

Finally, but not least, my gratitude is for my family members who support my studies with strong encouragement and sympathy. My deepest

love is for my parents, my wife, and my three little babies, Phuong Thao, Bich Ngoc, and Thanh Son, who are an endless source of inspiration and motivation for me to overcome all obstacles. Without their invaluable help, this work would have never been completed.

Author

Vu Van Truong

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Meaning
EA	Evolutionary Algorithm
GA	Genetic Algorithm
ES	Evolution Strategies
EP	Evolution Programming
GP	Genetic Programming
MOP	Multi-objective Optimization Problem
MOEA	Multi-objective Evolutionary Algorithm
POF	Pareto Optimal Front
POS	Pareto Optimal Set
SOO	Single-objective Optimization
SOP	Single-objective Optimization Problem
MOEA/D	Multiobjective Evolutionary Algorithm based on Decomposition
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
SPEA2	Strength Pareto Evolutionary Algorithm 2
MOEA/D	Multi-objective Evolutionary Algorithm Based on Decomposition
MOGA	Multi-objective Genetic Algorithm
MOPSO	Multi-objective Particle Swarm Optimization
DM	Decision Maker
GD	Generational Distance
IGD	Inverse Generational Distance
HYP	Hypervolume
RMS	Restricted mating selection mechanism
NBSM	The neighbor-based selection mechanism
EC	Evolutionary Computing
CoEA	Coevolutionary algorithm
HoF	Hall of Fame
CCEA	Cooperative Coevolutionary algorithms
AI	artificial intelligence
CCEA	Competitive coevolutionary algorithms
ML	Machine learning
SDM	Sequential decision making
SGD	Stochastic gradient descent
FS	Feature selection
IS	Instance selection
DPP	Dual-population Paradigm
DPPCP	The dual-population competitive co-evolutionary approach

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Problem statement

In real life, there are many practical problems in which often-conflicted objectives need to be optimized simultaneously, especially in machine learning, where we are seeking a model with the best performance in both accuracy and generalization measures. These problems are called multi-objective optimization problems (MOPs). Unlike single-objective optimization, where it has to find the best single solution, in multi-objective optimization (MOO), a set of optimal solutions (called Pareto-optimal solutions) will usually be selected. Obviously, finding the largest number of Pareto-optimal solutions possible from MOO is a vital but time-consuming task. Therefore, MOO tries to find a set of solutions that satisfy both criteria (Figure. 1): as close as possible to the Pareto-optimal front and as diverse as possible [107].

Maintaining a balance between diversity and convergence is a key concern in the field of multi-objective optimization. However, in the context of multi-objective optimization, this is a particularly challenging problem to solve. Each of these goals will typically have a certain priority with every algorithm. The algorithms will handle these two goals in a variety of ways, depending on how to balance them. Algorithms can be classified into two categories based on this criterion (Figure.2) single algorithms and hybrid algorithms (i.e., groups that combine many algorithms together)

Recently, the group of single multi-objective evolutionary algorithms

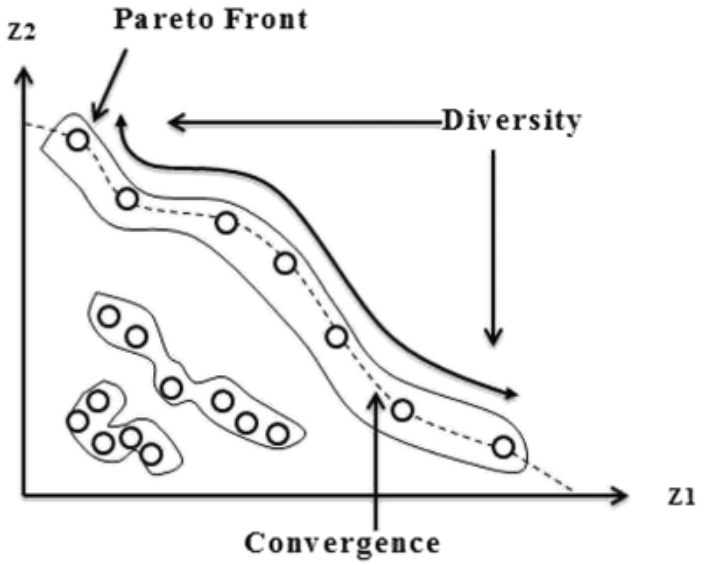


Figure 1: Illustrate two key concepts: diversity and convergence in Multi-objective optimization problems

(MOEAs) can be divided into three groups: Pareto-based algorithms ([30], [133]), indicator-based algorithms [132] and decomposition-based algorithms [130]. These MOEAs differ both in convergence and diversity preservation. The first group (i.e., Pareto-based algorithms) allocates priority to handling convergence, and the second one (i.e., the decomposition algorithm) focuses on diversity. Meanwhile, the last group (i.e., indicator-based algorithms) considers both convergence and diversity by using an indicator like hypervolume (HV). Typical indicator-based algorithms are IBEA (Indicator-based Evolutionary Algorithm; [132]); dynamic neighborhood MOEA based on HV indicator (DNMOEA/HI) [68]; an HV estimation algorithm (HypE) [6], and S-metric selection evolutionary multiobjective optimization algorithms (SMS-EMOA) [11]. These algorithms have the advantage that they do not require any additional diversity preservation mechanisms. However, when the number of objectives increases, the computational complexity of these algorithms also increases very quickly. This is their biggest weakness. This drawback has limited its application to solving multi- and many-objective problems.

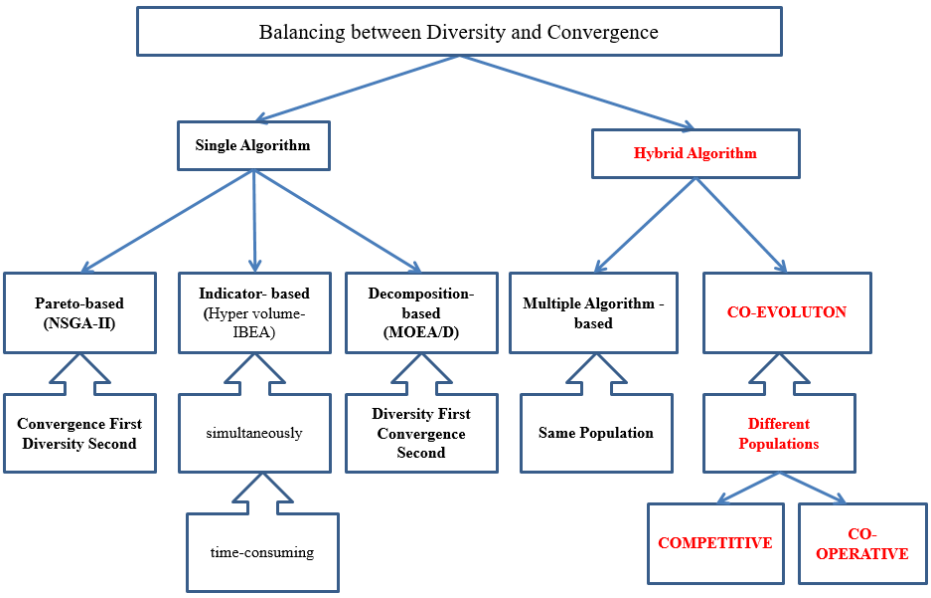


Figure 2: Division of multi-objective evolutionary algorithms based on the balance between diversity and convergence. The boxes with red text indicate the methods used in this study.

In general, using only a single algorithm to solve the problem of balancing convergence and diversity in MOPs is not easy. Therefore, the current trend is to combine multiple algorithms. This approach can be divided into two main groups: the multi-algorithm approach [121] (i.e., using multiple algorithms on the same population) and the multi-population approach [125] (i.e. using multiple populations, each of which corresponds to one objective). The multi-population approach can be regarded as a ***co-evolutionary algorithm (CoEA)***. The general idea of CoEA is to break down a problem into a set of sub-problems and use multiple populations to optimize different sub-problems. The CoEA can be categorized into two groups [127] which are competitive and cooperative. In the competitive approach, the fitness of each individual in one population is measured by their competition with others in other populations. With regard to the latter group, a collaborative mechanism is used to determine the fitness of each individual.

The ***diversity and accuracy*** (i.e., convergence) are also keys to ***ensemble learning methods*** and the importance of them was explained

in [33]. However, there is always a trade-off between classifier diversity and accuracy [106]. From this point, it can be seen that multi-objective evolutionary algorithms in general and ***co-evolutionary algorithms in particular are ideal for ensemble learning*** because they can identify a collection of solutions that ensure both convergence and diversity [100]. Instead of generating just one classifier, they force the training process to produce a set of diverse and optimal classifiers. An ensemble of classifiers can be created using Pareto-optimal solutions. Typically, a population-based approach is used to create candidate classifiers, and these classifiers are then improved using a multi-objective optimization strategy so that only Pareto-optimal solutions are kept [19]. The aforementioned strategy not only promotes the selection of the precise classifiers in the ensemble framework but also their distribution along the Pareto optimal front.

Beginning with the aforementioned issues, along with conducting theoretical research in the area of co-evolution, in this thesis, the author will concentrate on resolving two significant issues (Figure.3): first, proposing co-evolutionary algorithms for conventional multi-objective optimization issues (i.e., balancing diversity and convergence). Second, applying these co-evolutionary methods to machine learning issues (i.e., classification)

The next parts will provide a full presentation of the broad theory of co-evolution and the concept of solving practical challenges.

Motivation

Evolutionary algorithms (EAs) are regarded as effective algorithms for solving Pareto optimization problems because of their simplicity, capacity to operate in populations, and broad applicability. Multi-objective Evolutionary Algorithms (MOEAs) are currently one of the hottest topics in EAs research. MOEAs have undergone much research,

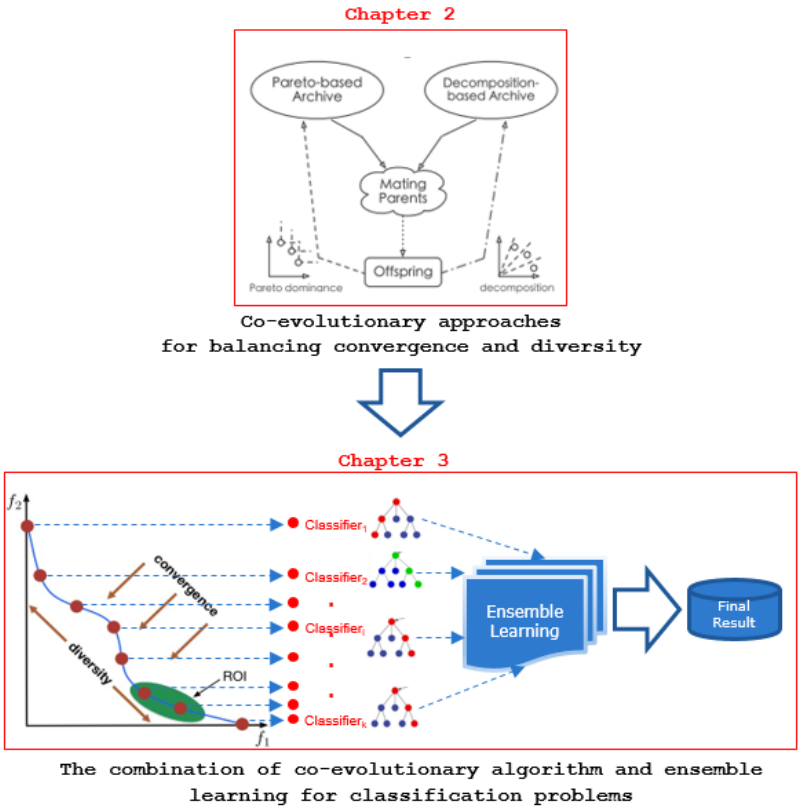


Figure 3: Illustrate the two main problems of this thesis. The first problem (i.e., balancing convergence and diversity in MOPs) is addressed in Chapter 2, while the remaining problems (i.e., designing co-evolutionary algorithms for imbalanced classification problems) are addressed in Chapter 3 of this thesis.

development, and improvement during the last three decades. In [25], C. A. Coello examined the background, current trends in development, and difficulties facing the field of evolutionary multi-objective optimization. The author stated that many people believe that the evolving multi-objective optimization area will be difficult for scientists, especially Ph.D. students, to make major contributions to after 20 years of rapid progress. However, ***the author did highlight that there are still a lot of exciting research topics being developed***. According to the author, there are currently two main development trajectories: one is in terms of ***objective space***, and the other is in terms of ***variable space*** (Figure.4). The majority of multi-objective optimization research is presently conducted in terms of variable space, particularly

for large-scale multi-objective optimization problems. The author underlines that the ***Co-operative co-evolutionary technique is the most well-liked and successful study direction to address this issue in this development direction***. Today's practical problems are typically complex multi-objective optimization problems that are challenging to resolve with just one optimization solution. As a result of this practice, hybrid algorithms have become a more widely utilized technique. One current trend in this development path is the employment of co-evolutionary approaches, which involve the deployment of numerous populations, each of which is concentrated on addressing a particular criterion.

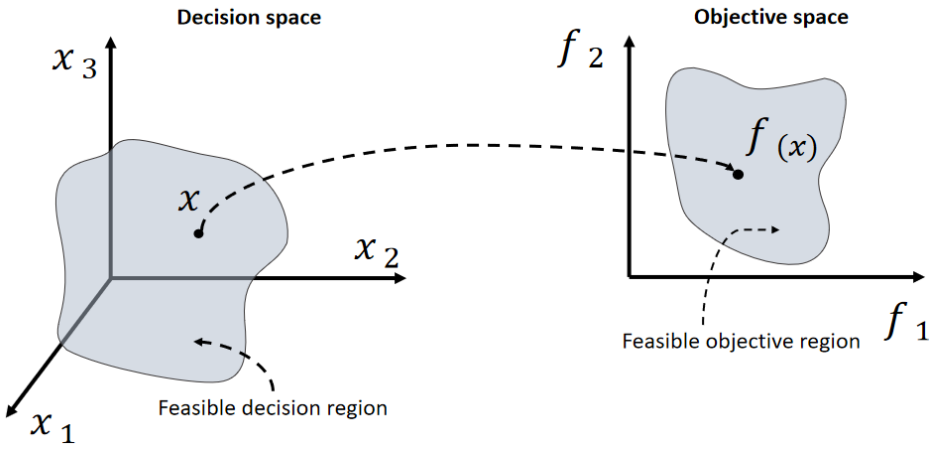


Figure 4: Illustration of the objective space corresponding to the decision variable space

In the field of multi-objective optimization, convergence and diversity are the two most crucial criteria to attain. ***The balance between these two factors is still a big challenge that the current multi-objective optimization algorithms are facing***. Well-known MOEAs now in use, including NSGA-II and MOEA/D, cannot address these two issues concurrently. Instead, each algorithm has a specific priority. While NSGA-II prioritizes convergence first, MOEA/D does the opposite. The CoEA can address this issue by utilizing a dual-population approach. This is a process whereby one population is used to obtain the

highest degree of convergence and another is used to achieve the greatest degree of diversity. A new population that fully converges on the two criteria will be created when these two populations combine. Up until now, there have been many studies using CoEA to solve this problem. Some typical studies can be mentioned as [65] [69] [126]. In addition, some recent studies are still focusing on addressing this balance issue for both the objective and decision spaces [84] [117], or in special cases such as changing decision variables [122] or constrained multi-objective optimization problems with the dynamic dual-population solution [61]. Although these studies have achieved feasible results, there are still many details that can be improved, as well as many new methods that can be proposed to deal with this problem.

After the initial success of applying the co-evolution algorithm to conventional multi-objective optimization problems, there have been an increasing number of studies using the co-evolution algorithm in conjunction with machine learning techniques to address real-world issues like classification, prediction, and clustering problems. The machine learning field has been dominated by two techniques: ensemble learning and deep learning [82]. The term ***Ensemble learning*** describes methods that aggregate the results of at least two different models. In general, ensemble methods yield more accurate results than a single model. According to empirical findings [62], the accuracy of the ensemble and the diversity of the base classifiers are positively correlated. Many strategies have been put forth to build a strong classifier ensemble by looking for both the diversity among them and the accuracy of the classifiers, and the multi-objective co-evolutionary approach is one of them. To generate individuals that fulfill both of these criteria, the multi-objective optimization algorithms often use accuracy (or convergence) and diversity as objective functions [14] [18]. After that, utilize a non-dominant

sorting mechanism (as in NSGA-II) to find the set of Pareto optimal solutions. As mentioned above, ***although this approach can find a set of solutions, the balance between convergence and diversity is still not guaranteed***. Meanwhile, multi-objective co-evolution is likely to ensure this balance. Some of the latest research using co-evolution combined with ensemble learning can be mentioned as [72] [119] [87] [15]. These studies demonstrate the effectiveness of using co-evolution to generate diverse and high-quality ensembles of classifiers for various classification tasks. By generating a Pareto set of diverse solutions, these methods ensure that the ensemble is both accurate and diverse, leading to improved classification performance. From this point, it can be seen that the combination of a co-evolutionary method and ensemble learning algorithms has great potential for solving machine learning problems.

To summarize, through the process of researching and examining this area, the following are the explanations for why the author chose this topic:

1. This remains an open topic and a promising study area in the multi-objective optimization community these days. There is still plenty of issues and challenges that need to be resolved (Especially the problem of balance between convergence and diversity in multi-objective optimization problems).

2. There haven't been many in-depth studies on co-evolution in the world or in Vietnam up to this point. These frequently concentrate on thoroughly addressing each minor issue in the realm of co-evolution. A comprehensive and complete study of the field of co-evolution is still necessary, and this has significant scientific implications.

3. Machine learning is currently gaining popularity across many facets of society. A significant issue they are currently dealing with is the growing number of large-scale, imbalanced datasets, etc. Studies have

been done on the basic idea of using a co-evolutionary approach to help machine learning algorithms further promote performance while tackling this challenge. The combination of machine learning (especially ensemble learning) and a co-evolutionary method has been continually researched and developed in recent years.

The three factors listed above are the main motivations leading the author to select the topic "*Enhancing the effectiveness of co-evolutionary methods in multi-objective optimization and applying to data classification problems*" as the main focus of the thesis's research.

Objectives and scopes

Objectives The thesis' primary objectives are: a thorough examination of the notion of co-evolution; developing a dual population co-evolution solution for the multi-objective optimization problems that balance convergence and diversity at the same time, and proposing co-evolutionary algorithms that can be used to solve classification problems.

Scopes The scope of the thesis is limited as follows:

- Experimental datasets: All these datasets are benchmarks, widely utilized by scientists around the world. The following information is specific to each data collection used to solve each problem:

+ The problem of balancing convergence and diversity in multi-objective optimization utilizes the four datasets: ZDT, WFG, DTLZ, and UF (More details of these datasets are described in the Appendix 4 of the thesis.)

+ The classification problem uses imbalanced datasets in KEEL dataset repository.

- Regarding methods of co-evolution:

+ Using co-operative and competitive co-evolutionary methods

+ The co-evolutionary methods use two populations (i.e., the dual

population).

+ The multi-objective models use two objectives.

Contributions

Following is a summary of the thesis' main contribution to the field of research:

First, proposing a dual-population paradigm (DPP)-based co-operative co-evolutionary algorithm for solving multi-objective problems (named DPP2) with new features:

1. *Using a new restricted mating selection mechanism (named RMS2)* to increase the probability of finding one solution in A_p.
2. *Using a new strategy of choosing alternative solutions* to increase the probability the offspring are generated from parents in different populations so they can take advantage of both the diversity and the convergence).
3. *Using a new update mechanism* to reduce the running time.

Second, proposing a DPP-based competitive co-evolutionary algorithm for the multi-objective evolutionary algorithms (named DPPCP) with new features:

1. *Using the neighbor-based selection mechanism (NBSM selection)* to address the imbalanced issue that previous methodologies frequently have with the co-evolutionary processes between two populations.
2. *Using competitive co-evolutionary mechanisms* to make two offspring interact with each other instead of the cooperative co-evolutionary mechanism.

Third, proposing a multi-objective competitive co-evolutionary algorithm for imbalanced dataset classification problems (named IBDPPCP) with new features:

1. *Data sampling*: IBDPPCP uses a combination of upper and lower sampling techniques instead of using only the upper sampling as in the premise research. By using this method, the imbalance is resolved without causing noisy data in the overlapping area.

2. *The combination of a DPP-based algorithm and an ensemble learning algorithm:* Thanks to the ability to find individuals that can satisfy both convergence and diversity factors, IBDPPCP is suitable when combined with an ensemble learning algorithm for solving classification problems.

Fourth, proposing a multi-objective co-operative co-evolutionary algorithm (named IBMCCA) for solving classification with imbalanced data. The primary contribution of this algorithm is a dual-population cooperative co-evolutionary model to address both FS and IS problems. This new model allows for finding a set of individuals (or sub-datasets) that have both convergence and diversity factors. IBMCCA utilizes the same data sampling and ensemble learning strategies as IBDPPCP. The main difference between the two algorithms is the co-evolutionary model. IBDPPCP uses a competitive model with two populations having the same individual encoding (i.e., FS and IS); in IBMCCA, two populations use a cooperative model with two different individual encodings.

Structure of the thesis

This thesis is organized into four chapters as follows:

1. Chapter 1 introduces the background knowledge related to the research problem. Multi-objective optimization techniques will come initially. An overview of multi-objective co-evolutionary methods is then introduced. It will go into great length about both cooperative and competitive co-evolution. The connection between co-evolution and ensemble learning, in particular, is covered in the final section.

This is the basis for the following chapter, which presents in more detail the application of co-evolution to solving classification problems.

2. The first significant issue that the thesis attempts to address is introduced in Chapter 2. It is a balancing problem between convergence and diversity in multi-objective optimization problems using the dual population paradigm (DPP). There are two proposed solutions given here. The first is an upgraded version of the original DPP algorithm (named DPP2). Ideas, details of improvements, and experiments will be presented. After this version of DPP2, a main proposed algorithm for this problem will be presented (named DPPCP). The details on contributions, advancements, and experiments are presented.

3. Chapter 3 introduces the applications of co-evolution in the field of machine learning. Two multi-object cooperative and competitive-based algorithms for imbalanced classification problems are presented. The author has employed two dual-population co-evolutionary methods in this chapter to solve classification challenges.

4. Conclusion and future works: Summary of thesis contents, achieved issues, and main contributions of the thesis and future research directions.

Chapter 1
BACKGROUNDS

1.1. Multi-objective optimization

1.1.1. Preliminary concepts

A multi-objective optimization problem (MOP) can be defined as follows:

Minimize:

$$F(x) = (f_1(x), ..., f_m(x))^T \tag{1.1}$$

Subject to: $g_i(x) \leq 0; \forall i = 1, ..., p.$ $h_j(x) = 0; \forall j = 1, ..., q.$

Where, a solution $x = (x_1, ..., x_n) \in \Omega$ is a vector of decision variables; is the decision variable space or simply the decision space. $g_i(x)$ and $h_j(x)$ are called constraint functions. If any solution x satisfies all constraints and variable bounds, it is known as a feasible solution, otherwise, it is called an infeasible solution. There are m objective functions $F(x) = (f_1(x), ..., f_m(x))^T$; $F : \Omega \to \Re_+^m$.

where $\Re_+^m$ is called the objective space. For each solution x in the decision variable space, there exists a point in the objective space.

Definition 1. A solution $x^{(1)}$ can *dominate* another solution $x^{(2)}$, denoted as $x^{(1)} \prec x^{(2)}$ if and only if: $\forall i \in \{1, ..., m\} : f_i(x^{(1)}) \leq f_i(x^{(2)})$ and $\exists j \in \{1, ..., m\} : f_i(x^{(1)}) < f_i(x^{(2)})$.

Definition 2. A feasible solution $x^* \in \Omega$ is a *Pareto optimal solution* if $\nexists x \in \Omega$ such that $x < x^*$.

Definition 3. The set of all Pareto optimal solutions is called the *pareto*

set (PS), denoted as $PS = \{x^* \in \Omega \mid \nexists x \in \Omega, x \prec x^*\}$.

Definition 4. The set of all objective function values corresponding to the solutions in PS is called the *Pareto front* (PF), denoted as $PF = \{F(x) \mid x \in PS\}$.

Definition 5. The ideal objective vector is $Z^* = (f_1^*, ..., f_m^*)^T$. Where f_m^* is the minimum value of the m-th objective function.

Definition 6. The nadir objective vector is $Z^{nad} = (f_1^{nad}, ..., f_m^{nad})^T$. Where f_m^{nad} is the maximum value of the m-th objective function.

1.1.2. Typical MOEAs

a. Non-dominated sorting genetic algorithm II (NSGA-II)

NSGA-II [30] is one of the most common algorithms among Pareto-based EMO algorithms. The pseudocode of the NSGA-II algorithm is shown on Algorithm 1. Convergence and diversity are taken into account in turn in NSGA-II. Individuals are ranked at each generation using a non-dominated sorting technique. A population is split into various fronts as a result. Individuals with lower ranks (i.e. corresponds to better convergence) are preselected. Next, by using a diversity selection strategy (i.e. crowding distance), individuals on the final front are chosen up to the size of a population. The maintenance of diversity is thus secondary in NSGA-II. It only ensures diversity for a small subset of the population's solutions; the rest are primarily chosen based on convergence, regardless of their diversity. Due to this, it is difficult to solve issues with many objectives (more than three), or challenging problems with a complex Pareto-optimal set.

b. The multiobjective evolutionary algorithm based on decomposition (MOEA/D)

MOEA/D [130] is a decomposition-based method. It decomposes MOPs into a set of single-objective optimization sub-problems through

Algorithm 1: Procedure for NSGAII

procedure NSGAII(N,N_A)
$t \leftarrow 0$
$P_t \leftarrow$ *new_population(N)*
$\mathcal{Q}_t \leftarrow \emptyset$
$A \leftarrow$ non_dominated(P_t)
while not stop criterion **do**
  $R_t \leftarrow P_t \cup \mathcal{Q}_t$
  $\mathcal{F} \leftarrow$ *fast_non_dominated_sorting*($\mathcal{R}_t$)
  $P_{t+1} \leftarrow \emptyset$
  $i \leftarrow 1$
  while $|\mathcal{F}_i| + |P_{t+1}| \leq N$ ***do***
    $\mathcal{C}_i \leftarrow$ *crowding_distance_assigment*($\mathcal{F}_i$)
    $P_{t+1} \leftarrow P_t \cup \mathcal{F}_i$
    $i \leftarrow i+1$
  end while
  $\mathcal{F}_i \leftarrow$ ***sort***($\mathcal{F}_i$,$\mathcal{C}_i$,'descending')
  $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$ ▷ *fill* P_{t+1} *with the N -* $|P_{t+1}|$ *less crowded individuals of* $\mathcal{F}_i$
  $\mathcal{Q}_t \leftarrow$***selection***(P_{t+1}, N)
  $\mathcal{Q}_t \leftarrow$ ***crossover***($\mathcal{Q}_t$)
  $\mathcal{Q}_t \leftarrow$ ***mutation***($\mathcal{Q}_t$)
  $t \leftarrow t+1$
  $A \leftarrow$ ***non_dominated***($A \cup \mathcal{Q}_t$)
end while
end procedure

an aggregation method (such as the weighted sum, Tchebycheff, and boundary intersection approaches [75]). In order to address these sub-problems, a population-based algorithm is applied. In MOEA/D, each solution is associated with a sub-problem, and the population consists of the best solution for each sub-problem. Therefore, the diversity among these sub-problems will result in diversity in the population. In addition, a set of evenly spread weight vectors is used by MOEA/D to identify the search directions. Therefore, MOEAD can produce a uniform distribution of Pareto solutions. The pseudocode of the algorithm is presented in Algorithm.2

Algorithm 2: The MOEA/D general framework

Input:

+ N: The number of the sub-problems considered in MOEA/D
+ $\lambda^1, ..., \lambda^N$: N weight vectors
+ T: the neighborhood size
+ gen_{max} : $The generation number$

Output:

EP

Step 0 - Setup:

Set EP = $\emptyset$

gen=0

Step 1 - Initialization

Uniformly randomly generate an initial internal population:
$IP = \{x^1, ..., x^N\}$ and set $\text{FV}^i = F(x^i)$.

Initialize $\mathcal{Z} = (\mathcal{Z}_1, ..., \mathcal{Z}_n)^T$

Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector.

$\forall i = 1, ..., N, set B(i) = \{i_1, ..., i_T\}$ where $\lambda^{i_1}, ..., \lambda^{i_T}$ are the T closest weight vectors to λ_i

Step 2 - Update

For i = 1 **to** N **do**

Randomly select two indexes k, l from B(i), and then generate a new solution y from x^k and x^l by using genetic operators.

Update of $\mathcal{Z}, \forall j = 1, ..., n, if\ \mathcal{Z}_j < f_j(y), then\ set \mathcal{Z}_j = f_j(y)$

Update of Neighboring Solutions:

For each index j $\in B(i)$ **if** $g^{tc}(y|\lambda^j, \mathcal{Z}) \leq g^{tc}(y|\lambda^j, \mathcal{Z}^*)$ **then** set $\text{x}^j = y\ and\ FV^i = F(y^j)$.

Update of EP: Remove from EP all the vectors dominated by F(y).

Add F(y) to EP if no vector in EP dominate F(y).

Step 3 - Stopping criteria

If gen = gen_{max}, **then** stop *and* output EP, **otherwise** gen = gen + 1, go to Step 2.

1.2. Co-evolutionary Algorithms

1.2.1. Defining co-evolution

Co-evolution has its roots in the field of biology. Charles Darwin was the first to mention how flowering plants and insects evolved together in 1895 [27]. He demonstrated how plants and insects can evolve through mutual evolutionary changes, even though he did not use the term "*co-evolution*". This work paves the way for numerous further in-

vestigations into how interactions between species can affect each other's evolutionary processes. Around the beginning of the 1940s, plant pathologists created breeding programs. At first, they created novel cultivars that were disease-resistant to various degrees. However, this has allowed disease populations to rapidly evolve in order to outpace the plant's defenses. This fact requires the development of new plant varieties that are resistant. As a result, there has been an ongoing cycle of reciprocal evolution in both plants and illnesses.

The study of the interactions between butterflies and plants by two authors, Ehrlich and Raven, in 1964 is where the term "*co-evolution*" first appeared [38]. Although they did not come up with the concept of co-evolution initially, their stimulating work helped to promote it and sparked the interest of numerous generations of co-evolution-focused scientists. The term "*co-evolution*" refers to the evolution of two or more evolutionary entities as a result of reciprocal beneficial selective effects. A change in plant morphology, for instance, might have an evolutionary impact on herbivore morphology, which in turn could have an impact on plant evolution, and vice versa. Although co-evolution is largely a biological concept, it has been used as an analogy in other disciplines, including computer science, sociology, and astronomy. Following are some of the concepts related to co-evolution that have been introduced.

Definition 1.1. *co-evolution is reciprocally generated evolutionary change between two or more species or populations*
(According to evolutionary biologist Price (1998))

Definition 1.2. *A system is considered co-evolutionary* ***if and only if*** $f_P^{Tr}(x)$*—the "true" fitness propensity of each evolving individual (or trait),* x*—varies with respect to other* ***reciprocally evolving*** *individuals (or traits).*

To help the reader comprehend the various types of potential metrics

for individuals, the following four definitions are presented [118].

Definition 1.3. ***Objective measure****: A measurement of an individual is objective if the measure considers that individual independently from any other individuals, aside from scaling or normalization effects.*

Definition 1.4. ***Subjective measure****: A measurement of an individual is subjective if the measure is not objective.*

Definition 1.5. ***Internal measure****: A measurement of an individual is internal if the measure influences the course of evolution in some way.*

Definition 1.6. ***External measure****: A measurement of an individual is external if the measure cannot influence the course of evolution in any way.*

Given the above definitions, it is tempting to define *co-evolution* as follows:

Definition 1.7. ***co-evolutionary algorithm*** *is an EA that employs a subjective internal measure for fitness assessment*

Traditional EAs evaluate an individual's fitness objectively, separate from the population environment in which they are located. CoEAs operate similarly to standard EAs, with the exception that fitness evaluations are subjective rather than objective. Through its interactions with other individuals in the evolutionary system, an individual is evaluated. Simple CoEAs [94] first choose a few individuals from the population to serve as the evaluators. Then, each member of the population is evaluated using these assessors. This evaluation approach ought to theoretically provide a good approximation of an individual's genuine fitness whenever the range of evaluators is sufficiently diverse. The key benefit of CoEA over regular EA is its divide-and-conquer deconstruction approach. The CoEA primarily has four benefits [79]. First, by

breaking the problem down into smaller components, parallelism can accelerate the optimization process. Second, each subproblem is resolved by a different subpopulation, maintaining a wide variety of solutions [20]. Third, breaking a system down into smaller components makes it more resilient to mistakes and failures in individual modules, which improves its capacity to be reused in dynamic contexts [89]. Finally, if the issue is correctly decomposed, the rapid decrease in performance with a rise in the number of decision variables can be somewhat mitigated.

1.2.2. Types of co-evolutionary methods

CoEA algorithms can be categorized in a variety of ways, but the most typical ones are determined by the number of populations and the way these populations co-evolve.

Based on population number, CoEA can be separated into the following three groups [78]:

1. **1-Population co-evolution:** A single population's individuals assess their fitness through competition with one another in games. It is frequently utilized to develop effective competitive strategies (e.g., for checkers or soccer).

2. **2-Population (or dual population) co-evolution**: There are two smaller populations within the larger population. How many members of sub-population 2 that an individual in sub-population 1 can defeat in a competition serves as a measure of its fitness (and vice versa). Essentially, sub-population 1 comprises the potential solutions that are of interest to us, and sub-population 2 contains test cases for those potential solutions. This method is usually utilized to help sub-population 1 identify strong candidate solutions despite whatever challenges sub-population 2 may present.

3. **N-Population (or multi-population) co-evolution**: The prob-

lem is broken down into n sub-problems; for instance, if the task is to come up with soccer plans for a team of n robots, each sub-problem is to figure out a plan for a single robot. An individual's fitness is evaluated by choosing members of the other sub-populations, combining them with this individual to make a whole *n-sized* solution (in this case, a complete soccer robot team), and then judging the fitness of that solution. This type of CoEA is frequently employed to break large problems down into smaller, more manageable problems in order to lessen their high dimensionality.

Based on the interactions between populations, CoEA can be divided into two main categories: competitive co-evolution [105] and co-operative co-evolution [96]. In competitive co-evolution, each individual's fitness is assessed by an adversarial battle with others. In contrast, in co-operative co-evolution, the collaboration and complementarity between individuals influence each individual's fitness. Below, a detailed explanation of these two algorithms' components will be provided.

1.2.3. co-operative co-evolutionary algorithms

co-operative co-evolutionary algorithms (CCEA) are frequently employed when an issue can be organically divided into smaller components (or sub-components). CCEA uses a different population (or species) for each of these sub-components. Since each individual in a given population only represents a portion of a possible solution to the issue. Therefore, to calculate fitness, a collaborator is chosen from the other populations to represent the other sub-components. The objective function is assessed once the individual is merged with this collaborator to form a complete solution. How successfully a sub-population "cooperates" with other species to achieve beneficial outcomes is a measure of its fitness.

The decompositional aspect of co-evolution may provide CCEA some advantages for handling issues that are complex yet highly structured, whereas classical evolution may be fully applicable to static single-objective optimization problems of arbitrary complexity. It would seem logical that a CCEA could coevolve the different sub-components independently more effectively than a typical EA could (conventional EAs often evolve the entire structure). In fact, this has been the main driving force behind co-operative co-evolutionary methods.

In 1994, Potter and De Jong [95] proposed a general CCEA framework for static function optimization problems. This study paved the way for future CCEA research. In 2000 [96], these authors used this paradigm to apply neural network learning to problems. In Potter's model, each population has individuals that represent a part of a whole solution, and these populations evolve nearly independently of one another while

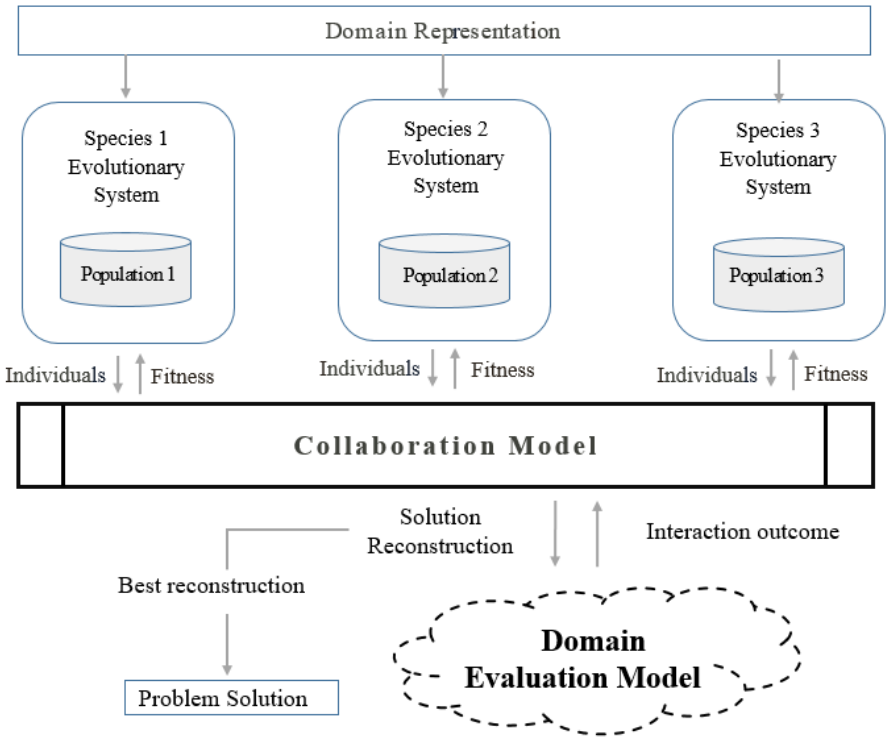


Figure 1.1: Co-operative co-evolution's architectural framework. The domain evaluation model's solid line indicates the requirement for an absolute fitness function.

Algorithm 3: co-operative co-evolutionary algorithms (CCEA)

```
Data: P← {P_1, P_2, ..., P_N}
Result: P
for population p_s ∈ P, all population do
    Initialize population p_s
for population p_s ∈ P, all population do
    Evaluate p_s
t:= 0
do
    for population p_s ∈ P, all population do
        Select parents from population p_s
        Generate offspring from parents
        Select collaborators from P
        Make a complete solution via combining offspring with collaborators
        Evaluate offspring via the fitness of complete solution
        Select survivors for new population p_s
    t:= t+1
until Terminating criteria is met
```

working in concert to maximize fitness. Such a process can be either static (i.e., the divisions for the various components are predetermined and never changed) or dynamic (i.e., populations of components may be added or subtracted as the run goes on). Other researchers have adapted or used Potter's techniques. The CCEA was utilized in [40] by Eriksson and Olsson to optimize inventory control. In order to cooperatively coevolve neural networks, Moriarty and Miikkulainen [86] adopted a different, perhaps more adaptable strategy in 1997. In this instance, the parent population stands in for various network designs, and the child population is employed to gather node data. Designs are judged on how successfully they work with their cooperating nodes to solve an issue, and the cooperating nodes partake in this fitness. Thus, a node only gains fitness indirectly by being rewarded for engaging more often and coming up with successful strategies. The CCEAs have been successful in a number of fields, including function optimization, production scheduling, constructing artificial neural networks, and room painting.

1.2.4. Competetive co-evolutionary algorithms

In 1992, the study [55] was the first to put forth a competitive co-evolution model that sorted networks using the Prey and Predator concept. The transfer of co-evolution from the realm of biology to the field of computer science is being published for the first time with this paper. Hillis used two independent populations, one generated by a set of Sorting Networks and the other by a set of test data sets. A fitness score is given to an individual in one population that represents a prospective sorting network depending on how successfully it sorts an opponent's data set from the other population. Individuals in the second population, meanwhile, represent potential data sets. The fitness is determined by how successfully they trick rival sorting networks. In reality, competitive co-evolution has been the focus of the majority of early research in co-evolutionary algorithms. Competitive co-evolution can take place between different populations or within a self-playing population. Multiple species interact in competitive co-evolution. They compete with one another for access to shared resources and space. The Iterated Prisoner's Dilemma [97] as well as identifying strong game strategies in games like Tic-Tac-Toe and backgammon, have all been solved using single population competitive co-evolution.

Competitive co-evolutionary algorithms (CPEA) have been used most frequently in relation to gaming tactics ([102], [93]). Additionally, by creating the idea of competitive fitness to offer a more comprehensive training environment than standalone fitness functions, it illustrates how competition can be used to evolve superior solutions. In attempts to coevolve complicated agent behaviors, the competition was crucial. Finally, a range of machine learning issues has been addressed using competitive approaches ([92], [81]).

In general, a competitive co-evolution solution goes through the fol-

lowing processing: Each population's individuals are first assigned at random. The first population will then make an effort to fit into the second group's environment. The members of the second population will simultaneously make an effort to fit into the environment that the first population has built. The relative fitness evaluation function for each member of both populations will then be computed. The level of adaptation of a member of this population to the environment produced by one or a few members of the other population is represented by this relative evaluation function. Better fit individuals will be chosen for the following generation based on these relative fitness scores.

In short, the battle for survival among individuals is what drives co-evolution in the competitive co-evolution paradigm. Competitive co-

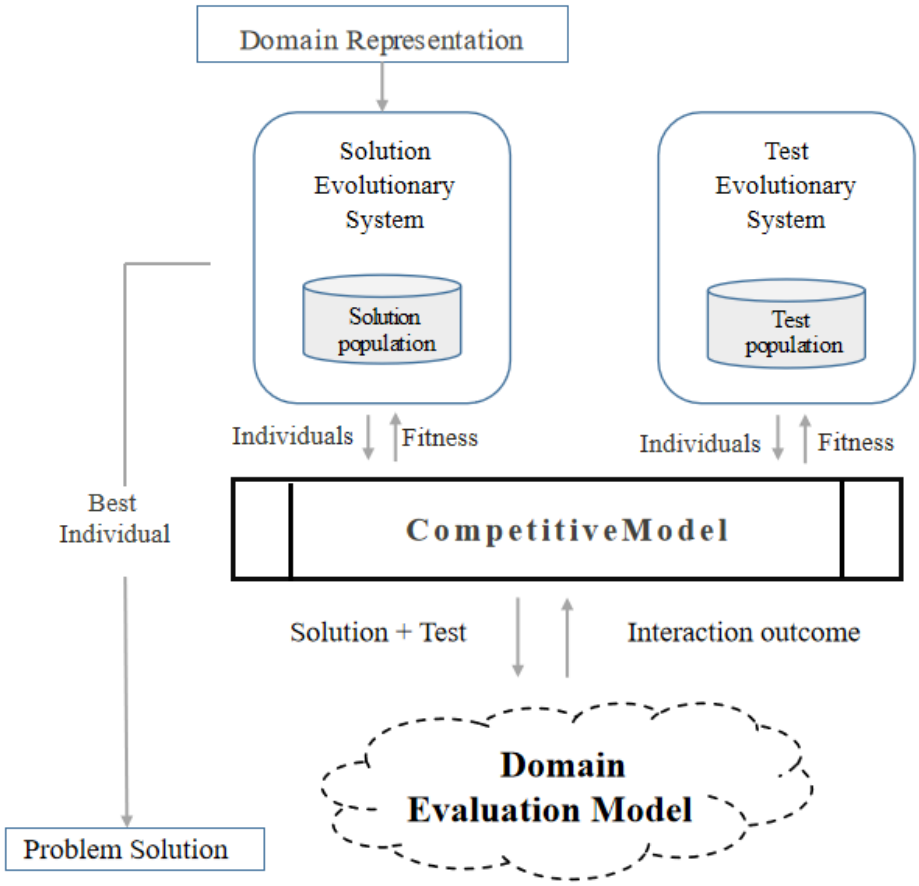


Figure 1.2: Competitive co-evolution's architectural framework. A possible relative interaction function is shown by the domain evaluation model's dashed line.

Algorithm 4: Competitive co-evolutionary algorithms (CPEA)

Data: P_s : $SolutionPopulation$; P_t : $TestPopulation$; P
Result: P

```
P ← {P_s, P_t}
for population p ∈ P, all population do
    Initialize population p
for population p ∈ P, all population do
    Evaluate p
t:= 0
do
    for population p ∈ P, all population do
        Select parents from population p
        Generate offspring from parents
        if p is P_s then
            Select competitors from P_t
        else
            Select competitors from P_s
        Evaluate offspring via competing against collaborators
        Select survivors for new population p
        t:= t+1
until Terminating criteria is met
```

evolution may result in an arms race when populations compete against one another to outperform one another and overcome more difficult issues.

1.2.5. Current co-evolution research directions

The algorithms based on the co-evolution technique can be divided as shown in Figure 2.3.

In general, algorithms are divided into three main groups:

+ The algorithms are based on the co-operative co-evolutionary approach

+ The algorithms are based on the competitive co-evolutionary approach

+The algorithms are based on hybridizing both co-operative and competitive approaches.

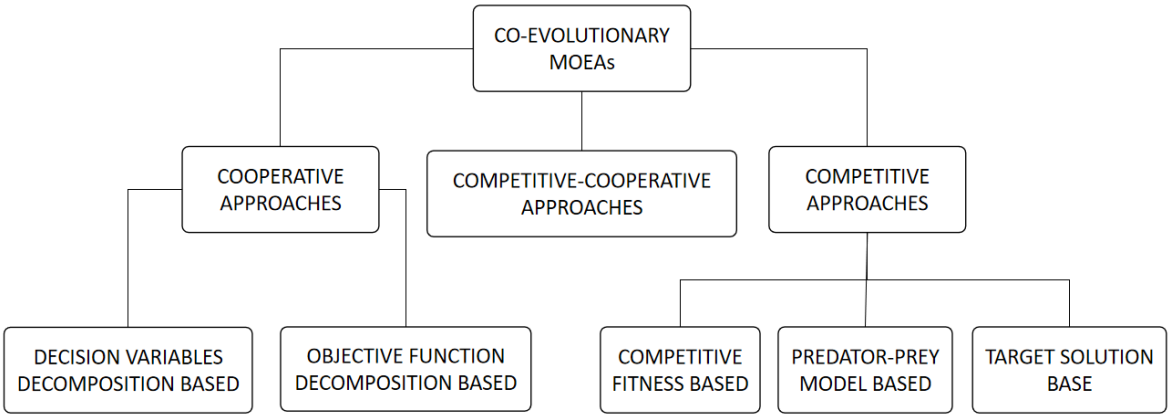


Figure 1.3: Classification of co-evolutionary algorithms

a. The algorithms are based on the co-operative co-evolutionary approach

The group of co-operative co-evolution algorithms can be further subdivided based on the way decomposing the problem into sub-problems. It is possible to decompose based on decision variables or objective functions.

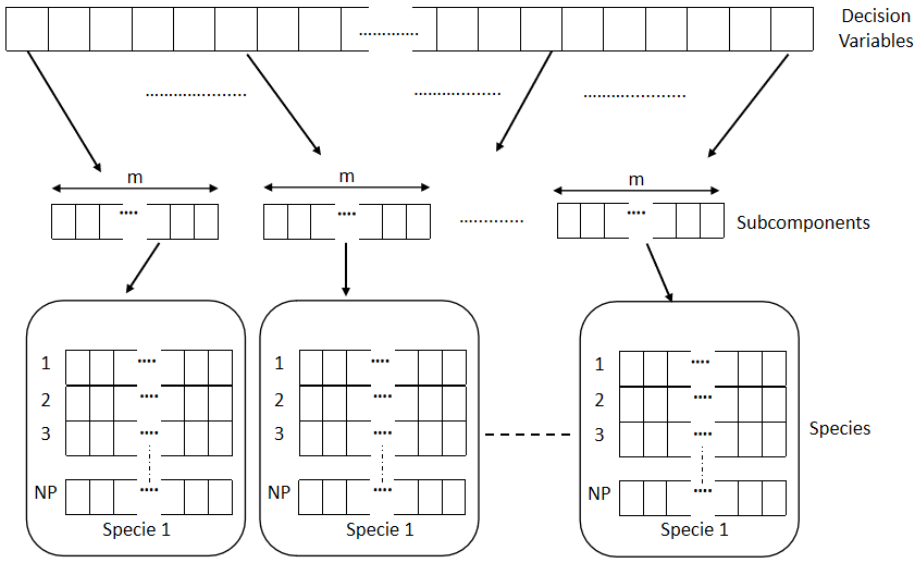


Figure 1.4: Co-operative co-evolutionary model based on decomposition by decision variable. Each sub-population is used to optimize a sub-components (i.e. a small part of the decision variables)

Figure 1.4 illustrates the main idea of the co-operative co-evolutionary algorithms that break down dependent decision variables. Multi-Objective

Problem (MOP) is broken down into sub-problems along the search space. Specifically, after being divided into sub-problems, each of these sub-parts becomes an individual and is assigned to a population. The task of each population now is to optimize these sub-parts simultaneously. After the optimization process, each offspring needs to be combined with individuals from other populations to form a complete solution. The most difficult of these algorithms is how to break down the original problem or how to optimize them as well as combine them after being refined. Division by variable space is difficult, especially for extremely complicated issues. Typical algorithms for this approach are [4], [8], [128].

Figure 1.5 displays the model that the MOP is decomposed based on the objective functions of the problem. Each objective function is assigned to a certain sub-population, and all these sub-populations are combined to create an approximation of a distribution over the entire Pareto front. In this model, each individual is a solution to the MOP. These individuals are computed with all objective functions, just like in conventional MOEAs. However, the distinction, in this case, is that the fitness value of each individual in each sub-population will only be evalu-

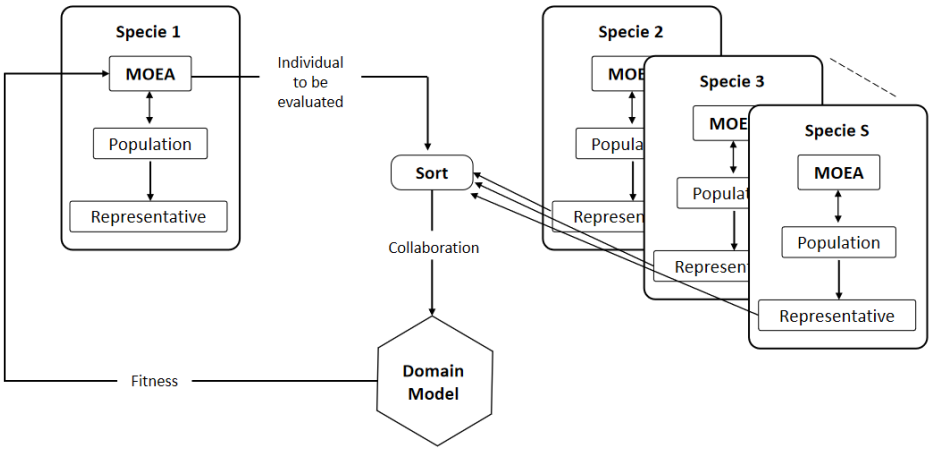


Figure 1.5: Collaborative co-evolution model based on objective function decomposition. Each sub-population represents a single objective function

ated using the value of the relevant sub-population's objective function. In this way, individuals within each sub-population are then navigated in accordance with each of that sub-population's objective functions in order to look for various Pareto Front regions. Typical algorithms for this approach are [115], [71] [116].

b. The algorithms are based on the competitive co-evolutionary approach

These competing co-evolutionary strategies can be further divided according to the manner of resistance, which can be antagonistic based on adaptive function, the Predator-Prey model, or target solutions. In the Predator-Prey-based antagonistic model, an instance of the decision variable space is represented by a predator. The prey, meanwhile, stands in for the objective function. This model attempts to simulate how a predator seeks its prey. The predator will only take the weakest prey (or solutions with the lowest objective function value). Studies that address this concept include [37] [53].

The adaptive antagonist-based competitive co-evolution algorithms evaluate fitness differently than traditional multi-objective optimization methods. It employs a special fitness function that takes into consideration population interdependencies. Different methods [39] can be used to calculate this function. If there are two populations A and B, we can use the following techniques to determine the n^{th} individual's relative fitness (named $C_{A,n}$) in population A.

1. **Simple fitness**: sampling a set of individuals $\{C_{B,1}, C_{B,2}, \ldots, C_{B,m}\}$) in population B then let $C_{A,n}$ antagonize each $C_{B,i}$. The relative fitness value of $C_{A,n}$ is the number of times it wins.

2. **Fitness sharing**: This method is interested in the similarity of individuals in the same population. After calculating the *SimpleFitness* value of each individual A, divide this value by the *Similarity* value.

The *Similarity* value of each individual is the number of individuals in population A that beat the same sample of individuals in population B. The utilization of this function will result in higher points being awarded to unusual individuals.

3. **Competitive fitness sharing**: Suppose that the sampling population B includes individuals $\{C_{B,1}, C_{B,2}, \ldots, C_{B,m}\}$. N_m is the total number of individuals in population A that beat individual $C_{B,m}$. The fitness value of each individual $C_{A,n}$ is defined as follows:

$$\mathcal{F}(C_{A,n}) = \sum_{m=1}^{M} \frac{1}{N_m} \tag{1.2}$$

This strategy raises the reward points for population A's members who defeat population B's members, which some other members of population A are unable to do.

In competitive coevolution, the method of sampling for confrontation is also quite important. There are many different ways of sampling (Figure 1.6), and some strategies can be mentioned as follows:

1. **All versus all sampling**: Each individual in population A will compete against all individuals in population B.

2. **Random sampling**: randomly sample one or more individuals in population B and bring them up against an individual in population A.

3. **Tournament sampling**: Use relative measurement to select the most effective opponent to battle.

4. **All versus best sampling**: all individuals in population A compete with the best individual in population B.

The competitive co-evolution model (Figure 1.7), which is based on

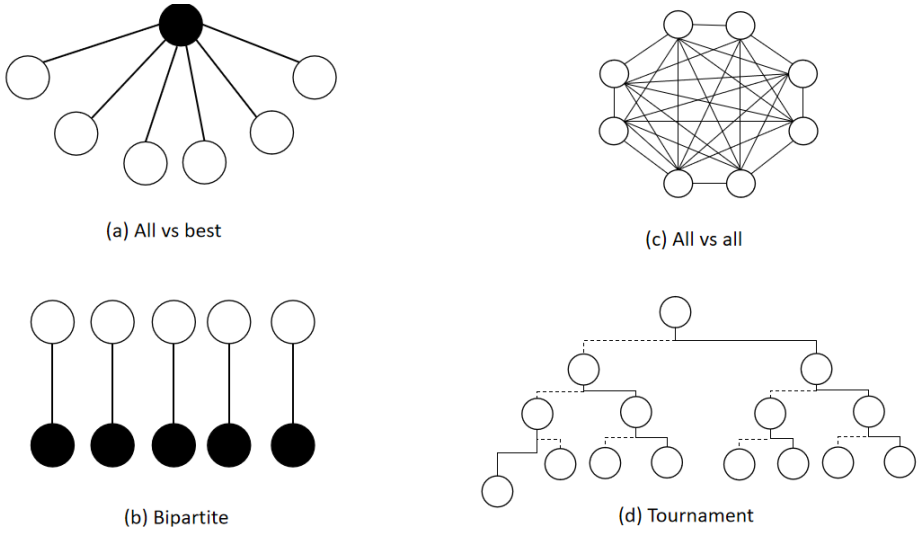


Figure 1.6: Some patterns of adversarial sampling: (a) all members of population A are pitted against the best member of population B; (b) each individual play a one-on-one match against each other; (c) all members of population A are pitted against each other, and (d) a duel is held within each population before a pair is chosen to engage in combat

the co-evolution of target solutions, operates under the following fundamental tenet: performing battle between two populations, one of which provides potential MOP solutions and the other of which has desired solutions for each objective function. In other words, for any objective function, this population contains the best feasible solutions. Following each evolution generation, these outstanding solutions are updated. The typical studies for this model are [76] [88].

c. The hybrid co-evolutionary algorithms

There are now numerous studies [43] demonstrating that in co-evolutionary techniques, maintaining a balance between cooperation and competition is necessary to avoid algorithmic instability. In this hybrid model, the decomposition process is self-adapting rather than being fixed from the start. The model is designed to achieve both convergence and diversity in the most efficient manner possible.

Similar to the co-operative co-evolution model, individuals from populations will evolve separately and collaborate with each other to solve the MOP, but this model also contains a co-evolutionary competitive

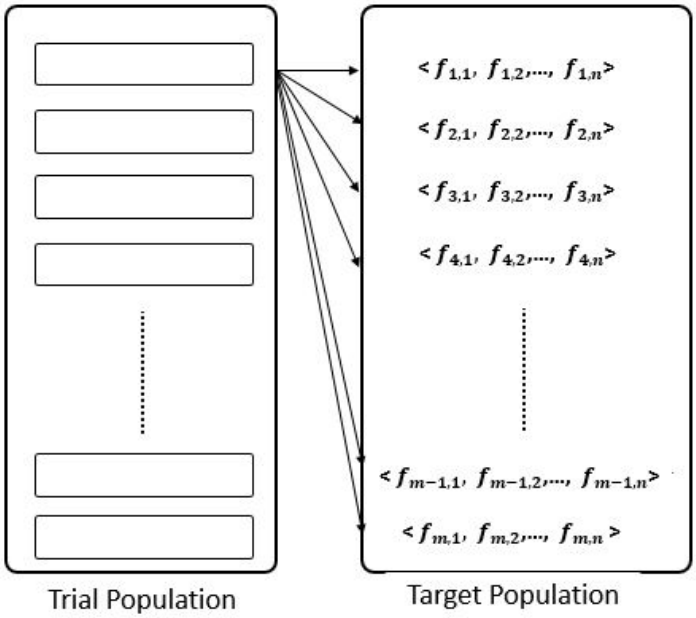


Figure 1.7: The competitive co-evolution model is based on the target solution set, the left population contains the set of possible solutions and the right population (the target population) contains the best achievable target vectors

mechanism. Individuals engage in an arms race as a result of the competitive mechanism, which enhances the performance of the ecosystem as a whole. Additionally, it enables the discovery of component interdependencies. This co-operative/competitive hybrid model offers a method that permits the employment of strategies that maintain the diversity of both mechanisms. In co-evolution, the division of MOP into various subpopulations and the concurrent evolution of these populations contribute to population variety. However, the inherent behaviors of each member of each sub-population do not preserve this diversity trait. This is made possible via competitive co-evolution [26]. The search space within each population is widened as a result of individuals competing with one another for survival. These case studies employ this strategy: [24, 52].

1.3. The co-evolutionary algorithms in machine learning

Co-evolutionary approaches have been increasingly popular in recent years as solutions to machine learning problems. Some common applica-

tions of co-evolution for machine learning problems can be mentioned as neural network co-evolution [90]; multi-agent reinforcement learning [60]; Clustering and classification [72] [54]; Time series prediction [21] [129] Currently, machine learning algorithms are facing a number of data-related problems such as *insufficient,* ***imbalanced****, incomplete, high-dimensional, or abundant* [90]. Within the scope of this thesis, the author has focused on solving classification problems with imbalanced data. The proposed methodology's core is composed of a number of components. First, ***feature selection (FS)*** is used to simplify overlapping areas and make it easier to create rules to differentiate between classes. Second, ***instance selection (IS)*** is utilized to select samples from all classes. Determining the best class distribution for the learning task will address the imbalance directly and may also help to eliminate noise and challenge borderline samples. Finally, the combination of a co-evolutionary approach and an ***ensemble learning algorithm*** is utilized to produce the final results. Up until now, there have been many proposals to use co-evolution to solve each of the above problems. In [58], the authors proposed a competitive co-evolution paradigm for FS and utilized this model for the diagnosis of pulmonary emphysema problems. The Spider Monkey Optimization (SMO) algorithm and the Paddy Field Algorithm (PFA) are the two bio-inspired algorithms that serve as the model's foundation. A co-operative co-evolution for FS in Big data with random feature grouping (named CCFSRFG) was proposed in [99]. The feature vector is dynamically divided by CCFSRFG into smaller, lower-dimensional sub-datasets, and each sub-dataset is represented by a sub-population. In [63], for three-objective feature selection, the authors suggested a multi-objective large-scale co-operative co-evolutionary method, named MLFS-CCDE. In this method, a framework for co-operative searching is created to quickly and effectively find

the best feature subset. Besides, this method established three objectives to direct the evolution of feature combinations: feature number, classification accuracy, and total information gain. In this method, a cluster-based decomposition technique is developed as part of the framework's decomposition process to minimize computation. In [50], the authors introduced a co-operative co-evolutionary approach for multilabel problem instance selection. co-operative evolution occurs between two separate populations. One population focuses on finding solutions for each label, while the second population integrates these findings to seek solutions for the multilabel dataset challenges.

It has been established that the combination of co-evolutionary algorithm and ***ensemble learning*** are superior. While the basis learners for the ensemble learning solution must simultaneously assure performance (i.e., **convergence**) and variety (i.e., **diversity**). Since these base learners are initialized randomly in conventional ensemble machine learning algorithms, it is challenging to achieve the two factors mentioned above right away. The co-evolution approach helps to speed up this procedure and improve the effectiveness of ensemble learning methods. There have been many studies on this issue. In [51], a co-operative co-evolution method for creating neural network ensembles was presented. This model has two key goals: first, improving the combination of the trained individual networks; and second, fostering the co-operative evolution of such networks rather than individual network training. To accomplish this, this method took into account not only how well each network performs in the given task but also how well it cooperates with the other networks. Finally, rather than choosing all networks for ensemble learning, a subset of networks is chosen after an evolution process. In [87], the authors proposed a co-operative co-evolutionary algorithm to create an ensemble of accurate and diverse multi-label classifiers. The

algorithm evolves several sub-populations simultaneously, each of which utilizes a different subset of the training data. Also, each individual is focused only on a small subset of labels. This way, the ensemble has a greater diversity of members.

1.4. The imbalanced data classification problem

1.4.1. Preliminary concepts

Definition 1: Imbalanced dataset

A dataset is said to be imbalanced when a class or a set of classes is represented in a smaller number than the other classes. The majority class, also known as the negative class, is the set of data that contains the greatest number of instances, whereas the minority class, also known as the positive class, contains the fewest examples.

Definition 2: Imbalanced Ratio

The degree of imbalance, which measures the proportion of data instances in the majority class ($n_{majority}$) to those in the minority class ($n_{minority}$), can be used to determine whether or not a set of data is unbalanced. The imbalanced ratio can be defined by Eq (1.3).

$$Imbalanced\ Ratio\ (IR) = \frac{n_{majority}}{n_{minority}} \tag{1.3}$$

Definition 3: Overlapping ratio

When there is shared data across each class, overlap happens. The scenario would be more complicated for classification if overlap happens along with unbalanced data. One technique for determining the overlapping ratio is the maximum Fisher's Discriminant Ratio (FD). The FD is defined by Eq (1.4).

$$FD_i = \frac{(\mu_{minority} - \mu_{majority})_i^2}{(\sigma^2_{minority} - \sigma^2_{majority})_i} \tag{1.4}$$

where

FD_i is Fisher's Discriminant Ratio of feature i.

$\mu_{minority}, \mu_{majority}$ are mean of minority class and majority class, respectively

$\sigma^2_{minority}, \sigma^2_{majority}$ are variance of minority class and majority class, respectively

1.4.2. Imbalanced approaches

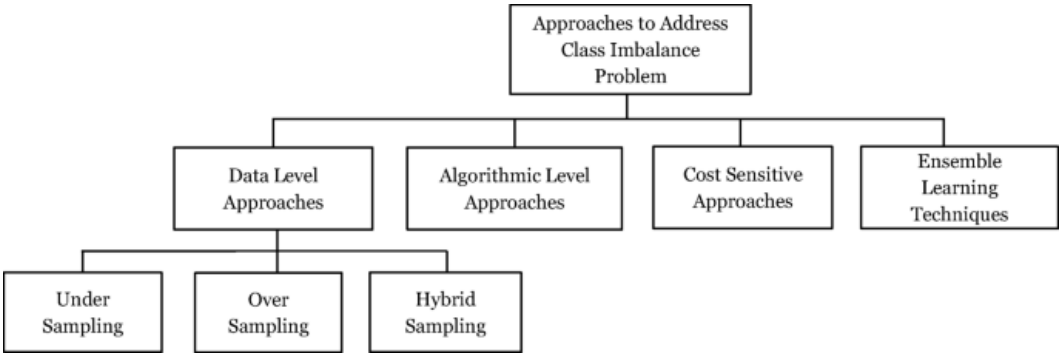


Figure 1.8: Approaches to address imbalanced data classification

To address this issue, numerous methods are now being considered. Three major groupings of these solutions can be identified: data-level algorithms [10], algorithm-level algorithms [7] and algorithms based on cost-sensitive learning [34]. Algorithm-level algorithms consist of developing brand-new algorithms or improving already-existing ones to cope with uneven datasets [77]. Cost-sensitive learning is a strategy combining data- and algorithmic-level approaches while taking into account larger costs for misclassifying samples from the positive class in comparison to the negative ones [5, 48]. Currently, a group of data-level algorithms is most commonly applied. Resampling can be done in three different ways: (a) undersampling the majority class; (b) oversampling the

minority class. (c) a hybrid strategy that incorporates (a) and (b) [59]. The undersampling technique involves taking out certain samples from the initial data set until an equilibrium ratio is obtained. The data to be removed may be chosen at random or, more effectively, in accordance with predetermined criteria, such as the exclusion of samples from the input space's outer regions. Edited Nearest Neighbor (ENN) [120], Tomek link [108], and Random Undersampling (RU) [85] are a few examples of representative undersampling algorithms. On the other hand, the oversampling strategy balances the original database's imbalance by including more samples from the minority class. This can be accomplished by both copying already-existing uncommon samples and producing new ones in a particular area of the input space. Duplication can take the form of random selection or the deliberate selection of samples that fall on the borderline between rare and common samples, forcing the classifier to assign certain spatial regions to the rare class. The SMOTE algorithm [22] is a typically efficient algorithm that has shown the most feasible results in methods related to data preprocessing. However, a significant problem with this method is that it usually produces noise between marginal outliers and inliers, and generating a lot of samples can result in overgeneralization. This issue can be resolved by using undersampling methods (e.g., ENN or Tomek's link) to clean the SMOTE-caused space (i.e., removing some instances from the overlapped areas). Because of this, the SMOTE-ENN and SMOTE-Tomek'slink combinations are two that are widely employed. In [17], a brand-new resampling technique called SUNDO (Similarity-based Undersampling and Normal Distribution-based Oversampling) was developed by the authors. This method combines oversampling and undersampling. The proposed approach shows better performance than the commonly used strategy that combines random undersampling with SMOTE oversampling. Some

publications [35,41] demonstrate that combining undersampling the majority class with oversampling the minority class can improve classifier performance. This is what spurred the author to employ a combined resampling approach to this issue.

1.4.3. Resampling algorithms

a. Synthetic minority over-sampling technique (SMOTE)

Based on the feature space similarities between real minority samples, SMOTE produces synthetic samples. Every sample in the minority class x_i is compared to its k nearest neighbors, with k as a user-defined parameter. These neighbors are determined based on Euclidean distance. The new synthetic instance's feature vector is then determined using the

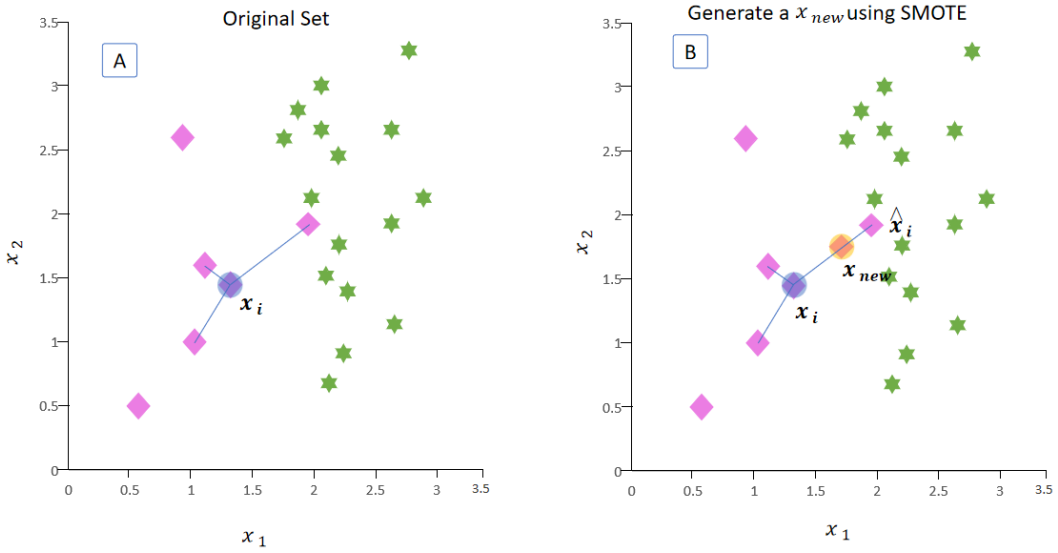


Figure 1.9: An example of generating new instance using the SMOTE algorithm. There are two main steps: the first step selects the K nearest neighbors to the current sample, and the second one chooses one of the K nearest neighbors, then generates a new sample on the line connecting the current sample and the selected neighbor.

formula 1.5:

$$x_{new} = x_i + (\widehat{x}_i - x_i) \text{ x } \delta \tag{1.5}$$

where x_i is the current minority sample.

Finally, at the line between x_i and $\widehat{x}_i$ a new synthetic instance, x_{new} is produced. This technique is repeated until enough instances are produced to reach the user-specified target balancing ratio.

b. SMOTE-Tomek

To understand how this algorithm works, it is necessary to first understand the concept of a Tomek Link. Mathematically, it can be expressed as follows:

Definition 1.8. ***Tomek Link***
Let $d(x_i, x_j)$ stand for the Euclidean distance between x_i and x_j, where x_i denotes sample that belongs to the minority class and x_j represents sample that belongs to the majority class. If there isn't a sample, x_k meets the following condition:

1. $d(x_i, x_k) < d(x_i, x_j), or$
2. $d(x_j, x_k) < d(x_i, x_j)$

then the pair of (x_i, x_j) is a ***Tomek Link****.*

Figure 1.10 shows an illustration of a Tomek link. When two samples are connected by a Tomek link, either one of the samples is a noise or both samples are in close proximity to a border. Following the oversampling process, the classification performance can be enhanced by using this result to remove undesirable overlaps between the minor and major classes. The SMOTE-Tomek technique [9] employs SMOTE to balance the unbalanced dataset first, and then it removes all Tomek linkages from the dataset to eliminate overlapping instances and, as a result, improve the performance of the classifiers with regard to the class imbalance problem.

c. SMOTE-ENN

SMOTE-ENN [10,48] implements SMOTE for the oversampling phase and then employs ENN to remove instances that are overlapping. The

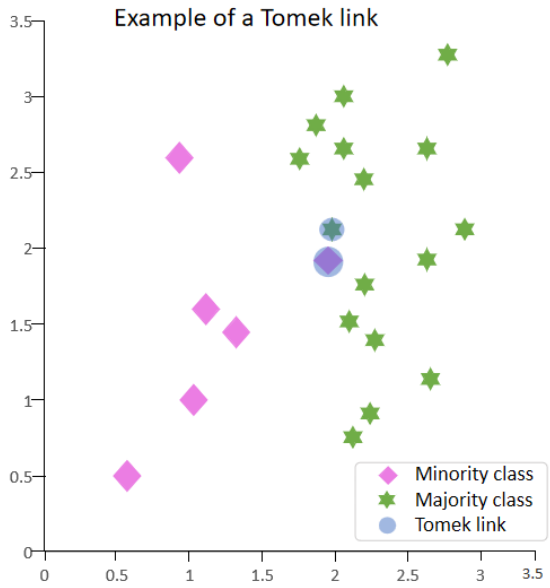


Figure 1.10: An example of Tomek link. When two samples are connected by a Tomek link, either one of the samples is a noise or both samples are in close proximity to a border.

edited nearest-neighbor (ENN) approach uses the nearest-neighbor algorithm to eliminate samples whose class labels don't match those of most of their K-nearest neighbors. In other words, if a high proportion of the neighbors are from the majority class, this sample (from the minority

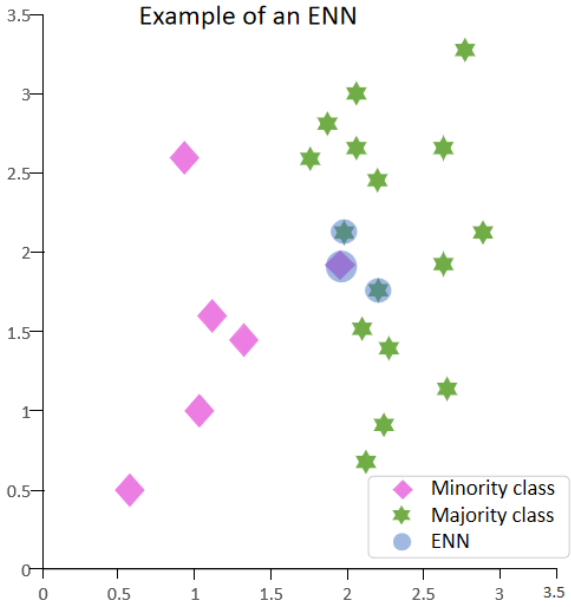


Figure 1.11: An example of ENN.The samples whose class labels don't match those of most of their K-nearest neighbor will be eliminated.

class) and its K-nearest neighbor are eliminated. In this way, ENN helps to clean overlapping occurrences. When the number of nearest neighbors is set to three, for instance, ENN eliminates all instances that don't match two of the three nearest neighbor examples. Figure 1.11 depicts an illustration of an ENN.

It can be seen that when the class of the observation and the majority class of the observation's K-nearest neighbor are different, this method is more effective than Tomek Links because it removes both the observation and its K-nearest neighbor rather than just the observation and its 1-nearest neighbor when the classes are different. As a result, ENN is anticipated to provide more thorough data cleansing than Tomek Links. In this study, SMOTE-ENN is used as a data preprocessing step.

1.4.4. Ensemble learning

Algorithm-level algorithms, in general, can be in the form of a single or a combination of many algorithms in the form of ensemble learning (EL) [48]. A group (or ensemble) of base learners, or models, collaborate to make a better final prediction. This technique is referred to as ensemble learning. Due to significant bias or variation, a single model (sometimes referred to as a base or weak learner) may not perform effectively on its own. However, when weak learners are combined, they might become strong learners since this minimizes bias or variation and improves model performance. *Bagging* (also known as bootstrap aggregation) [13] and *boosting* [47] are the two most common types in ensemble learning.

The main idea of bagging (Figure.1.12A) is to subdivide data into various sub-groups with replacement (meaning that the individual data points can be chosen more than once). These groups are randomly chosen. Following independent training of these weak models, the average or majority of those predictions will produce a more accurate estimate,

depending on the task (regression or classification). As a point of interest, the random forest technique is viewed as an extension of the bagging approach, employing both bagging and feature randomness to produce an uncorrelated forest of decision trees. One thing to keep in mind is that by generating a large number of sub-datasets from which to build models, these models have a high degree of *diversity* (i.e., a large difference between them). This is one of the important factors that will help the ensemble learning model perform better.

One significant distinction between bagging and boosting procedures is that, while models are trained concurrently in bagging, they are taught sequentially in boosting (Figure.1.12B). The fundamental idea behind boosting solutions is that difficult data samples are weighted higher than other data so that in the next iteration the models will focus on processing these samples. This approach is helpful for handling classification issues with an imbalanced dataset. Minority-class data will be viewed as more challenging data, and there is a greater likelihood that the models will choose them over samples from the majority class to concentrate on solving.

One point to emphasize here, EL is a method that combines multiple

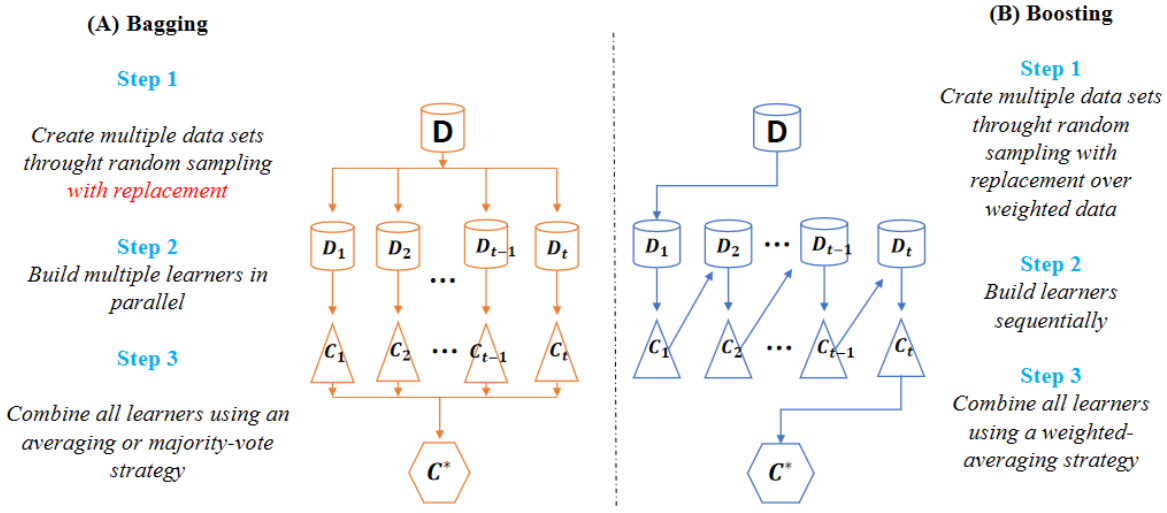


Figure 1.12: Illustrations of (A) bagging and (B) boosting ensemble algorithms [123]

machine learning models to improve prediction performance compared to using a single individual model. To be successful, EL requires diversity in the base models and the ability to efficiently synthesize information from these models. Specifically, EL needs to employ diverse base models (i.e. classifiers). This means that the models should operate differently or be trained on different subsets of data. This helps reduce reliance on a specific model and creates diversity in predictions. While diversity is important, EL also needs convergence, meaning that base models should start from different initial points but ultimately converge to the same optimal solution or close to it. This ensures that the ensemble is not scattered and does not produce contradictory predictions.

1.4.5. C4.5 algorithm

The C4.5 algorithm [103], an improvement on the ID3 algorithm, was put forth by Ross Quinlan in 1991. Based on the normalized information gain, a tree classifier is built using this algorithm. It is thought that this technique, which is frequently applied to classification problems, is "a landmark decision tree program that is possibly the most widely used machine learning tool in practice to date". Three factors led to the selection of this algorithm as the fundamental learner. First of all, this approach is frequently applied to unbalanced data in modern times. The second is its effectiveness and speed of execution. In order to avoid distorting the overall complexity of the approach due to the abundance of individual fitness assessments in evolution, a learner with a quick computation speed must be utilized, and finally, this is the algorithm frequently used in ensemble learning algorithms. One point to note here is that the C4.5 is known to do an internal feature selection process on its own based on the information obtained. By doing a preselection of the variables based on the intrinsic characteristics of the problem, the way the author in this research seeks to aid C4.5 learning.

1.5. Performance evaluation in multi-objective optimization

When measuring the performance of MOEAs, two common factors are considered: convergence (the closeness between the obtained solution set and the true Pareto optimal front) and diversity (the spread and distribution of solutions on the Pareto front). There exist a number of performance metrics to evaluate these factors, such as generational distance (GD) [11], spacing metric (SP) [11], hypervolume (HV) [73] and inverted generational distance (IGD) [64], inverted generational distance plus (IGD+), or stability [29]. Figure.2.9 shows the ability of each metric. The GD and SP metrics evaluate convergence and uniformity, respectively. Meanwhile, the IGD as well as the HV metrics measure not only the convergence but also the diversity of a solution set.

The generational distance (GD) [112] The average distance between a set of evolutionarily discovered solutions (denoted P), and the global POF is known as the GD. The first-norm formula is given as:

$$GD = \frac{\sum_{i=1}^{n} d_i}{n} \tag{1.6}$$

where n is the size of P, and d_i is the Euclidean distance (in objective space) between solution I and the closest solution in the POF. The convergence component of performance is taken into account with this measurement. As a result, it is possible that the set of solutions is extremely close to the POF but does not entirely cover it.

The Inverse generational distance (IGD) [112]: This metric accounts for both convergence and diversity across the entire POF. The following is the IGD first-norm equation:

$$IGD = \frac{\sum_{i=1}^{\overline{N}} \overline{d}_i}{\overline{N}} \tag{1.7}$$

where N is the size of the POF, $\overline{d}_i$ is the Euclidean distance (in objective space) between solution I in the POF and the closest solution in P.

Hypervolume indicator (HV) [131]. For real-world applications, which frequently do not have a POF, GD and IGD are not practical. HV is measured as the hypervolume in an objective space that is dominated by a collection of non-dominated points. Typically, the reference points in objective space are the anti-optimal or worst-possible points. The disadvantage of this measurement is that the computation time is longer than the two IGD and GD measurements.

1.6. Benchmark MOPs

In this thesis, a collection of multi-objective test functions is employed to test the MOEAs' performance. These test functions include multi-modality, non-convexity, and discontinuity problems that are known to be challenging for most MOEAs to solve in general. Some issues include two or three objective functions with disconnected and asymmetric Pareto fronts, which makes it difficult for MOEAs to reach all the regions in the true Pareto front.

1. **ZDT test problems**: these test problems are proposed by [131]. The ZDT set contains convex, concave, non-concave, multi-modal, and disconnected POF.

2. **DTLZ test problems:** [31] have proposed a set of test functions to check an MOEA's ability to converge to the true Pareto front in problems with three or more objectives. This set contains seven test problems.

3. **UF test problems:** For the 2009 IEEE Congress on Evolutionary Computation (CEC) algorithm competition, Qingfu Zhang et al. [126] offered a set of unconstrained (bound constrained) MOP

test instances and a collection of generally constrained test instances. Ten UC test problems are included in this collection.

4. **WFG test problems:** Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston [56] introduced the WFG. There are nine distinct scalable multi-objective unconstrained problems in this collection (both in their objectives and in their decision vectors). Nonseparable problems, deceiving problems, a genuinely degenerate problem, a mixed-shape Pareto front problem, problems with dependencies between position- and distance-related factors, and problems with a scalable number of position-related parameters are all included in the WFG. The WFG test suite offers a more accurate way to evaluate how well optimization methods work on a variety of issues.

More details of these test problems are described in the Appendix 4 of the thesis.

1.7. Summary

In this chapter, the author introduces fundamental knowledge related to the contents used in the thesis. Specifically, definitions of multi-objective optimization as well as typical multi-objective optimization algorithms are introduced. Definitions, methods of co-evolution (specifically, co-operative and competitive co-evolution), related studies on these types of co-evolution, and the relationship between co-evolution and the field of machine learning are also introduced. Next, the problem of classification with imbalanced data, approaches to solving it, and solving algorithms are presented. Finally, standard multi-objective optimization measures and problems are introduced.

Chapter 2

THE DUAL-POPULATION CO-EVOLUTIONARY METHODS FOR SOLVING MULTI-OBJECTIVE PROBLEMS

This chapter studies a very important issue in the field of multi-objective optimization: the balance between convergence and diversity in these algorithms. Two methods introduced in this chapter all use the dual population co-evolution approach. Two populations are utilized in these methods, one of which operates on the Pareto mechanism (which prefers convergence) and the other on the decomposition mechanism (which prefers diversity). This dual population evolution will hopefully produce new solutions that meet both convergence and diversity criteria. The first proposed algorithm is a co-operative co-evolution algorithm, which is an extended version of a prerequisite study. This algorithm allows for improving the probability that a child can be produced from two parents drawn from two populations so that the traits of both parents can be obtained. Besides, the proposed algorithm also has a mechanism to reduce its execution time. The second algorithm uses a competitive co-evolution mechanism. The way in which the child is created as well as the interaction between the two parents in this method are completely different from the methods proposed earlier. By using a solution selection mechanism to create solution parents in each population and then using a competitive mechanism to produce offspring that fulfill both convergence and diversity criteria. The first proposed method was published in [C1] while the second one was published in [J1].

2.1. Introduction

Recently, there have been many studies addressing the problem of balancing convergence and diversity in solving more complex problems such as constrained multi-objective optimization problems (CMOPs) ([126]-[131]), dynamic multi-objective optimization [67], many objectives ([73]-[1]), or ensemble learning problems (with the objectives of maximizing accuracy and diversity of the ensemble). The main idea of these studies is based on a combination of Pareto-based and decomposition-based methods. In [126], the authors used a co-evolutionary algorithm using the two-archive strategy (called C-TAEA) for solving the CMOPs. In particular, C-TAEA utilized two populations, one named the convergence-oriented archive (CA) and the other named the diversity-oriented archive (DA). CA's mission is to maintain convergence and feasibility. The DA, meanwhile, is responsible for preserving the convergence and diversity of the evolution process. The empirical results on benchmark and real-world problems showed the competitiveness of the proposed method in comparison with other state-of-the-art algorithms.

In [69], Ke Li et.al. dealt with convergence and diversity simultaneously by employing ***a dual-population co-operative co-evolution paradigm (named ED/DPP or abbreviated as DPP)***. With the first population, a Pareto-based mechanism was operated in order to maintain a solution set that was satisfactory. The solutions for this population are randomly spread. Regarding the second population, diversity was preserved by the application of a decomposition-based mechanism. In order to guarantee this trait, solutions in this population must be uniformly spread. Finally, a restricted mating selection mechanism (RMS) was employed to harmonize interactions between two co-evolving populations. In the RMS, two mating parents are chosen from both populations. Each of them is restrictively selected from its neigh-

boring sub-regions with a high probability. Because of this selection, there is a possibility that the solution in the first population may not be found. If this happens, a alternative solution can be taken from the corresponding sub-region in the second population. In such a case, both mating parents are selected from the same population, rendering the co-evolutionary mechanism meaningless. Ke Li et.al. and other authors have continued to develop this approach in order to solve additional problems, such as constrained [65], dynamic [23, 66] and combinatorial [16] multi-objective optimization, or even practical applications like SQL injection testing [74], cross-project defect prediction [70] and vehicle routing and scheduling problem [114], as a result of the initial success in using the dual population approach for common multi-objective optimization problems. ***Inspired by the co-evolution paradigm with encouraging results [69]***, ***this study continues to explore in this direction with some improvements***. Specifically, firstly, the author improves this model by proposing a new restricted selection mechanism (named RMS2) and some improvements in the DPP model to shorten the running time as well as achieve better results (this algorithm is known as DPP2). Secondly, a competitive co-evolutionary algorithm is developed to solve multiobjective optimization problems (named DPPCP). The difference between DPPCP and existing studies is detailed as follows: First, this study utilizes an alternative mating selection mechanism instead of the RMS mechanism to select two mating parents. Second, to generate two offspring from the selected parents, this study uses a competitive model instead of the co-operative one.

2.2. The dual-population paradigm (DPP)

Given in Figure.2.1 is the general architecture of the DPP model [69], which employed two co-evolving populations. The Pareto-based mecha-

nism is used in the first population (named A_p) and the decomposition-based mechanism is used in the second population (named A_d). These populations engage in parallel evolution. At each generation, a restricted mating selection mechanism (RMS) allows them to interact with each other.

In the RMS, the mating parent includes three solutions, of which two are selected from A_d and the remaining one is selected from A_p. Thanks to this way, the parents could pass on all the positive characteristics (i.e., convergence and diversity) to the offspring. To update both A_p and A_d, the offspring utilizes the corresponding archiving mechanism.

In the RMS process, there are two cases. In the first case, if no solution is included in the selected sub-region in A_p, an alternative solution will be chosen by the RMS in the corresponding one in A_d. In the second case, if more than one solution is found in the sub-region, only one solution will be selected.

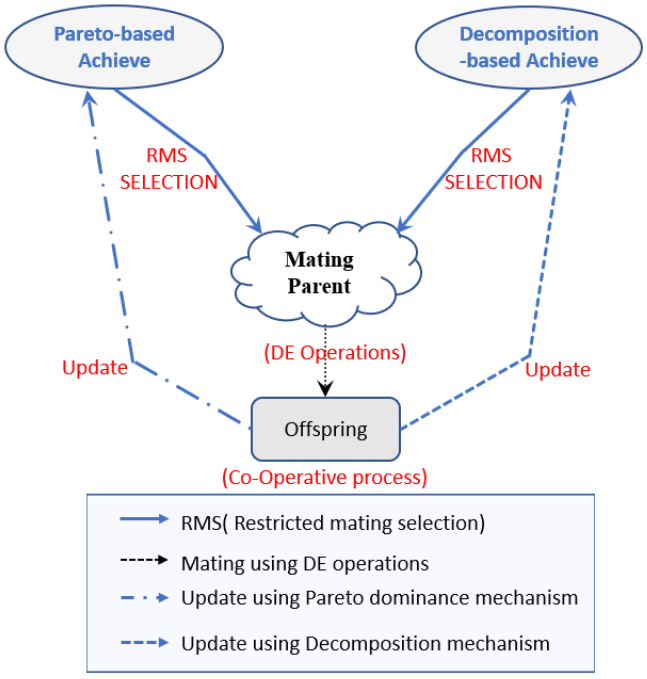


Figure 2.1: The pseudo-code of the DPP algorithm. After selecting three solutions from two populations, DPP uses the mate operator of the DE algorithm to generate an offspring solution. Then, this solution is updated into the two original populations.

This algorithm gives some promising results. However, there are two areas for possible improvement, as discussed below:

1. Restricted mating selection method:

In DPP, a neighborhood of a sub-region is defined as a set of its several closest sub-regions. To take advantage of neighborhood information, the authors specify the neighborhood of each sub-region based on the Euclidean distance between unit vectors. The authors restrict the mating parents to neighboring sub-regions with a high probability (and there is only a low probability that these mating parents will be selected from the whole population). However, they only randomly select a neighboring sub-region from A_p regardless of whether this sub-region contains any solutions in the A_p or not. This leads to a high possibility that the selected sub-region does not contain any solutions (so an alternative solution has to be borrowed from the corresponding sub-region in A_d). ***This may lead to an imbalance between the two populations.***

2. The interaction between two co-evolving populations:

In DPP, the authors define interaction as the way to generate offspring from mating parents. To be specific, they use differential evolution (DE) for offspring generation. This means they need three solutions (such as, x_{r1}^G, x_{r2}^G, x_{r3}^G, where x_{r3}^G is the current solution, x_{r1}^G is a solution selected from A_p and x_{r2}^G is a solution selected from A_d) to create new offspring (x_i^{G+1}).

$$x_i^{G+1} = x_{r3}^G + F * (x_{r1}^G - x_{r2}^G) \tag{2.1}$$

It is worth noting that in Eq.2.1, $F * (x_{r1}^G - x_{r2}^G)$ is a direction vector. This vector is vital because it may help to direct the current vector to a new location that is closer to the global extremes, or maybe even make it move further away from this position. Take Figure.[2.2] as an example. In case 1, using Eq.2.1, from the three parents x_{r1}^G, x_{r2}^G, x_{r3}^G, we can obtain an offspring solution x_i^{G+1} whose position is closer to the global

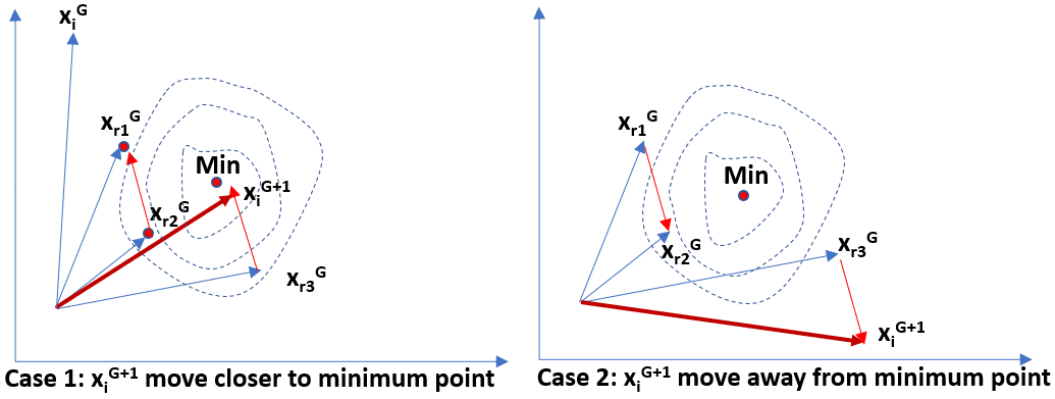


Figure 2.2: The way to generate offspring from mating parents using DE operators

extreme position (denoted by *Min*) than its parents. On the contrary, in case 2, also using Eq.2.1, although the parents x_{r1}^G, x_{r2}^G, x_{r3}^G are close to *Min*, the offspring solution x_i^{G+1} is actually further away from *Min* than its parents. In DPP, the authors select x_{r3}^G and x_{r1}^G from A_d and x_{r2}^G from A_p with the hope that x_{r1}^G has good convergence properties and x_{r2}^G has promising diversity. In this way, we have a large chance of generating offspring that have both advantages. ***However, there still exist two major drawbacks:***

(+) Choosing two out of three solutions from the A_d and only one from the A_p may cause an imbalance in the co-evolutionary process.

(+) Since the direction vector is made up of two solutions in two different populations, it could lead to unpromising outcomes, especially when the two populations are imbalanced (i.e., the convergence of one population is much better than the other). Let us consider a simple example in Figure.2.3. x_{r2}^G is quite close to the Pareto front. Meanwhile, x_{r1}^G is far from the Pareto front. Suppose that we are running with the A_d population. By iterating over each sub-region, for each sub-region (assuming the current sub-region contains x_{r3}^G), the author makes a random selection of two neighboring sub-regions (e.g., NB1, NB2). In these 2 sub-regions, NB1 contains a solution (e.g., x_{r2}^G), while NB2 does not contain any solutions. In this case, NB2 will borrow a solution in the

corresponding sub-region for the A_p population (e.g., x_{r1}^G). After mating, based on Eq.2.1,we might obtain the offspring x_i^{G+1}. It can be seen that x_i^{G+1} has shifted to a position that is far from the Pareto-front. This leads to poorer results.

This study attempts to address the aforementioned draw-

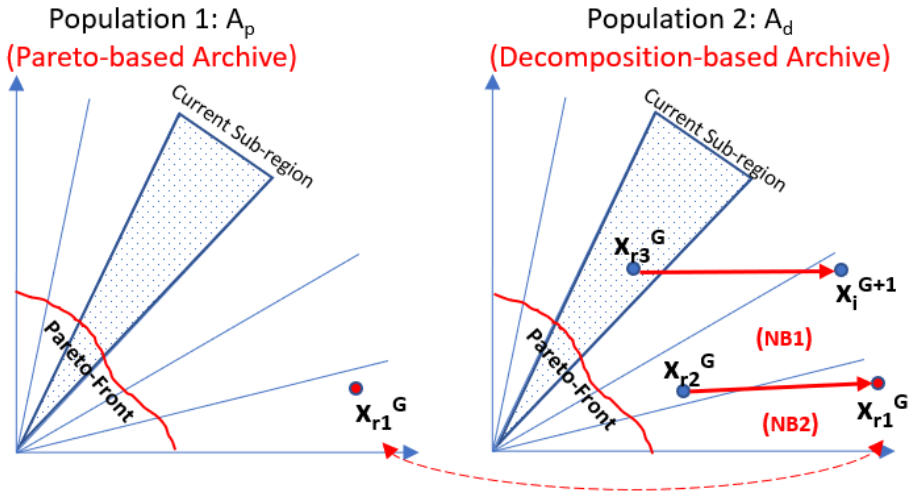
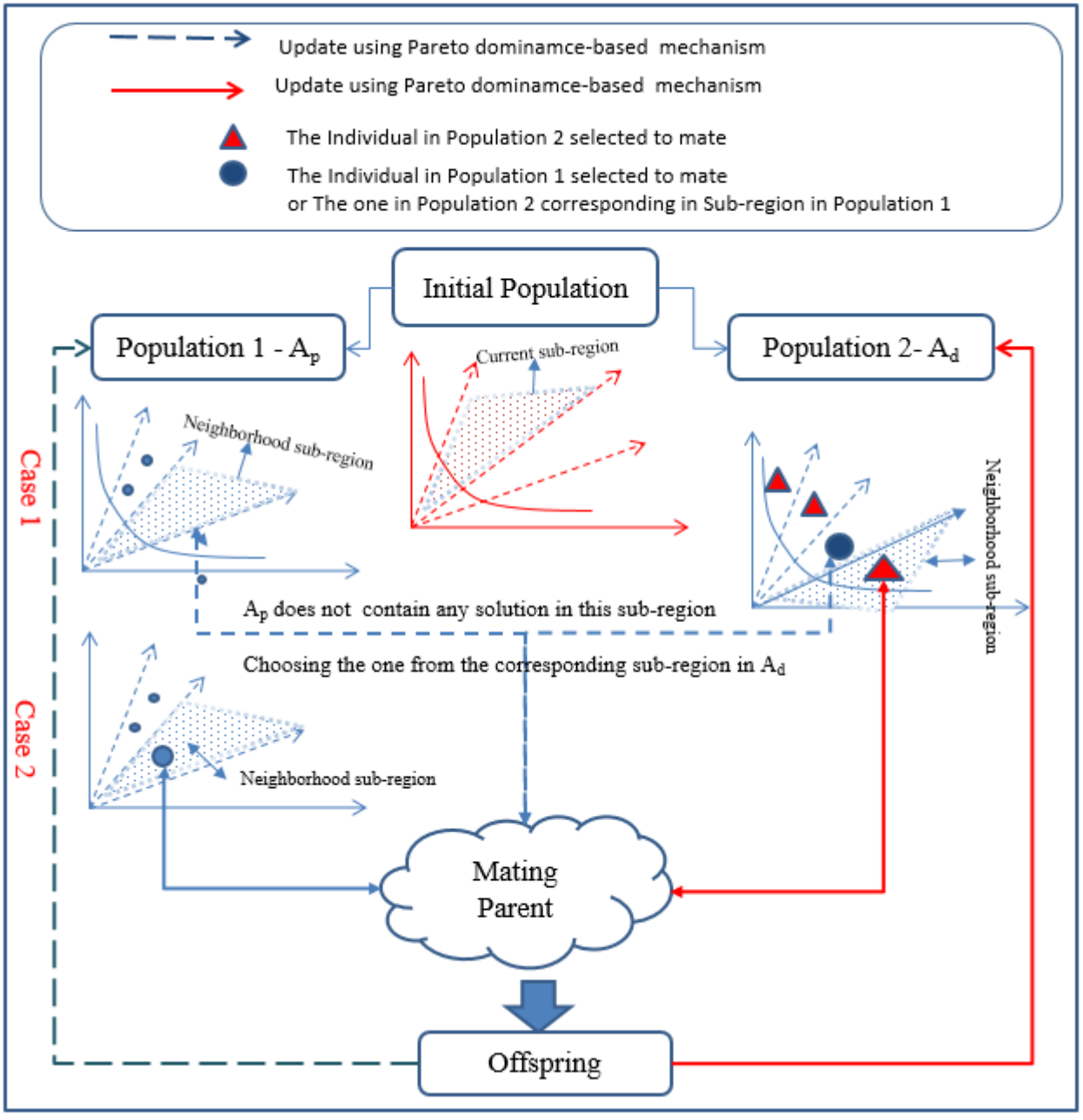


Figure 2.3: A simple illustration of generating offspring from mating parents

backs. The author proposes a new dual-population competitive co-evolutionary algorithm named DPPCP (*The dual-population competitive co-evolutionary algorithm*). This algorithm differs from the DPP model in two ways. First, it uses competitive co-evolution rather than co-evolution to interact between two co-evolving populations. Second, it uses a neighbor-based selection mechanism (NBSM) instead of the RMS to select three different solutions for each distinct population. These two proposed algorithms (i.e., DPP2 and DPPCP) are explained in more detail in the next sections.

2.3. A dual-population co-operative co-evolutionary method for solving multi-objective problems (DPP2)

The general diagram of the modified dual-population algorithm (DPP2) is given in Figure.2.4 and the pseudo-code of this algorithm is shown in



0.9

Figure 2.4: Diagram of the DPP algorithm. In the first case, the selected neighborhood sub-region does not contain any solution (the alternative solution is selected from the corresponding sub-region in A_d), whereas in the second case, this sub-region contains at least one solution (a random solution in this sub-region is selected).

Algorithm.5 A more detailed explanation of DPP will be shown here.

In the first step, A_p and A_d (for simplicity, they have the same size N) are randomly initialized. N solutions in A_d are evenly assigned to N sub-regions (according to N unit vectors). Later, in the process of evolution, each sub-region always has only one solution. This is to guarantee that A_d always has an even distribution (i.e., diversity) in objective space. Whereas, N solutions in A_p will be randomly assigned to N sub-regions. This means that more than one solution can be in the same sub-region, and there are also sub-regions that don't contain any solutions. Next, each solution specifies the T closest neighborhood sub-regions based on the Euclidean distance between unit vectors.

As mentioned previously. In [124], the authors used an RMS mechanism to select two mating parents. Ideally, one solution is selected from Ap and the other from Ad. However, there is no guarantee that each solution in A_p is associated with a sub-region. Hence, when A_p does not contain any solution in the selected sub-region, RMS utilizes an alternative solution from the corresponding sub-region in A_d (see case 1 in Figure.2.4). At this time, the offspring are generated from parents in the same population. As a result, the offspring cannot take advantage of both populations. This might lead to an imbalance between diversity and convergence.

After the selection process, two mating parents (denoted to x_{r2} and x_{r3}) will be selected for the co-operative process. To generate new offspring from these mating parents, the author borrows the reproduction idea from MOEA/D-DE [64].

One thing to be underlined here is that the new offspring need to be assigned to a certain sub-region. In this research, this offspring belongs to the sub-region that has the minimum Euclidian distance between its unit vector and the offspring's objective vector.

Finally, the new offspring is used to update each of A_p and A_d, respectively.

In general, the DPP2 has three main differences from the DPP:

First, when choosing one solution in A_p, instead of just selecting from a selected neighborhood sub-region, the author will select from all T neighborhood sub-regions. By doing this, the probability of finding one solution in A_p will be much higher than in RMS.

Second, in case all T neighborhood sub-regions do not contain any solution. Instead of choosing an alternative solution in Ad, the author randomly selects a solution in A_p. In this way, the offspring are generated from parents in different populations, so they can take advantage of all the advantages of both parents (i.e., diversity and convergence).

Third, the update procedure for A_p is different from the original DPP. In particular, whenever a new offspring is generated, it will be stored in an offspring list (i.e., $offSpringAp$ in Algorithm.2.4) instead of being updated right away to A_p. After a generation finishes, $offSpringAp$ will be combined with A_p and the author uses the crowding distance sorting method (CDSM) in the combined population to select the N best solutions for the new population. The reason is that the CDSM is a really time-consuming method. Assuming that the maximum number of generations is M = 300.000 and the population size is N = 300, then there are 300.000 new offspring generated. Thus, the CDSM will be called 300.000 times in the DPP. Meanwhile, 1000 is the number for DPP2. Apparently, with this new update mechanism, the computing time has been greatly reduced (i.e., N times).

In summary, based on the above discussion, we can conclude that DPP2 has, in principle, many potentials for tackling this kind of problem.

Algorithm 5: DPP2 algorithm

input : Maximum number of generations (M)
Neightborhood Size (T)
Population size (N)
output: Final Population P

$[A_p, A_d] = initializePopulation()$
$W = InitializeUniformWeight()$
$B = InitializeNeighborhood()$
$Z^* = InitializeIdealPoint()$
$Z^{nad} = InitializeNadirPoint()$
$m \leftarrow 0$
while $m < M$ **do**
  $offspringAp \leftarrow \varnothing$
  for $i \leftarrow 1$ **to** N **do**
    Q= RMS2 (A_p, A_d, m, Bm) (*Algorithm* 6)
    $Child = CoOperativeMating(Q)$
    Mutate(Child);
    Update Sub-Region index for *Child*
    Update Idea point Z^* and nadir point Z^{nad}
    Update A^d
    Add Child to *offspringAp*
    m++;
  $U =$ **Union**($offspringAp, A_p$)
  $Ap =$ **crowdingDistanceSelection**(U)

Algorithm 6: RMS2(Ap,Ad,m,Bm)

input : A_p (Pareto-based archive)
A_d (Decomposition-based archive)
m: current subregion index
B_m: A Set contains neighborhood indexes of current subregion.
T: the neighborhood size;
N: the Population size
output: Two mating parent(Q).

$P_1 =$ **MatSelectionAp**() (*Algorithm* 7)
$P_2 =$ **MatSelectionAd**() (*Algorithm* 8)
$ReturnQ = (P1, P2)$

Algorithm 7: MatSelectionAp(A_p,m,B_m)

input : A_p(Pareto-based archive)
m: current subregion index
B_m: A Set contains neighborhood indexes of the current subregion.
T: the neighborhood size;
N: the Population size
output: A subregion index.

```
listNeighborAp ← ∅
if rand < neighborhoodSelectionProbability then
    // Select a sub-region index in A_p
    for i ← 0 to T do
        for j ← 0 to N do
            if Ap[j] ∈ B_m[i] then
                Return j;
else
    Randomly select an index from {1, 2, ..., N}
```

Algorithm 8: MatSelectionAd(A_d, m, B_m)

input : A_d(Decomposition-based archive)
m: current subregion index
B_m: A Set contains neighborhood indexes of the current subregion.
T: the neighborhood size;
N: the Population size
output: a subregion index.

```
if rand < neighborhoodSelectionProbability then
    Randomly select an index from B_m
else
    Randomly select an index from {1, 2, ..., N}
```

2.4. The dual-population competitive co-evolutionary method for solving multi-objective problems (DPPCP)

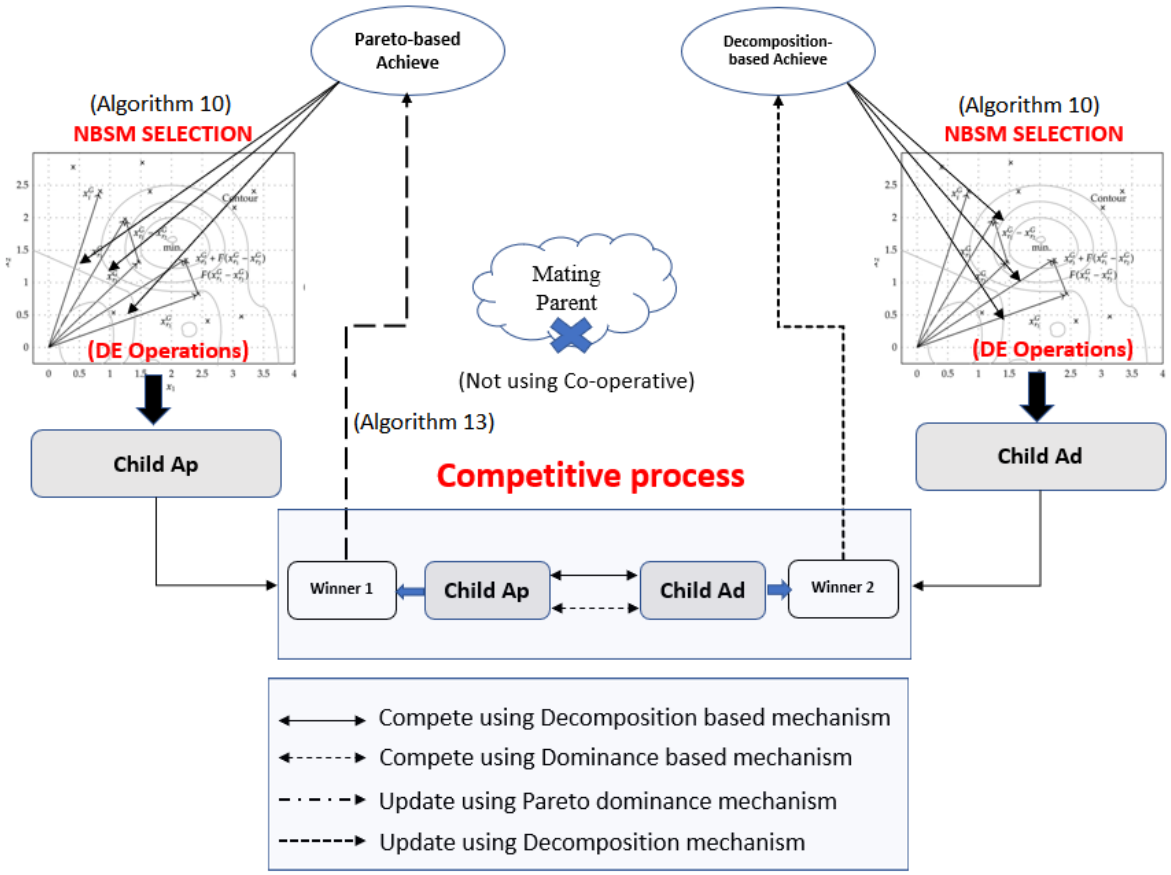


Figure 2.5: System architecture of the dual-population competitive co-evolutionary method. Each population selects three solutions to create offspring. Then, these two offspring compete against each other using two different mechanisms. The winner of the competition is selected to update the population using the corresponding mechanism.

The general diagram of the DPPCP is given in Figure.2.5 and the pseudo-code of the proposed algorithm DPPCP is shown in Algorithm 9. There are two co-evolving populations: the first one (named A_p) evolves by using the Pareto-based mechanism; the other one (named A_d) utilizes the decomposition-based mechanism to evolve. At each generation, the author uses a neighbor-based selection mechanism (NBSM) to select three candidate solutions from each of the populations. After that, the author uses differential evolution (DE) to create two offspring named $Child_{Ap}$ (i.e., the offspring in population A_p) and $Child_{Ad}$ (the

offspring in population A_d). Next, let $Child_{Ad}$ compete with $Child_{Ap}$ using Pareto dominance-based metrics and choose the winner to update A_p. Similarly, let $Child_{Ap}$ compete with $Child_{Ad}$ using decomposition-based metrics and use the winner to update A_d. At the end of the co-evolution process, the final population is a combination of both A_p and A_d populations. The reason for this decision is that each of them uses a different optimal mechanism. While A_p uses the true Pareto front, A_d utilizes the idea point (a solution with the best objective values known since running the algorithm) as the best goal to achieve. The roles of the two populations are the same. Therefore, in order to preserve the good properties of both populations (i.e., diversity and convergence), the author decided to keep both populations in the final selected population.

As mentioned above, there are two differences between the DPPCP model and the DPP model: *First*, in the DPPCP, the author does not use a co-operative co-evolutionary mechanism. In other words, this study has eliminated the mating parents' steps to generate offspring. Instead, this study uses a competitive mechanism to make two offspring interact with each other. *Second*, this study uses the NBSM mechanism to select three solutions in each population and use them to create two separate offspring. In general, the model is divided into four main steps: *Initialization, NBSM selection, Competitive process, and Update population.*

Initialization

In the first step, A_p and A_d (with the same size N) are randomly generated. However, the distribution of solutions in the two populations is different. In A_d, N solutions are assigned to different N sub-regions. To make sure that there is only one solution for each sub-region, the algorithm divides the original region into N sub-regions (denoted as S_i) by using N uniformly distributed unit vectors denoted as λ_i (see Figure.2.6).

Algorithm 9: DPPCP Algorithm

input : M: The number of generations.
T: The neighboring numbers
N: The population size
output: Final Population A_p and A_d

$[A_p, A_d]$ = *initializePopulation()*
W = *InitializeUniformWeight()*
B = *InitializeNeighborhood()*
Z^* = *InitializeIdealPoint()*
Z^{nad} = *InitializeNadirPoint()*
$m \leftarrow 0$
while $m < M$ **do**
  offspringAp $\leftarrow \varnothing$
  for $i \leftarrow 1$ **to** N **do**
    $Child_{Ap}, Child_{Ad}$ = **NBSMSelection**(A_p, A_d, i, Bi) (*Algorithm* 10)
    $Winner1$ == **CompeteDominate**$(Child_{Ap}, Child_{Ad})$
    $Winner2$ == **CompeteDecompostion**$(Child_{Ap}, Child_{Ad})$
    UpdateAp(Winner1, A_p); (*Algorithm* 13)
    UpdateAd(Winner2, A_d);
    Update Z^* and Z^{nad}
    m++;
Return $P \leftarrow A_p \cup A_d$

Each λ_i will be identified as corresponding to $solution_i$ (or each solution will be assigned to only one sub-region). The algorithm utilizes the λ vectors to calculate the Euclidean distance between these vectors. Based on these distances, the algorithm can determine which sub-regions are neighbors of a solution. In the next step (i.e., the evolutionary step), when a new solution is created, it is necessary to determine which sub-region it belongs to. This is done based on the calculation of the distance between the new solution and the λ vectors. A sub-region will be selected if it contains the λ vector that is closest to the new solution. However, it should be noted that, instead of including this new solution directly in this sub-region, a competition between the new solution and the existing solution in this sub-region will take place. The better solution (based on the fitness functions) will be selected and assigned to this sub-region. In this way, there is exactly one solution in each sub-region, and A_d is

distributed evenly (i.e., with diversity) in the objective space. Unlike A_d, A_p does not rely on the even spread of N unit vectors. Therefore, N solutions in A_p are randomly assigned to N sub-regions (Figure.2.7 gives an intuitive example of the distribution of solutions in each population. A_p does not contain any solution in sub-regions 0, 1, 3, 5, while sub-regions 2 and 4 contain more than one solution). This leads to the situation that a sub-region may either not have any solutions or contain more than one solution. Next, this study finds the T closest neighborhood sub-regions for each solution (by using the Euclidean distance). These neighborhoods play a vital role in the next steps.

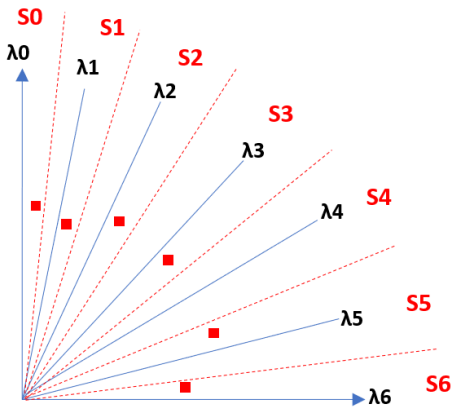


Figure 2.6: A simple illustration of initializing the population for A_d. The algorithm divides the original region into N sub-regions. N solutions are assigned to different N sub-regions

The neighbor-based selection mechanism (NBSM)

In [130] the authors showed that, when solving continuous MOPs, in some mild conditions, neighborhood solutions should have similar structures. This means the neighborhood information is very important, and it would be better if we used this important information in the orientation process for new solutions. For that reason, this study prefers to choose mating parents from several neighboring sub-regions. As for the traditional DE operator, [57], x_{r1}^G and x_{r2}^G (two components of the direction vectors in Eq.2.2) are randomly selected from the whole popu-

Algorithm 10: NBSMSelection(A_p,A_d,i,B_i)

input : A_p: the Pareto-based population
A_d: the decomposition-based population
i: the current sub-region index
B_i: a set contains neighborhood indexes of the current sub-region.
T: the neighborhood size;
N: the population size

output: Q: Two mating parents

$(r_{1p}, r_{2p}) =$ **MatSelectTwoAp**() (*Algorithm* 11)
$(r_{1d}, r_{2d}) =$ **MatSelectTwoAd**() (*Algorithm* 12)
$Solution1 = A_p[r_{1p}]$
$Solution2 = A_p[r_{2p}]$
if *SubRegion*(r_{1p}) *does not contain any solutions* **then**
    $Solution1 = A_d[r_{1p}]$
if *SubRegion*(r_{2p}) *does not contain any solutions* **then**
    $Solution2 = A_d[r_{2p}]$
$Child_{A_p} =$ **DE**($Solution1, Solution2, A_p[i]$)
$Child_{Ad} =$ **DE**($A_d[r_{1d}], A_d[r_{2d}], A_d[i]$)
Return Q = ($Child_{A_p}, Child_{A_d}$)

lation. This random mating selection mechanism can be explored well. However, since there is no guidance information regarding the Pareto set, it may lead to a degeneration problem. The RMS mechanism in [65] improved this weakness by using more information from neighboring sub-regions than from the whole population. However, as mentioned above, a drawback of the RMS mechanism is that the probability of selecting a sub-region in A_p that contains at least a solution is relatively low. At that time, the RMS borrows an alternate solution in the A_d, which can lead to an imbalance between two populations in the co-evolutionary process. This is the reason why this study proposes another selection mechanism (i.e., the NBSM).

The pseudo-code of the NBSM mechanism is presented in Algorithm 10.

There are two underlying principles of the NBSM. Firstly, the author wants fairness in choosing the number of solutions to hybridize in the co-evolutionary step. Secondly, the three chosen solutions used in the

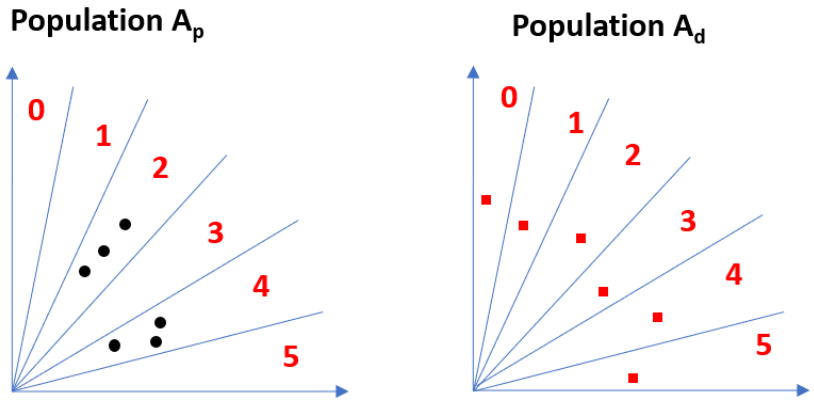


Figure 2.7: A simple illustration of the distribution of solutions in sub-regions. While in the A_d population, each partition has only one solution, in the A_p population, there are partitions without any solutions, and some partitions have more than one solution.

Algorithm 11: MatSelectTwoAd(A_d, i, B_i)

input : A_d: the decomposition-based population
i: the current sub-region index
B_i: a set contains neighborhood indexes of the current sub-region.
T: the neighborhood size;
N: the population size
θ: the neighborhood selection probability

output: [I,J]: Two sub-region indexes.

if $rand < \theta$ **then**
  Randomly select two indices, I and J, from B_i
else
  Randomly select two indices I and J from $\{1, 2, ..., N\}$

DE operator must be on the same population (in order to avoid the phenomenon as shown in Figure 2.3).

To generate new offspring (i.e. $Child_{Ap}$ or $Child_{Ad}$), this study imitates the idea from MOEA/D-DE [64]. Specifically, in MOEA/D-DE, a solution $\overline{y}$ is generated from x^{r1} (i.e. the current solution), x^{r2} and x^{r3} according to Eq.2.2, and a new solution is generated by a mutation operator on $\overline{y}$ with a small probability, according to Eq.2.3

$$\overline{y}_k = \begin{cases} x_k^{r1} + F * (x_k^{r2} - x_k^{r3}), & \text{with probability} < \text{CR} \\ x_k^{r1}, & \text{with probability 1-CR} \end{cases} \tag{2.2}$$

Algorithm 12: MatSelectTwoAp(A_p,i,B_i)

input : A_p: the Pareto-based population
i: the current sub-region index
B_i: a set contains neighborhood indexes of the current sub-region.
T: the neighborhood size;
N: the population size
θ: the neighborhood selection probability

output: [I,J]: Two sub-region indexes.

```
listNeighborAp ← ∅
if rand < θ then
    // Select two sub-region indexes in A_p
    for i ← 0 to T do
        for j ← 0 to N do
            if A_p[j] ∈ B_i[i] then
                Add j to listNeighborAp;
    while size of listNeighborAp < 2 do
        Randomly select an index r from 1,2....,N
        Add r to listNeighborAp
    Randomly select two indices, J and K, from listNeighborAp.
else
    Randomly select an index from {1, 2, ..., N}
```

where CR and F are two control parameters

$$y_k = \begin{cases} \overline{y}_k + \sigma_k * (u_k - l_k), & \text{with probability } p_m \\ \overline{y}_k, & \text{with probability 1-}p_m \end{cases} \tag{2.3}$$

$$\sigma_k = \begin{cases} (2 * rand)^{\frac{1}{\eta}+1} - 1, & \text{if } rand < 0.5 \\ 1 - (2 - 2 * rand)^{\frac{1}{\eta}+1}, & \text{otherwise} \end{cases} \tag{2.4}$$

where *rand* is a uniform random number in [0, 1], p_m is the mutation rate, and u_k and l_k are the upper and lower bounds of the k^{th} decision variable, respectively.

Another major difference between the two RMS and NBSM mechanisms is the solution selection procedure in A_p. For each small partition, this study conducts a search across the entire T neighborhood sub-regions instead of just choosing a random sub-region, as in the RMS

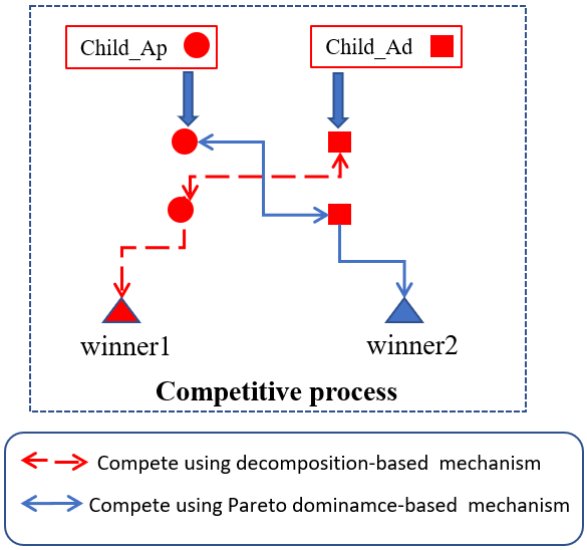


Figure 2.8: The competitive mechanism

mechanism. This way, the probability of finding three solutions is much higher. In the case that any solutions cannot be found in the neighborhood sub-regions, this study borrows from the A_d.

The competitive co-evolutionary mechanism (Competitive process)

In this step, two offspring solutions, $Child_{Ap}$ and $Child_{Ad}$ in each population are selected to participate in tournaments. Figure.2.8 gives an intuitive explanation of this mechanism. Specifically, $Child_{Ap}$ competes against $Child_{Ad}$ using the Pareto-based rule (i.e., *CompeteDominance* method in Algorithm 9), a winner is selected to update A_p. Meanwhile, $Child_{Ad}$ competes against $Child_{Ap}$ using the decomposition-based rule (i.e., *CompeteDecomposition* method in Algorithm 9). The author would like to highlight the benefits of competitive co-evolution by considering two possible cases:

(+) If two winning solutions belong to two different populations, it means we have one solution with good convergence and another with good diversity. This is what this study expected.

(+) If two winning solutions belong to the same population (e.g., A_p). It means one solution in A_p has better convergence than its competitor

(this is normal). Along with that, the remaining solution in A_p is more diverse than its contestant in A_d. It is interesting to note that, in this case, we have one mating solution that has good convergence and another that has not only good convergence but also excellent diversity. Therefore, the offspring can inherit both good traits. This is in stark contrast to the DPP's cooperative co-evolution.

Update Population

Algorithm 13: UpdateAp (winner1, A_p)

input : *LimitedNum*: The limited number of updated times
N: the Population size
output: A_p after updated

```
1  isNonDominte= False ; Flag = False;
2  for i ← 0 to N do
3      if winner1 dominate A_p[j] then
4          Update A_p[j] by winner1;
5          Flag = True;
6          Num++;
7          if Num = LimitedNum then
8              Break;
9          else if winner1 and A_p[j] are nondominated then
10             isNonDominte = True;
11         ;
12 if isNonDominte = True and Flag = False then
13     Add winner1 to A_p
14     A_p = crowdingDistanceSelection(A_p)
15 else
16     Randomly select an index from {1, 2, ..., N}
```

The update mechanism in each population will be different. In [65], the authors only updated Offspring to the nearest sub-region. This is to ensure population diversity, but the probability that this solution will replace the sub-region is rather small (because it only compares to only one sub-region, while there are some other sub-regions having much worse solutions). This may lead to the possibility that convergence will decrease. To improve this disadvantage, the author used the updated

idea of the MOEA/D algorithm for both populations. Specifically, this study will iterate through all solutions in the neighborhood sub-regions in question and update them a limited number of times (this helps to avoid having many similar solutions as well as speeding up the convergence of the population).

Specifically, the *UpdateAd (winner2,A_d)* method in algorithm 9 is used in the same way as the MOEA/D-DE algorithm; whereas the *UpdateAp (winner1,A_p)* method is shown in the algorithm 13.

It can be easy to see that, the way to update A_p is different from the one in the NSGA-II algorithm. this study updates A_p as soon as the winner dominates a solution in A_p and conducting *Ranking* and *CrowdingDistanceSelection* methods every time updating offspring. Meanwhile, the NSGA-II uses a list to store all of the offspring and performs ranking when the loop is finished.

It is highlighted that the new offspring requires being assigned to a certain sub-region. In this study, to determine the suitable sub-region, the authors first measure the distance between its unit vector and the offspring's objective vector. Whichever sub-region has the shortest distance will be selected to contain the offspring. It is also noted that the scaling of each objective function differs from the other. It means the value of objective functions can vary from low to high. This leads to the situation that, in the calculation of the Euclid distance, the low objective value is of no importance. Therefore, it is essential to standardize the objective functions within the same range of values. Here, this study performs the standardization [28] to the interval [0, 1] as follows:

$$f_i^{norm} = \frac{f_i - Z_i^*}{Z_i^{nad} - Z_i^*} \tag{2.5}$$

Where i $\in \{1, 2, ..., m\}$; m is the number of objective functions.

The above analysis shows that basically A_p is similar to NSGA-II, and A_d is similar to MOEA/D in terms of how they work, but the details

of the implementation are quite different. In light of the experimental results, this study will further analyze these differences.

2.5. Experimental design

This step will perform experiments with the proposed algorithm to clarify the following problems: First, this study compares the proposed methods with some baseline algorithms (i.e., NSGA-II and MOEA/D-DE) and the state-of-the-art algorithm (DPP in [65]). Via the comparison results, we could see how good the performance of the proposed method was when compared to the others. Next, this study develops a variant named *DPPCP-Variant1* and compares it to the DPPCP to know the effects of competitiveness. In order to know the effects of the NBSM mechanism, this study creates two variants named *DPPCP-Variant2* and *DPPCP-Variant3* and compares them to DPPCP. Finally, to know the interaction between two co-evolving populations, this study creates two other variants named *DPPCP-Ap* and *DPPCP-Ad*. After that, this study conducts three test cases between NSGA-II and A_p; MOEAD and A_d; and *DPPCP-Ap* and *DPPCP-Ad*.

2.6. Test problems

In this study, 31 test instances (ZDT1 to ZDT6, UF1 to UF10, WFG1 to WFG9, and DTLZ1 to DTLZ7) are used as benchmark problems. Among these, UF1 to UF7 [69] and WFG1 to WFG9 [91] and ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [134] are the bi-objective test problems (the ZDT5 is not discussed or utilized in this thesis due to it is a binary problem), UF8 to UF10 and DTLZ1 to DTLZ7 are the tri-objective benchmarks. More detailed test problems are described in the Appendix 4 of the thesis. Table 2.1 shows some summary information on the DTLZ problems.

Table 2.1: The DTLZ series test instances

Name	Variable number	Objective number	Geometry
DTLZ1	7	3	Linear
DTLZ2	12	3	Concave
DTLZ3	12	3	Concave
DTLZ4	12	3	Concave
DTLZ5	12	3	-
DTLZ6	12	3	-
DTLZ7	22	3	Disconnected

2.6.1. Performance metrics

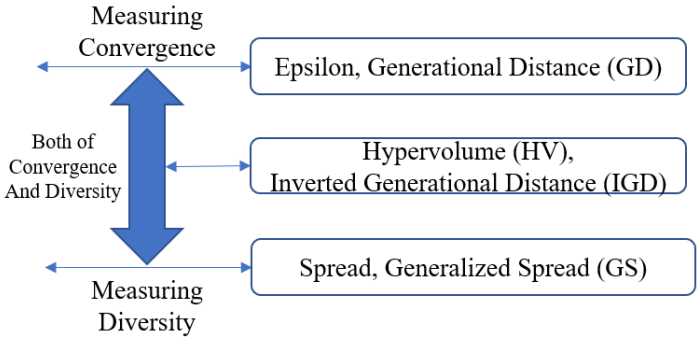


Figure 2.9: Several performance metrics are used in MOEAs

In this study, the IGD and HV are chosen as the main metrics. It is worth highlighting that the quality of a solution depends on the HV value. The greater the HV value, the better the solution is, while a lower IGD value indicates a better result.

2.6.2. Parameters settings of MOEAs

Table 2.2: The parameter setting of the MOEAs

MOEAs	Parameters settings
NSGA-II	pc=0.9, pm=$1/n_{variables}$; $\mu_c = 20; \mu_m = 20$
MOEA/D-DE	pm=$1/\ n_{variables}$, $\mu_m = 20$, CR=1.0, F=0.5, $\sigma = 0.9$, T=20, $'rand/1/bin'$; $n_{replaced}$=2

Given in table 2.2 are the parameters of the NSGA-II and MOEA/D-DE. In each test trial, every algorithm is independently run 20 times. The population size (N) is set to 300, and the termination criterion of an algorithm is a predefined number of generations (M), which is

constantly set to 1000. These parameter values are selected similarly to the previous prerequisite studies [69] to facilitate the comparison of execution performance between algorithms.

2.7. Results and discussions

2.7.1. Comparing with state-of-the-art algorithm

In [65], the authors compared the DPP algorithm with some state-of-the-art algorithms (e.g. MOEA/D-FRRMAB [67], MOEA/D-RMS [73], MOEA/D-M2M [73], D^2 MOPSO [1] and HyPE [6]) with competitive results. Therefore, DPP can be considered a state-of-the-art algorithm. In this experimental scenario, the author focuses on comparing the DPPCP with the original algorithm, the DPP, and the DPP2 algorithm.

The results in Table 2.3 and Table 2.4 show that the proposed method (DPPCP) is clearly better than DPP and DPP2 (it gives a better metric value in 24 out of 31 comparisons). In ZDT instances, DPPCP gives better results than DPP in all instances, especially in ZDT4, where DPPCP outperforms DPP about 10,000 times. In UF instances, DPP achieves better performance on UF5 and UF6 instances. However, DPPCP obtains better IGD metric values in other UF instances; even with UF1, it is better about 100 times. Similar to WFG instances, DPPCP achieves better metric values in all of the comparisons (except WFG5). These findings demonstrate that the competitive co-evolution model proposed in this study outperforms co-operative co-evolution methods in ([65], [104]).

2.7.2. Comparing with baseline algorithms

As mentioned above, the DPPCP uses two populations, one based on the Pareto mechanism (using the NSGA-II algorithm) and the other

Table 2.3: Performance comparisons between the proposed algorithms with state-of-the-art algorithm using the HV metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPP	DPP2	DPPCP
ZDT1	$6.503543e-01_{1.9e-02}$	$6.648341e-01_{4.6e-05}$	$6.655793e-01_{0.0e+00}$
ZDT2	$3.062317e-01_{7.8e-02}$	$3.315689e-01_{3.8e-05}$	$3.321892e-01_{0.0e+00}$
ZDT3	$5.137847e-01_{2.3e-03}$	$5.162233e-01_{9.3e-06}$	$5.170450e-01_{0.0e+00}$
ZDT4	$0.000000e+00_{0.0e+00}$	$6.649716e-01_{1.0e-05}$	$6.655913e-01_{0.0e+00}$
ZDT6	$4.052969e-01_{1.9e-05}$	$4.047282e-01_{5.4e-08}$	$4.053136e-01_{0.0e+00}$
UF1	$6.095147e-01_{1.5e-02}$	$6.635853e-01_{1.4e-04}$	$6.639659e-01_{0.0e+00}$
UF2	$6.071246e-01_{9.4e-03}$	$6.568236e-01_{1.6e-03}$	$6.616692e-01_{0.0e+00}$
UF3	$4.299732e-01_{5.4e-02}$	$6.521115e-01_{1.5e-02}$	$5.965543e-01_{0.0e+00}$
UF4	$2.384583e-01_{7.3e-03}$	$2.438856e-01_{4.2e-03}$	$2.551797e-01_{0.0e+00}$
UF5	$1.642984e-03_{6.7e-03}$	$1.208023e-01_{9.1e-02}$	$8.999985e-02_{0.0e+00}$
UF6	$9.514643e-02_{5.0e-02}$	$2.001687e-01_{8.5e-02}$	$1.725421e-01_{0.0e+00}$
UF7	$4.701857e-01_{8.9e-03}$	$4.958239e-01_{8.2e-04}$	$4.954061e-01_{0.0e+00}$
UF8	$2.564091e-01_{3.3e-02}$	$3.271188e-01_{1.6e-02}$	$3.623309e-01_{0.0e+00}$
UF9	$5.107049e-01_{3.8e-02}$	$5.646465e-01_{3.1e-02}$	$5.565798e-01_{0.0e+00}$
UF10	$0.000000e+00_{0.0e+00}$	$7.045457e-02_{1.4e-02}$	$9.315613e-02_{0.0e+00}$
WFG1	$4.500378e-01_{6.6e-02}$	$6.347178e-01_{3.6e-04}$	$6.370891e-01_{0.0e+00}$
WFG2	$5.635474e-01_{7.7e-04}$	$5.646761e-01_{1.2e-05}$	$5.654082e-01_{0.0e+00}$
WFG3	$4.922166e-01_{2.6e-03}$	$4.979994e-01_{4.2e-06}$	$4.987931e-01_{0.0e+00}$
WFG4	$2.117675e-01_{1.3e-03}$	$2.211018e-01_{1.4e-04}$	$2.220795e-01_{0.0e+00}$
WFG5	$1.995414e-01_{2.1e-03}$	$1.987193e-01_{2.8e-03}$	$1.989817e-01_{0.0e+00}$
WFG6	$2.109626e-01_{1.5e-03}$	$2.128725e-01_{3.8e-06}$	$2.136358e-01_{0.0e+00}$
WFG7	$2.129709e-01_{1.8e-04}$	$2.128533e-01_{6.8e-06}$	$2.136157e-01_{0.0e+00}$
WFG8	$1.576003e-01_{1.8e-02}$	$1.732010e-01_{2.6e-02}$	$2.114787e-01_{0.0e+00}$
WFG9	$2.412725e-01_{1.6e-04}$	$2.438372e-01_{4.6e-05}$	$2.449073e-01_{0.0e+00}$
DTLZ1	$3.981344e-02_{1.6e-01}$	$7.848863e-01_{1.7e-04}$	$8.027772e-01_{0.0e+00}$
DTLZ2	$4.325969e-01_{1.5e-03}$	$4.185189e-01_{9.8e-04}$	$4.294453e-01_{0.0e+00}$
DTLZ3	$8.816880e-02_{1.7e-01}$	$4.182285e-01_{1.1e-03}$	$4.298513e-01_{0.0e+00}$
DTLZ4	$4.198804e-01_{1.5e-03}$	$4.065588e-01_{3.0e-02}$	$4.245304e-01_{0.0e+00}$
DTLZ5	$8.724304e-02_{2.3e-03}$	$9.469878e-02_{9.7e-06}$	$9.579286e-02_{0.0e+00}$
DTLZ6	$9.656330e-02_{1.2e-05}$	$9.566300e-02_{8.0e-07}$	$9.678412e-02_{0.0e+00}$
DTLZ7	$3.129314e-01_{2.5e-03}$	$2.800935e-01_{6.8e-03}$	$3.103702e-01_{0.0e+00}$

based on the decomposition mechanism (using the MOEAD/DE algorithm). In order to assess the effectiveness of using the co-evolutionary mechanism, the author compares the proposed algorithm with these two baseline algorithms (i.e., NSGA-II and MOEAD/DE). Tables 2.5 and 2.6 provide the performance comparisons of DPPCP, MOEA/D-DE, and NSGA-II on 32 test instances with respect to the IGD and HV metrics, respectively. Based on experimental results, we can see that DPPCP achieves a better outcome than both NSGA-II and MOEA-D/DE. It wins 26 out of 31 comparisons using the HV and the IGD metrics. It is worth noting that although NSGA-II is the worst among the three candidates, it achieves the best IGD metric values on the UF4 and the

Table 2.4: Performance comparisons between the proposed algorithms and state-of-the-art algorithm using the IGD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPP	DPP2	DPPCP
ZDT1	$3.945162e-04_{6.4e-04}$	$5.554228e-05_{2.2e-07}$	$3.093510e-05_{0.0e+00}$
ZDT2	$4.602254e-05_{3.6e-05}$	$4.661050e-05_{7.3e-08}$	$3.241247e-05_{0.0e+00}$
ZDT3	$6.907624e-05_{8.2e-05}$	$8.860561e-05_{8.5e-07}$	$2.623904e-05_{0.0e+00}$
ZDT4	$3.273414e-01_{2.5e-01}$	$5.866687e-05_{4.8e-07}$	$3.104248e-05_{0.0e+00}$
ZDT6	$3.147239e-05_{8.0e-07}$	$4.628070e-05_{6.9e-09}$	$3.135962e-05_{0.0e+00}$
UF1	$1.219677e-03_{4.7e-04}$	$6.864305e-05_{3.0e-06}$	$5.686509e-05_{0.0e+00}$
UF2	$1.983361e-03_{6.9e-04}$	$4.221505e-04_{1.3e-04}$	$1.700001e-04_{0.0e+00}$
UF3	$6.117055e-03_{1.7e-03}$	$2.379576e-04_{5.5e-04}$	$1.820430e-03_{0.0e+00}$
UF4	$2.055704e-03_{3.4e-04}$	$1.955396e-03_{2.4e-04}$	$1.676999e-03_{0.0e+00}$
UF5	$1.319009e-01_{4.5e-02}$	$6.085614e-02_{2.8e-02}$	$1.547979e-01_{0.0e+00}$
UF6	$1.023728e-02_{2.6e-03}$	$6.511947e-03_{4.9e-03}$	$2.268674e-02_{0.0e+00}$
UF7	$7.577647e-04_{4.1e-04}$	$1.073177e-04_{3.7e-05}$	$1.185823e-04_{0.0e+00}$
UF8	$1.105829e-03_{3.2e-04}$	$1.033388e-03_{2.1e-04}$	$8.438084e-04_{0.0e+00}$
UF9	$2.296300e-03_{2.4e-04}$	$2.186628e-03_{1.5e-04}$	$2.269156e-03_{0.0e+00}$
UF10	$1.254666e-02_{4.0e-03}$	$4.860551e-03_{5.9e-04}$	$4.938679e-03_{0.0e+00}$
WFG1	$4.091125e-03_{2.0e-03}$	$2.672370e-04_{1.8e-05}$	$7.811839e-05_{0.0e+00}$
WFG2	$3.898765e-04_{1.9e-04}$	$6.061874e-04_{2.3e-05}$	$8.474260e-05_{0.0e+00}$
WFG3	$1.837856e-04_{8.8e-05}$	$5.480337e-05_{3.4e-08}$	$3.343249e-05_{0.0e+00}$
WFG4	$2.109915e-04_{2.9e-05}$	$6.344580e-05_{1.9e-06}$	$3.390037e-05_{0.0e+00}$
WFG5	$9.304804e-04_{2.1e-06}$	$9.328236e-04_{8.8e-07}$	$9.303355e-04_{0.0e+00}$
WFG6	$1.249467e-04_{9.1e-05}$	$9.143875e-05_{2.0e-07}$	$5.394634e-05_{0.0e+00}$
WFG7	$2.832908e-05_{3.6e-06}$	$4.054432e-05_{2.5e-08}$	$2.255773e-05_{0.0e+00}$
WFG8	$3.479606e-03_{9.6e-04}$	$3.172940e-03_{2.8e-03}$	$8.095268e-04_{0.0e+00}$
WFG9	$5.823859e-05_{3.0e-06}$	$4.076749e-05_{3.5e-07}$	$2.269694e-05_{0.0e+00}$
DTLZ1	$2.274694e-02_{1.4e-02}$	$3.471784e-04_{1.4e-06}$	$2.526425e-04_{0.0e+00}$
DTLZ2	$3.282544e-04_{9.9e-06}$	$4.301280e-04_{1.9e-06}$	$3.343429e-04_{0.0e+00}$
DTLZ3	$1.647616e-01_{3.3e-01}$	$7.229934e-04_{7.5e-06}$	$5.305681e-04_{0.0e+00}$
DTLZ4	$4.906612e-04_{2.2e-05}$	$8.454214e-04_{2.2e-04}$	$5.417136e-04_{0.0e+00}$
DTLZ5	$2.934037e-05_{6.4e-06}$	$1.518117e-05_{1.8e-07}$	$3.674228e-06_{0.0e+00}$
DTLZ6	$1.138830e-05_{1.1e-06}$	$3.454072e-05_{2.2e-08}$	$8.785615e-06_{0.0e+00}$
DTLZ7	$1.091168e-03_{5.5e-05}$	$2.612434e-03_{2.2e-04}$	$1.181486e-03_{0.0e+00}$

UF5. Meanwhile, MOEA/D-DE obtains the best IGD metric values on the UF3, UF6, UF9, and WFG5. By contrast, DPPCP shows a poor result on the UF5 test instance. However, DPPCP shows better performance than the baseline algorithm on all the ZDT and DTLZ instances. These results indicate the effectiveness of DPPCP in achieving both convergence and diversity criteria.

2.7.3. Statistical test for comparing performance

The detailed comparison results of the proposed algorithm with other algorithms have been presented in the previous sections. Based on these results, it can be concluded that the proposed algorithm yields better

Table 2.5: Performance comparisons between the DPPCP and baseline algorithms using the HV metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	NSGAII	MOEAD	DPPCP
ZDT1	$6.647858e-01_{5.3e-05}$	$6.648386e-01_{4.7e-05}$	$6.655793e-01_{0.0e+00}$
ZDT2	$3.314961e-01_{3.3e-05}$	$3.315780e-01_{5.1e-05}$	$3.321892e-01_{0.0e+00}$
ZDT3	$5.168567e-01_{1.2e-05}$	$5.162200e-01_{1.1e-05}$	$5.170450e-01_{0.0e+00}$
ZDT4	$6.648548e-01_{2.2e-05}$	$6.649686e-01_{9.8e-06}$	$6.655913e-01_{0.0e+00}$
ZDT6	$4.031456e-01_{1.4e-04}$	$4.047282e-01_{1.7e-08}$	$4.053136e-01_{0.0e+00}$
UF1	$5.484432e-01_{1.9e-02}$	$6.636184e-01_{1.3e-04}$	$6.639659e-01_{0.0e+00}$
UF2	$6.396909e-01_{3.6e-03}$	$6.572483e-01_{2.8e-03}$	$6.616692e-01_{0.0e+00}$
UF3	$4.571061e-01_{3.4e-02}$	$6.569119e-01_{9.5e-03}$	$5.965543e-01_{0.0e+00}$
UF4	$2.726602e-01_{3.2e-04}$	$2.458777e-01_{5.8e-03}$	$2.551797e-01_{0.0e+00}$
UF5	$2.374327e-01_{2.8e-02}$	$3.632481e-02_{6.5e-02}$	$8.999985e-02_{0.0e+00}$
UF6	$2.778372e-01_{4.1e-02}$	$2.024907e-01_{7.9e-02}$	$1.725421e-01_{0.0e+00}$
UF7	$4.195309e-01_{6.1e-02}$	$4.952861e-01_{2.4e-03}$	$4.954061e-01_{0.0e+00}$
UF8	$1.069511e-01_{2.2e-02}$	$3.295556e-01_{2.3e-02}$	$3.623309e-01_{0.0e+00}$
UF9	$4.296275e-01_{1.2e-01}$	$6.014756e-01_{6.2e-02}$	$5.565798e-01_{0.0e+00}$
UF10	$5.292133e-04_{1.1e-03}$	$5.944409e-02_{2.7e-02}$	$9.315613e-02_{0.0e+00}$
WFG1	$6.336915e-01_{3.8e-04}$	$6.347114e-01_{2.2e-04}$	$6.370891e-01_{0.0e+00}$
WFG2	$5.652985e-01_{1.2e-05}$	$5.646742e-01_{1.3e-05}$	$5.654082e-01_{0.0e+00}$
WFG3	$4.978645e-01_{4.7e-05}$	$4.980009e-01_{6.1e-06}$	$4.987931e-01_{0.0e+00}$
WFG4	$2.214379e-01_{8.9e-05}$	$2.212084e-01_{1.1e-04}$	$2.220795e-01_{0.0e+00}$
WFG5	$1.982352e-01_{8.4e-05}$	$1.988567e-01_{2.2e-03}$	$1.989817e-01_{0.0e+00}$
WFG6	$2.104230e-01_{3.2e-03}$	$2.128723e-01_{4.5e-06}$	$2.136358e-01_{0.0e+00}$
WFG7	$2.129812e-01_{4.4e-05}$	$2.128570e-01_{8.1e-06}$	$2.136157e-01_{0.0e+00}$
WFG8	$1.676837e-01_{2.3e-02}$	$1.704603e-01_{2.4e-02}$	$2.114787e-01_{0.0e+00}$
WFG9	$2.433160e-01_{5.0e-04}$	$2.438831e-01_{5.5e-05}$	$2.449073e-01_{0.0e+00}$
DTLZ1	$7.952908e-01_{2.2e-03}$	$7.849171e-01_{2.9e-04}$	$8.027772e-01_{0.0e+00}$
DTLZ2	$4.145143e-01_{2.7e-03}$	$4.185581e-01_{8.5e-04}$	$4.294453e-01_{0.0e+00}$
DTLZ3	$4.237451e-01_{2.2e-03}$	$4.192297e-01_{9.9e-04}$	$4.298513e-01_{0.0e+00}$
DTLZ4	$4.135101e-01_{1.9e-03}$	$4.121623e-01_{2.2e-02}$	$4.245304e-01_{0.0e+00}$
DTLZ5	$9.540642e-02_{3.0e-05}$	$9.469685e-02_{7.7e-06}$	$9.579286e-02_{0.0e+00}$
DTLZ6	$6.407350e-02_{1.0e-02}$	$9.566239e-02_{7.8e-07}$	$9.678412e-02_{0.0e+00}$
DTLZ7	$3.120551e-01_{1.2e-03}$	$2.770128e-01_{1.6e-02}$	$3.103702e-01_{0.0e+00}$

average performance than the other algorithms. To further strengthen this claim, the authors conducted statistical evaluations to determine whether there is a significant difference between the algorithms. Specifically, the Friedman test, a non-parametric test, is used to check whether there are significant differences among the results. The null hypothesis (Ho) is that there is no difference between the algorithms. If the p-value is smaller than a significance level (i.e., 0.05), the null hypothesis is rejected (or there are significant differences between the algorithms), and vice versa.

Table 2.7 shows the Friedman statistic of the IGD metric considering reduction performance (distributed according to chi-square with 4 de-

Table 2.6: Performance comparisons between the DPPCP and baseline algorithms using the IGD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	NSGAII	MOEAD	DPPCP
ZDT1	$5.788071e-05_{4.8e-06}$	$5.556843e-05_{3.3e-07}$	$3.093510e-05_{0.0e+00}$
ZDT2	$5.968199e-05_{2.9e-06}$	$4.660528e-05_{4.7e-08}$	$3.241247e-05_{0.0e+00}$
ZDT3	$4.146423e-05_{1.9e-06}$	$8.838159e-05_{7.9e-07}$	$2.623904e-05_{0.0e+00}$
ZDT4	$5.699439e-05_{2.2e-06}$	$5.906846e-05_{5.0e-07}$	$3.104248e-05_{0.0e+00}$
ZDT6	$7.378607e-05_{4.2e-06}$	$4.628025e-05_{6.2e-09}$	$3.135962e-05_{0.0e+00}$
UF1	$3.546947e-03_{6.0e-04}$	$6.903693e-05_{5.0e-06}$	$5.686509e-05_{0.0e+00}$
UF2	$1.066904e-03_{2.9e-04}$	$3.608501e-04_{2.4e-04}$	$1.700001e-04_{0.0e+00}$
UF3	$7.154636e-03_{1.8e-03}$	$1.702418e-04_{1.5e-04}$	$1.820430e-03_{0.0e+00}$
UF4	$1.358224e-03_{2.4e-05}$	$1.940253e-03_{2.3e-04}$	$1.676999e-03_{0.0e+00}$
UF5	$4.394656e-02_{8.3e-03}$	$6.607502e-02_{1.7e-02}$	$1.547979e-01_{0.0e+00}$
UF6	$8.797878e-03_{3.8e-03}$	$3.479782e-03_{8.8e-03}$	$2.268674e-02_{0.0e+00}$
UF7	$1.859278e-03_{1.7e-03}$	$1.085036e-04_{2.4e-05}$	$1.185823e-04_{0.0e+00}$
UF8	$2.981191e-03_{1.7e-04}$	$9.841127e-04_{4.3e-04}$	$8.438084e-04_{0.0e+00}$
UF9	$2.732983e-03_{2.0e-03}$	$2.165702e-03_{1.5e-03}$	$2.269156e-03_{0.0e+00}$
UF10	$5.161469e-03_{3.7e-03}$	$4.986122e-03_{6.2e-04}$	$4.938679e-03_{0.0e+00}$
WFG1	$3.200666e-04_{2.3e-05}$	$2.702204e-04_{2.7e-05}$	$7.811839e-05_{0.0e+00}$
WFG2	$1.174109e-04_{1.1e-05}$	$6.057198e-04_{6.6e-06}$	$8.474260e-05_{0.0e+00}$
WFG3	$6.512669e-05_{3.3e-06}$	$5.476262e-05_{9.3e-08}$	$3.343249e-05_{0.0e+00}$
WFG4	$5.717426e-05_{2.8e-06}$	$6.238275e-05_{1.3e-06}$	$3.390037e-05_{0.0e+00}$
WFG5	$9.330129e-04_{5.7e-07}$	$9.337936e-04_{3.8e-07}$	$9.303355e-04_{0.0e+00}$
WFG6	$1.122812e-04_{6.5e-05}$	$9.139109e-05_{1.4e-07}$	$5.394634e-05_{0.0e+00}$
WFG7	$3.877603e-05_{1.7e-06}$	$4.054086e-05_{1.9e-08}$	$2.255773e-05_{0.0e+00}$
WFG8	$2.747423e-03_{2.2e-03}$	$3.174261e-03_{2.4e-03}$	$8.095268e-04_{0.0e+00}$
WFG9	$4.337451e-05_{5.0e-06}$	$4.071753e-05_{1.1e-07}$	$2.269694e-05_{0.0e+00}$
DTLZ1	$3.201110e-04_{1.9e-05}$	$3.474762e-04_{1.3e-06}$	$2.526425e-04_{0.0e+00}$
DTLZ2	$4.355620e-04_{2.4e-05}$	$4.306742e-04_{3.0e-06}$	$3.343429e-04_{0.0e+00}$
DTLZ3	$6.953162e-04_{2.1e-05}$	$7.211632e-04_{6.2e-06}$	$5.305681e-04_{0.0e+00}$
DTLZ4	$7.724318e-04_{1.2e-04}$	$7.911537e-04_{1.3e-04}$	$5.417136e-04_{0.0e+00}$
DTLZ5	$6.185971e-06_{4.8e-07}$	$1.516055e-05_{1.1e-07}$	$3.674228e-06_{0.0e+00}$
DTLZ6	$3.789467e-04_{2.9e-04}$	$3.453845e-05_{3.1e-08}$	$8.785615e-06_{0.0e+00}$
DTLZ7	$1.204211e-03_{6.4e-05}$	$2.612999e-03_{1.7e-04}$	$1.181486e-03_{0.0e+00}$

Table 2.7: Average ranking of the algorithms using the IGD metric

Algorithm	Ranking
NSGAII	3.58065
MOEAD	3.1290
DPP	3.7742
DPP2	2.8710
DPPCP	1.6452

grees of freedom: 34.787), and the p-value is approximately 1.084e-06. From these p-values, it can be concluded that there is a significant difference between the compared algorithms. Combining the comparison results in the previous sections, it can be concluded that the DPPCP algorithm gives the best results, followed by the DPP2 algorithm.

2.7.4. Effects of competitiveness

To verify the effect of the competitive co-evolutionary method, the author has developed a variant (denoted as *DPPCP-Variant1*). In *DPPCP-Variant1*, there is no interaction between the two populations, except for a connection in the selection stage (NBSM), where some solutions on the A_d side can be borrowed from the A_p. Two offspring, $Child_{Ap}$ and $Child_{Ad}$ will be used to update the population immediately instead of competing in the DPPCP. The result is a combination of the output from each population.

The performance comparisons are shown in Tables 2.8 and 2.9 via the mean and standard deviation values. For each row in the table, the best value is highlighted in bold.

In Table 2.8, this study conducts the comparison between DPPCP and *DPPCP-Variant1* using the HV metric. The DPPCP attains better metric values in all of the comparisons (except UF5, UF6, and UF9). Especially in Table 2.9, the DPPCP's results are about ten times as good as the DPP's with ZDT1, ZDT3, ZDT4, UF1, WFG3, WFG4, DTLZ5, DTLZ6, and about 100 times with WFG1, WFG2.

It can be seen that DPPCP shows better performance than *DPPCP-Variant1* in most instances. Especially, DPPCP outperforms *DPPCP-Variant1* in the WFG series. Based on the results, it is easy to see the advantage of the competitive co-evolutionary method. It helps to achieve better results on both criteria (i.e., convergence and diversity).

2.7.5. Effects of the NBSM mechanism

To further understand the effects of the NBSM mechanism, this study extends this mechanism to two other variants as follows:

1. DPPCP-Variant2: This variant is different from DPPCP in that it chooses two sub-regions in the A_p. If the sub-regions do not contain any

Table 2.8: Performance comparisons between the DPPCP and DPPCP-Variant1 using HV metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPPCP-Variant1	DPPCP
ZDT1	$6.630161e-01_{7.2e-04}$	$6.655793e-01_{0.0e+00}$
ZDT2	$3.308929e-01_{2.9e-04}$	$3.321892e-01_{0.0e+00}$
ZDT3	$5.146396e-01_{6.9e-04}$	$5.170450e-01_{0.0e+00}$
ZDT4	$6.636102e-01_{2.7e-04}$	$6.655913e-01_{0.0e+00}$
ZDT6	$4.045542e-01_{1.9e-05}$	$4.053136e-01_{0.0e+00}$
UF1	$6.507534e-01_{1.6e-02}$	$6.639659e-01_{0.0e+00}$
UF2	$6.571386e-01_{1.9e-03}$	$6.616692e-01_{0.0e+00}$
UF3	$5.920556e-01_{1.8e-02}$	$5.965543e-01_{0.0e+00}$
UF4	$2.470220e-01_{4.8e-03}$	$2.551797e-01_{0.0e+00}$
UF5	$1.429759e-01_{9.4e-02}$	$8.999985e-02_{0.0e+00}$
UF6	$2.563506e-01_{5.3e-02}$	$1.725421e-01_{0.0e+00}$
UF7	$4.927444e-01_{3.7e-03}$	$4.954061e-01_{0.0e+00}$
UF8	$3.124623e-01_{1.6e-02}$	$3.623309e-01_{0.0e+00}$
UF9	$6.119489e-01_{5.9e-02}$	$5.565798e-01_{0.0e+00}$
UF10	$5.908046e-02_{1.7e-02}$	$9.315613e-02_{0.0e+00}$
WFG1	$5.860027e-01_{7.6e-02}$	$6.370891e-01_{0.0e+00}$
WFG2	$5.636535e-01_{3.4e-04}$	$5.654082e-01_{0.0e+00}$
WFG3	$4.961655e-01_{4.1e-04}$	$4.987931e-01_{0.0e+00}$
WFG4	$2.194508e-01_{5.5e-04}$	$2.220795e-01_{0.0e+00}$
WFG5	$1.974580e-01_{2.2e-04}$	$1.989817e-01_{0.0e+00}$
WFG6	$2.113252e-01_{2.7e-04}$	$2.136358e-01_{0.0e+00}$
WFG7	$2.115485e-01_{2.1e-04}$	$2.136157e-01_{0.0e+00}$
WFG8	$1.716974e-01_{2.4e-02}$	$2.114787e-01_{0.0e+00}$
WFG9	$2.414078e-01_{6.9e-04}$	$2.449073e-01_{0.0e+00}$
DTLZ1	$7.868666e-01_{1.2e-03}$	$8.027772e-01_{0.0e+00}$
DTLZ2	$4.063181e-01_{1.5e-03}$	$4.294453e-01_{0.0e+00}$
DTLZ3	$4.069326e-01_{1.5e-03}$	$4.298513e-01_{0.0e+00}$
DTLZ4	$4.027093e-01_{2.5e-03}$	$4.245304e-01_{0.0e+00}$
DTLZ5	$9.467247e-02_{3.2e-05}$	$9.579286e-02_{0.0e+00}$
DTLZ6	$9.562214e-02_{8.9e-06}$	$9.678412e-02_{0.0e+00}$
DTLZ7	$2.873680e-01_{6.2e-03}$	$3.103702e-01_{0.0e+00}$

solutions, it randomly selects from the A_p instead of borrowing from the A_d such as DPPCP. This experiment aims to demonstrate the significance of selecting solutions in neighboring sub-regions.

2. DPPCP-Variant3: In A_p, instead of carefully selecting two mating parents from all neighborhoods of the current sub-region, this variant randomly selects two neighborhood sub-regions regardless of whether they contain any solutions or not. If two sub-regions do not contain any solution, they borrow from two sub-regions in the A_d respectively. This experiment aims to show the importance of searching for neighborhood solutions in the entire neighborhood sub-regions.

The performance comparisons between DPPCP and its two variants,

Table 2.9: Performance comparisons between the DPPCP with DPPCP-Variant1 using the IGD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPPCP-Variant1	DPPCP
ZDT1	$1.300515e-04_{2.0e-05}$	$3.093510e-05_{0.0e+00}$
ZDT2	$9.765445e-05_{2.2e-05}$	$3.241247e-05_{0.0e+00}$
ZDT3	$1.731261e-04_{2.4e-05}$	$2.623904e-05_{0.0e+00}$
ZDT4	$1.363398e-04_{1.4e-05}$	$3.104248e-05_{0.0e+00}$
ZDT6	$5.354539e-05_{5.6e-07}$	$3.135962e-05_{0.0e+00}$
UF1	$6.941814e-04_{8.8e-04}$	$5.686509e-05_{0.0e+00}$
UF2	$3.218592e-04_{1.0e-04}$	$1.700001e-04_{0.0e+00}$
UF3	$2.212806e-03_{6.5e-04}$	$1.820430e-03_{0.0e+00}$
UF4	$2.022856e-03_{1.5e-04}$	$1.676999e-03_{0.0e+00}$
UF5	$6.183181e-02_{1.6e-02}$	$1.547979e-01_{0.0e+00}$
UF6	$5.570435e-03_{4.3e-03}$	$2.268674e-02_{0.0e+00}$
UF7	$3.427245e-04_{4.4e-04}$	$1.185823e-04_{0.0e+00}$
UF8	$1.005022e-03_{1.5e-04}$	$8.438084e-04_{0.0e+00}$
UF9	$1.387835e-03_{6.8e-04}$	$2.269156e-03_{0.0e+00}$
UF10	$4.734256e-03_{4.8e-04}$	$4.938679e-03_{0.0e+00}$
WFG1	$1.711538e-03_{1.9e-03}$	$7.811839e-05_{0.0e+00}$
WFG2	$1.185953e-03_{8.2e-05}$	$8.474260e-05_{0.0e+00}$
WFG3	$1.448185e-04_{2.6e-05}$	$3.343249e-05_{0.0e+00}$
WFG4	$1.497142e-04_{2.8e-05}$	$3.390037e-05_{0.0e+00}$
WFG5	$9.367849e-04_{2.1e-06}$	$9.303355e-04_{0.0e+00}$
WFG6	$2.140030e-04_{3.1e-05}$	$5.394634e-05_{0.0e+00}$
WFG7	$8.367173e-05_{1.1e-05}$	$2.255773e-05_{0.0e+00}$
WFG8	$2.419494e-03_{1.0e-03}$	$8.095268e-04_{0.0e+00}$
WFG9	$9.602084e-05_{1.4e-05}$	$2.269694e-05_{0.0e+00}$
DTLZ1	$3.483237e-04_{6.9e-06}$	$2.526425e-04_{0.0e+00}$
DTLZ2	$4.540029e-04_{7.2e-06}$	$3.343429e-04_{0.0e+00}$
DTLZ3	$7.371303e-04_{1.6e-05}$	$5.305681e-04_{0.0e+00}$
DTLZ4	$6.460156e-04_{7.3e-05}$	$5.417136e-04_{0.0e+00}$
DTLZ5	$1.263027e-05_{1.2e-06}$	$3.674228e-06_{0.0e+00}$
DTLZ6	$3.071575e-05_{4.2e-07}$	$8.785615e-06_{0.0e+00}$
DTLZ7	$3.071591e-03_{3.0e-03}$	$1.181486e-03_{0.0e+00}$

regarding the IGD and the SPREAD metrics, are presented in Tables 2.10 and 2.11. It is clear that DPPCP is the best candidate: it obtains better metric values in 20 out of 31 comparisons. On the contrary, DPPCP-Variant2 is the worst among them with the IGD metric. Meanwhile, DPPCP-Variant3 has a low spread.

In short, our proposed NBSM mechanism, which fully utilizes the guidance information of the neighborhood, is effective.

2.7.6. Interaction between two co-evolving populations

In this section, this study clarifies the effect of using dual populations. Specifically, this study considers two main points:

Table 2.10: Performance comparisons between the DPPCP with DPPCP-Variant2 and DPPCP-Variant3 using IGD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPPCP	DPPCP-Variant3	DPPCP-Variant2
ZDT1	$\mathbf{6.655793e-01_{0.0e+00}}$	$6.655222e-01_{3.5e-05}$	$6.654682e-01_{8.0e-05}$
ZDT2	$3.321892e-01_{0.0e+00}$	$3.321594e-01_{4.2e-05}$	$\mathbf{3.322645e-01_{4.6e-05}}$
ZDT3	$5.170450e-01_{0.0e+00}$	$\mathbf{5.170514e-01_{4.4e-06}}$	$5.170115e-01_{2.3e-05}$
ZDT4	$6.655913e-01_{0.0e+00}$	$\mathbf{6.655955e-01_{1.2e-05}}$	$6.655771e-01_{1.7e-05}$
ZDT6	$4.053136e-01_{0.0e+00}$	$\mathbf{4.053195e-01_{1.0e-05}}$	$4.053177e-01_{2.0e-05}$
UF1	$\mathbf{6.639659e-01_{0.0e+00}}$	$6.638367e-01_{1.7e-04}$	$6.611297e-01_{1.3e-02}$
UF2	$\mathbf{6.616692e-01_{0.0e+00}}$	$6.584050e-01_{2.3e-03}$	$6.585740e-01_{1.8e-03}$
UF3	$5.965543e-01_{0.0e+00}$	$\mathbf{6.489444e-01_{1.2e-02}}$	$6.341751e-01_{4.1e-02}$
UF4	$\mathbf{2.551797e-01_{0.0e+00}}$	$2.522927e-01_{5.0e-03}$	$2.527116e-01_{3.6e-03}$
UF5	$8.999985e-02_{0.0e+00}$	$1.511017e-01_{9.1e-02}$	$\mathbf{1.851661e-01_{1.0e-01}}$
UF6	$1.725421e-01_{0.0e+00}$	$\mathbf{1.954255e-01_{1.0e-01}}$	$1.720137e-01_{8.3e-02}$
UF7	$\mathbf{4.954061e-01_{0.0e+00}}$	$4.472362e-01_{1.0e-01}$	$4.918281e-01_{8.7e-03}$
UF8	$\mathbf{3.623309e-01_{0.0e+00}}$	$3.418781e-01_{2.7e-02}$	$3.032140e-01_{8.0e-02}$
UF9	$5.565798e-01_{0.0e+00}$	$5.540199e-01_{7.7e-03}$	$\mathbf{5.885163e-01_{5.4e-02}}$
UF10	$\mathbf{9.315613e-02_{0.0e+00}}$	$6.069509e-02_{1.6e-02}$	$8.027631e-02_{4.1e-02}$
WFG1	$\mathbf{6.370891e-01_{0.0e+00}}$	$6.353138e-01_{3.2e-04}$	$6.353696e-01_{1.4e-04}$
WFG2	$5.654082e-01_{0.0e+00}$	$\mathbf{5.654134e-01_{6.5e-06}}$	$5.654117e-01_{4.9e-06}$
WFG3	$\mathbf{4.987931e-01_{0.0e+00}}$	$4.987877e-01_{8.3e-06}$	$4.987841e-01_{1.7e-05}$
WFG4	$2.220795e-01_{0.0e+00}$	$2.220849e-01_{1.1e-04}$	$\mathbf{2.221528e-01_{4.8e-05}}$
WFG5	$1.989817e-01_{0.0e+00}$	$\mathbf{1.997983e-01_{2.6e-03}}$	$1.993798e-01_{1.8e-03}$
WFG6	$2.136358e-01_{0.0e+00}$	$\mathbf{2.136419e-01_{1.0e-05}}$	$2.136229e-01_{9.6e-06}$
WFG7	$2.136157e-01_{0.0e+00}$	$\mathbf{2.136398e-01_{9.0e-06}}$	$2.136212e-01_{8.9e-06}$
WFG8	$\mathbf{2.114787e-01_{0.0e+00}}$	$1.950397e-01_{2.6e-02}$	$1.762630e-01_{2.7e-02}$
WFG9	$\mathbf{2.449073e-01_{0.0e+00}}$	$2.446959e-01_{8.0e-05}$	$2.447730e-01_{1.6e-04}$
DTLZ1	$8.027772e-01_{0.0e+00}$	$\mathbf{8.034714e-01_{6.8e-04}}$	$7.993489e-01_{8.5e-04}$
DTLZ2	$4.294453e-01_{0.0e+00}$	$\mathbf{4.318317e-01_{1.2e-03}}$	$4.275582e-01_{8.8e-04}$
DTLZ3	$4.298513e-01_{0.0e+00}$	$\mathbf{4.337437e-01_{1.2e-03}}$	$4.282988e-01_{8.1e-04}$
DTLZ4	$4.245304e-01_{0.0e+00}$	$\mathbf{4.270484e-01_{6.5e-04}}$	$4.241378e-01_{7.9e-04}$
DTLZ5	$\mathbf{9.579286e-02_{0.0e+00}}$	$9.578658e-02_{1.1e-05}$	$9.574149e-02_{1.1e-05}$
DTLZ6	$\mathbf{9.678412e-02_{0.0e+00}}$	$9.676609e-02_{6.0e-06}$	$9.676292e-02_{7.1e-06}$
DTLZ7	$\mathbf{3.103702e-01_{0.0e+00}}$	$2.811700e-01_{4.6e-02}$	$2.462745e-01_{3.2e-02}$

1. The effect of using competitive co-evolution on each population.
2. The effect of the interaction between two populations.

Specifically, this study first compares A_p with the NSGA-II algorithm and A_d with the MOEA/D-DE algorithm. As discussed above, the algorithms used in A_p and A_d differ from baseline algorithms (i.e., NSGA-II and MOEA/D) at three main points: (a) the mating parent selection mechanism (i.e. NBSM); (b) the way to generate offspring (i.e., competitive method); and (c) how to update Offspring to populations. Through this experiment, we will know whether co-evolution helps solution populations evolve better than independent evolution.

To implement this comparison, this study creates two new variants of

Table 2.11: Performance comparisons between the DPPCP with DPPCP-Variant2 and DPPCP-Variant3 using the SPREAD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPPCP	DPPCP-Variant3	DPPCP-Variant2
ZDT1	$5.090732e-01_{0.0e+00}$	$5.147352e-01_{1.8e-02}$	$5.096334e-01_{1.4e-02}$
ZDT2	$5.234142e-01_{0.0e+00}$	$4.927756e-01_{1.3e-02}$	$4.992091e-01_{2.2e-02}$
ZDT3	$8.496801e-01_{0.0e+00}$	$8.649409e-01_{3.5e-03}$	$8.585498e-01_{5.6e-03}$
ZDT4	$4.978226e-01_{0.0e+00}$	$5.190670e-01_{1.3e-02}$	$5.083129e-01_{1.4e-02}$
ZDT6	$1.262919e+00_{0.0e+00}$	$1.100262e+00_{3.3e-01}$	$7.020149e-01_{3.2e-01}$
UF1	$4.395728e-01_{0.0e+00}$	$4.341900e-01_{1.9e-02}$	$4.813619e-01_{9.4e-02}$
UF2	$5.609735e-01_{0.0e+00}$	$5.422028e-01_{2.4e-02}$	$5.267486e-01_{2.3e-02}$
UF3	$9.158526e-01_{0.0e+00}$	$8.435704e-01_{2.0e-01}$	$8.540161e-01_{2.0e-01}$
UF4	$5.607462e-01_{0.0e+00}$	$6.460472e-01_{6.7e-02}$	$6.239098e-01_{6.3e-02}$
UF5	$1.116620e+00_{0.0e+00}$	$1.210485e+00_{2.2e-01}$	$1.219553e+00_{1.7e-01}$
UF6	$1.000035e+00_{0.0e+00}$	$1.329176e+00_{1.8e-01}$	$1.217140e+00_{1.6e-01}$
UF7	$9.278187e-01_{0.0e+00}$	$6.441274e-01_{3.1e-01}$	$5.959498e-01_{1.7e-01}$
UF8	$8.112851e-01_{0.0e+00}$	$8.631439e-01_{4.4e-02}$	$8.516223e-01_{1.1e-01}$
UF9	$1.172136e+00_{0.0e+00}$	$1.055870e+00_{1.0e-01}$	$1.016694e+00_{1.4e-01}$
UF10	$1.113429e+00_{0.0e+00}$	$1.139191e+00_{2.0e-01}$	$9.984497e-01_{1.4e-01}$
WFG1	$5.910268e-01_{0.0e+00}$	$5.980603e-01_{8.6e-03}$	$6.014965e-01_{1.4e-02}$
WFG2	$9.265058e-01_{0.0e+00}$	$9.275593e-01_{3.8e-03}$	$9.266749e-01_{3.2e-03}$
WFG3	$5.067168e-01_{0.0e+00}$	$5.092385e-01_{9.3e-03}$	$5.061781e-01_{1.5e-02}$
WFG4	$5.102968e-01_{0.0e+00}$	$5.259370e-01_{8.2e-03}$	$5.184175e-01_{1.5e-02}$
WFG5	$5.124313e-01_{0.0e+00}$	$5.297967e-01_{1.2e-02}$	$5.311470e-01_{1.1e-02}$
WFG6	$5.025690e-01_{0.0e+00}$	$5.081665e-01_{1.7e-02}$	$5.058109e-01_{1.5e-02}$
WFG7	$5.117605e-01_{0.0e+00}$	$5.070984e-01_{1.2e-02}$	$5.085303e-01_{1.4e-02}$
WFG8	$5.850444e-01_{0.0e+00}$	$6.678814e-01_{9.4e-02}$	$6.883508e-01_{7.6e-02}$
WFG9	$5.152843e-01_{0.0e+00}$	$5.435935e-01_{1.6e-02}$	$5.384225e-01_{1.1e-02}$
DTLZ1	$7.480045e-01_{0.0e+00}$	$7.700924e-01_{1.7e-02}$	$7.612776e-01_{1.5e-02}$
DTLZ2	$6.909910e-01_{0.0e+00}$	$7.058144e-01_{2.1e-02}$	$6.936715e-01_{1.4e-02}$
DTLZ3	$7.258689e-01_{0.0e+00}$	$7.074477e-01_{1.5e-02}$	$6.992068e-01_{1.7e-02}$
DTLZ4	$7.042373e-01_{0.0e+00}$	$7.078723e-01_{1.9e-02}$	$6.894604e-01_{1.2e-02}$
DTLZ5	$6.133811e-01_{0.0e+00}$	$6.366189e-01_{1.3e-02}$	$6.351343e-01_{1.7e-02}$
DTLZ6	$6.074105e-01_{0.0e+00}$	$6.360939e-01_{1.2e-02}$	$6.328897e-01_{9.2e-03}$
DTLZ7	$9.165426e-01_{0.0e+00}$	$8.179126e-01_{1.1e-01}$	$7.521809e-01_{1.1e-01}$

the DPPCP algorithm: *DPPCP-Ad* and *DPPCP-Ap*. These variants are very similar to the DPPCP algorithm, except that at the last step they only get the results done by the A_p (for *DPPCP-Ap*) and by the A_d (for *DPPCP-Ad*). The performances of NSGA-II and *DPPCP-Ap* are presented in Tables 2.12, 2.13; 2.14, 2.15 show the results of comparisons between MOEA/D-DE and DPPCP-Ad. It is clear that *DPPCP-Ap* and *DPPCP-Ad* give better results than NSGA-II and MOEA/D-DE respectively. *DPPCP-Ap* wins in 25 out of 31 comparisons, *DPPCP-Ad* obtains better results in 22 out of 31 comparisons using the IGD metric. Through experimental results, it can be seen that the effectiveness of baseline algorithms is enhanced by utilizing a competitive

Table 2.12: Performance comparisons between NSGAII with Ap using HV metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	NSGAII	Ap
ZDT1	$6.647712e-01_{5.6e-05}$	$6.650548e-01_{3.2e-05}$
ZDT2	$3.314915e-01_{2.8e-05}$	$3.317714e-01_{1.2e-05}$
ZDT3	$5.168546e-01_{1.3e-05}$	$5.169530e-01_{2.6e-06}$
ZDT4	$6.648566e-01_{2.0e-05}$	$6.651099e-01_{3.9e-06}$
ZDT6	$4.031377e-01_{1.2e-04}$	$4.047653e-01_{6.9e-06}$
UF1	$5.419462e-01_{2.1e-02}$	$6.639871e-01_{8.5e-05}$
UF2	$6.409438e-01_{3.6e-03}$	$6.602307e-01_{9.2e-04}$
UF3	$4.692771e-01_{1.8e-02}$	$6.531597e-01_{1.2e-02}$
UF4	$2.726242e-01_{3.8e-04}$	$2.538280e-01_{4.0e-03}$
UF5	$2.395064e-01_{3.2e-02}$	$1.533705e-01_{9.1e-02}$
UF6	$2.702487e-01_{4.2e-02}$	$1.845366e-01_{7.1e-02}$
UF7	$4.238320e-01_{6.3e-02}$	$4.944873e-01_{3.7e-03}$
UF8	$1.065488e-01_{2.3e-02}$	$3.201665e-01_{2.2e-02}$
UF9	$4.384664e-01_{1.2e-01}$	$4.968106e-01_{6.8e-02}$
UF10	$8.641870e-04_{1.4e-03}$	$3.287506e-02_{2.3e-02}$
WFG1	$6.337007e-01_{3.9e-04}$	$6.354226e-01_{1.9e-04}$
WFG2	$5.653015e-01_{1.1e-05}$	$5.653629e-01_{2.6e-06}$
WFG3	$4.978586e-01_{4.8e-05}$	$4.982742e-01_{5.1e-06}$
WFG4	$2.214275e-01_{1.0e-04}$	$2.218771e-01_{3.7e-06}$
WFG5	$1.982329e-01_{9.5e-05}$	$1.986686e-01_{7.8e-06}$
WFG6	$2.097628e-01_{4.0e-03}$	$2.133093e-01_{2.2e-06}$
WFG7	$2.129740e-01_{5.5e-05}$	$2.133029e-01_{1.6e-06}$
WFG8	$1.752138e-01_{2.6e-02}$	$1.734960e-01_{2.6e-02}$
WFG9	$2.435342e-01_{4.9e-04}$	$2.444773e-01_{1.5e-04}$
DTLZ1	$7.951600e-01_{2.9e-03}$	$7.902245e-01_{1.8e-03}$
DTLZ2	$4.146214e-01_{2.9e-03}$	$4.210202e-01_{1.9e-03}$
DTLZ3	$4.231073e-01_{2.8e-03}$	$4.225103e-01_{2.4e-03}$
DTLZ4	$4.132746e-01_{1.8e-03}$	$4.148104e-01_{2.7e-03}$
DTLZ5	$9.541452e-02_{2.5e-05}$	$9.562316e-02_{6.0e-06}$
DTLZ6	$6.790182e-02_{1.2e-02}$	$9.656647e-02_{1.0e-05}$
DTLZ7	$3.121336e-01_{1.4e-03}$	$2.459266e-01_{3.4e-02}$

co-evolutionary method.

The author continues comparing the results of each independent population (i.e., A_p and A_d) using co-evolutionary mechanisms with the result of combining both dual populations. Through this comparison, this study examines whether or not the use of dual populations combines the quintessence of both populations.

Tables 2.16 and 2.17 show the results of comparisons between DPPCP, *DPPCP-Ap*, and *DPPCP-Ad*. It is clear that DPPCP achieves better values in most instances. This shows that thanks to the co-evolution mechanism, with interactions between solutions in two populations, the final population can get the advantages of both populations. It can be

Table 2.13: Performance comparisons between NSGAII with Ap using the IGD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	NSGAII	Ap
ZDT1	$5.919099e-05_{2.6e-06}$	$4.484072e-05_{2.0e-07}$
ZDT2	$6.035183e-05_{2.2e-06}$	$4.586843e-05_{2.9e-07}$
ZDT3	$4.190027e-05_{1.5e-06}$	$3.235044e-05_{4.4e-07}$
ZDT4	$5.677679e-05_{1.1e-06}$	$4.468699e-05_{2.6e-07}$
ZDT6	$7.329672e-05_{2.5e-06}$	$4.395604e-05_{1.6e-07}$
UF1	$3.531346e-03_{6.7e-04}$	$5.458855e-05_{1.5e-06}$
UF2	$1.014096e-03_{2.9e-04}$	$2.503008e-04_{5.8e-05}$
UF3	$7.093487e-03_{1.6e-03}$	$3.015407e-04_{2.9e-04}$
UF4	$1.351779e-03_{1.3e-05}$	$1.721485e-03_{8.0e-05}$
UF5	$4.420051e-02_{4.3e-03}$	$1.027762e-01_{4.5e-02}$
UF6	$8.820306e-03_{2.6e-03}$	$1.716127e-02_{5.2e-03}$
UF7	$3.519909e-03_{3.8e-03}$	$4.313322e-04_{5.1e-04}$
UF8	$2.991707e-03_{8.8e-05}$	$1.032012e-03_{1.4e-04}$
UF9	$2.324010e-03_{1.1e-03}$	$2.173429e-03_{5.4e-04}$
UF10	$5.264139e-03_{2.0e-03}$	$5.715941e-03_{9.0e-04}$
WFG1	$3.146505e-04_{2.1e-05}$	$2.353121e-04_{1.7e-05}$
WFG2	$1.185735e-04_{8.1e-06}$	$9.394695e-05_{5.0e-06}$
WFG3	$6.509974e-05_{2.8e-06}$	$4.879745e-05_{8.5e-07}$
WFG4	$5.599029e-05_{3.1e-06}$	$4.188075e-05_{5.4e-07}$
WFG5	$9.334401e-04_{1.2e-06}$	$9.309544e-04_{1.1e-07}$
WFG6	$1.654294e-04_{1.3e-04}$	$7.376991e-05_{3.9e-06}$
WFG7	$3.933206e-05_{3.0e-06}$	$2.845054e-05_{2.6e-07}$
WFG8	$2.389023e-03_{1.3e-03}$	$2.778718e-03_{1.4e-03}$
WFG9	$4.201996e-05_{2.8e-06}$	$2.995207e-05_{1.9e-07}$
DTLZ1	$3.231009e-04_{9.7e-06}$	$3.214582e-04_{1.3e-05}$
DTLZ2	$4.433946e-04_{1.7e-05}$	$4.132936e-04_{9.8e-06}$
DTLZ3	$6.935231e-04_{1.5e-05}$	$6.602071e-04_{1.7e-05}$
DTLZ4	$7.956023e-04_{1.2e-04}$	$7.878465e-04_{4.7e-05}$
DTLZ5	$6.212032e-06_{2.9e-07}$	$4.629317e-06_{7.3e-08}$
DTLZ6	$2.898050e-04_{1.5e-04}$	$1.134035e-05_{3.0e-07}$
DTLZ7	$1.210533e-03_{4.6e-05}$	$1.846060e-02_{7.2e-03}$

said that this population is likely to be able to balance both convergence and diversity.

The final solutions obtained by the DPPCP algorithm and the true PF on the DTLZ, UF, WFG, and ZDT series are plotted in Figure.2.21-2.24. From these figures, the author finds that the proposed algorithm can find the approximation set that covers entirely the true PF.

2.7.7. The change of population quality over time

To observe the changes in population quality over time, the author used line charts to show the changes of two values, IGD and HV, after each iteration of the proposed algorithm. For each dataset, there are two

Table 2.14: Performance comparisons between MOEAD/DE with Ad using the HV metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	MOEAD/DE	Ad
ZDT1	$6.648280e-01_{5.4e-05}$	$6.649500e-01_{3.2e-05}$
ZDT2	$3.315831e-01_{5.4e-05}$	$3.316895e-01_{1.0e-05}$
ZDT3	$5.162168e-01_{1.4e-05}$	$5.162376e-01_{3.0e-06}$
ZDT4	$6.649704e-01_{8.3e-06}$	$6.649978e-01_{4.4e-06}$
ZDT6	$4.047282e-01_{1.5e-08}$	$4.047278e-01_{1.3e-06}$
UF1	$6.635979e-01_{1.3e-04}$	$6.639067e-01_{1.6e-04}$
UF2	$6.565669e-01_{2.8e-03}$	$6.602800e-01_{6.3e-04}$
UF3	$6.578853e-01_{4.4e-03}$	$6.528604e-01_{1.0e-02}$
UF4	$2.472456e-01_{5.4e-03}$	$2.530931e-01_{5.1e-03}$
UF5	$4.976365e-02_{5.8e-02}$	$1.901213e-01_{1.2e-01}$
UF6	$2.059300e-01_{8.3e-02}$	$1.557399e-01_{9.6e-02}$
UF7	$4.945292e-01_{3.3e-03}$	$4.947838e-01_{3.1e-03}$
UF8	$3.286710e-01_{2.2e-02}$	$3.347479e-01_{3.0e-02}$
UF9	$5.973945e-01_{6.0e-02}$	$5.653599e-01_{7.9e-03}$
UF10	$7.037545e-02_{1.9e-02}$	$7.481132e-02_{3.5e-02}$
WFG1	$6.347123e-01_{2.2e-04}$	$6.349902e-01_{2.2e-04}$
WFG2	$5.646696e-01_{1.5e-05}$	$5.646973e-01_{2.0e-06}$
WFG3	$4.980009e-01_{4.5e-06}$	$4.980084e-01_{2.8e-06}$
WFG4	$2.212079e-01_{8.0e-05}$	$2.213991e-01_{9.9e-06}$
WFG5	$1.987543e-01_{2.0e-03}$	$1.989529e-01_{2.6e-03}$
WFG6	$2.128719e-01_{4.6e-06}$	$2.128776e-01_{3.4e-06}$
WFG7	$2.128584e-01_{8.0e-06}$	$2.128669e-01_{3.0e-06}$
WFG8	$1.732240e-01_{2.6e-02}$	$1.677706e-01_{2.3e-02}$
WFG9	$2.439037e-01_{6.4e-05}$	$2.439922e-01_{1.7e-04}$
DTLZ1	$7.848600e-01_{2.6e-04}$	$7.849389e-01_{2.5e-04}$
DTLZ2	$4.187079e-01_{9.6e-04}$	$4.186242e-01_{7.6e-04}$
DTLZ3	$4.193613e-01_{9.2e-04}$	$4.193516e-01_{1.2e-03}$
DTLZ4	$4.169853e-01_{8.7e-04}$	$4.157544e-01_{1.1e-03}$
DTLZ5	$9.469885e-02_{7.5e-06}$	$9.472012e-02_{5.0e-06}$
DTLZ6	$9.566261e-02_{6.8e-07}$	$9.566209e-02_{6.4e-07}$
DTLZ7	$2.801997e-01_{6.6e-03}$	$2.316691e-01_{2.6e-02}$

corresponding charts, one for IGD and one for HV of the corresponding sub-problems. Note that the smaller the IGD value, the better, while the opposite is true for HV. Therefore, when looking at the chart, if the IGD line goes down and the HV line goes up, it means that the quality of the population is improving over time.

The results are shown in Figures 2.10 to 2.17. One common point that can be observed in the four datasets is that both IGD and HV reach convergence in most cases. However, each problem achieves that state at different speeds.

For the ZDT dataset (Figures 2.10, 2.11), all five problems achieve the best IGD state early (under 200 iterations). However, for IGD, besides

Table 2.15: Performance comparisons between MOEAD/DE with Ad using the IGD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	MOEAD	Ad
ZDT1	$5.561091e-05_{2.2e-07}$	$\mathbf{5.528320e-05_{1.7e-07}}$
ZDT2	$\mathbf{4.661063e-05_{4.8e-08}}$	$4.662861e-05_{8.1e-09}$
ZDT3	$8.897932e-05_{1.7e-06}$	$\mathbf{8.797083e-05_{9.4e-08}}$
ZDT4	$5.903217e-05_{4.4e-07}$	$\mathbf{5.892080e-05_{1.5e-07}}$
ZDT6	$4.627950e-05_{5.4e-09}$	$\mathbf{4.627584e-05_{1.6e-09}}$
UF1	$7.056534e-05_{5.6e-06}$	$\mathbf{6.249016e-05_{2.8e-06}}$
UF2	$4.789866e-04_{1.5e-04}$	$\mathbf{2.519000e-04_{3.9e-05}}$
UF3	$\mathbf{1.852946e-04_{9.2e-05}}$	$2.948503e-04_{2.2e-04}$
UF4	$1.919477e-03_{1.2e-04}$	$\mathbf{1.765559e-03_{1.1e-04}}$
UF5	$\mathbf{6.953943e-02_{1.0e-02}}$	$9.583202e-02_{5.3e-02}$
UF6	$\mathbf{7.166399e-03_{8.6e-03}}$	$2.003117e-02_{7.4e-03}$
UF7	$\mathbf{3.171871e-04_{4.7e-04}}$	$3.210059e-04_{4.0e-04}$
UF8	$\mathbf{1.082474e-03_{2.5e-04}}$	$1.122969e-03_{3.3e-04}$
UF9	$\mathbf{1.706092e-03_{7.8e-04}}$	$2.168509e-03_{7.8e-05}$
UF10	$\mathbf{5.162630e-03_{4.9e-04}}$	$5.195688e-03_{6.8e-04}$
WFG1	$2.771777e-04_{1.4e-05}$	$\mathbf{2.574595e-04_{1.9e-05}}$
WFG2	$\mathbf{6.031492e-04_{8.5e-06}}$	$6.081285e-04_{1.1e-06}$
WFG3	$5.477941e-05_{6.2e-08}$	$\mathbf{5.475384e-05_{3.2e-08}}$
WFG4	$6.262985e-05_{7.2e-07}$	$\mathbf{6.154841e-05_{2.9e-07}}$
WFG5	$9.156851e-04_{5.7e-05}$	$\mathbf{9.095219e-04_{7.3e-05}}$
WFG6	$\mathbf{9.134833e-05_{1.0e-07}}$	$9.137324e-05_{6.0e-08}$
WFG7	$4.053872e-05_{1.2e-08}$	$\mathbf{4.053413e-05_{2.3e-08}}$
WFG8	$\mathbf{2.601146e-03_{1.3e-03}}$	$2.976396e-03_{1.2e-03}$
WFG9	$4.064367e-05_{1.4e-07}$	$\mathbf{4.055560e-05_{2.0e-07}}$
DTLZ1	$3.475079e-04_{1.1e-06}$	$\mathbf{3.473201e-04_{8.9e-07}}$
DTLZ2	$4.309118e-04_{2.2e-06}$	$\mathbf{4.308753e-04_{1.9e-06}}$
DTLZ3	$7.219248e-04_{4.6e-06}$	$\mathbf{7.210165e-04_{3.3e-06}}$
DTLZ4	$\mathbf{7.737441e-04_{5.9e-05}}$	$8.445142e-04_{7.7e-05}$
DTLZ5	$1.517558e-05_{6.4e-08}$	$\mathbf{1.514454e-05_{5.4e-08}}$
DTLZ6	$\mathbf{3.453149e-05_{2.9e-08}}$	$3.454897e-05_{1.9e-08}$
DTLZ7	$\mathbf{4.015304e-03_{4.3e-03}}$	$2.044364e-02_{7.1e-03}$

ZDT4 which takes 1000 iterations to reach the best state, other problems can achieve a balanced state early (around 20 iterations).

For the DTLZ dataset (Figure 2.12, 2.13), while for HV, DTLZ1 needs around 250 iterations to reach a balanced and best state compared to other problems, the remaining problems can achieve a balanced state before 200 iterations. Similarly for IGD, DTLZ1 also needs nearly 200 iterations, while other problems can achieve a balanced state in under 50 iterations.

For the WFG dataset (Figures 2.14, 2.15), the results with HV are quite similar to those with DTLZ, where DTLZ1 requires the most iterations (around 400 iterations) to reach the best state, while other

Table 2.16: Performance comparisons between DPPCP with *DPPCP-Ap* and *DPPCP-Ad* using the HV metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPPCP-Ap	DPPCP-Ad	DPPCP
ZDT1	$6.650573e-01_{3.4e-05}$	$6.649439e-01_{3.5e-05}$	$6.654682e-01_{8.0e-05}$
ZDT2	$3.317733e-01_{1.2e-05}$	$3.316889e-01_{1.1e-05}$	$3.322645e-01_{4.6e-05}$
ZDT3	$5.169542e-01_{2.8e-06}$	$5.162387e-01_{3.6e-06}$	$5.170115e-01_{2.3e-05}$
ZDT4	$6.651104e-01_{5.3e-06}$	$6.649971e-01_{5.5e-06}$	$6.655771e-01_{1.7e-05}$
ZDT6	$4.047643e-01_{6.5e-06}$	$4.047280e-01_{8.9e-07}$	$4.053177e-01_{2.0e-05}$
UF1	$6.639747e-01_{7.9e-05}$	$6.639469e-01_{1.4e-04}$	$6.611297e-01_{1.3e-02}$
UF2	$6.599954e-01_{1.3e-03}$	$6.601949e-01_{9.0e-04}$	$6.585740e-01_{1.8e-03}$
UF3	$6.521510e-01_{1.2e-02}$	$6.512283e-01_{1.2e-02}$	$6.341751e-01_{4.1e-02}$
UF4	$2.541054e-01_{3.9e-03}$	$2.555352e-01_{5.5e-03}$	$2.527116e-01_{3.6e-03}$
UF5	$1.690427e-01_{1.0e-01}$	$1.736297e-01_{1.2e-01}$	$1.851661e-01_{1.0e-01}$
UF6	$1.857803e-01_{8.4e-02}$	$1.660392e-01_{1.0e-01}$	$1.720137e-01_{8.3e-02}$
UF7	$4.834219e-01_{5.2e-02}$	$4.951684e-01_{2.6e-03}$	$4.918281e-01_{8.7e-03}$
UF8	$3.158450e-01_{2.6e-02}$	$3.255501e-01_{4.0e-02}$	$3.032140e-01_{8.0e-02}$
UF9	$4.965614e-01_{6.8e-02}$	$5.620449e-01_{9.7e-03}$	$5.885163e-01_{5.4e-02}$
UF10	$3.846684e-02_{2.6e-02}$	$8.202228e-02_{3.5e-02}$	$8.027631e-02_{4.1e-02}$
WFG1	$6.353914e-01_{1.7e-04}$	$6.350033e-01_{2.3e-04}$	$6.353696e-01_{1.4e-04}$
WFG2	$5.653623e-01_{2.9e-06}$	$5.646976e-01_{2.3e-06}$	$5.654117e-01_{4.9e-06}$
WFG3	$4.982741e-01_{4.9e-06}$	$4.980092e-01_{3.6e-06}$	$4.987841e-01_{1.7e-05}$
WFG4	$2.218733e-01_{7.9e-06}$	$2.214039e-01_{1.0e-05}$	$2.221528e-01_{4.8e-05}$
WFG5	$1.990736e-01_{1.8e-03}$	$1.993133e-01_{2.9e-03}$	$1.993798e-01_{1.8e-03}$
WFG6	$2.133075e-01_{3.9e-06}$	$2.128778e-01_{3.0e-06}$	$2.136229e-01_{9.6e-06}$
WFG7	$2.133019e-01_{3.1e-06}$	$2.128669e-01_{3.4e-06}$	$2.136212e-01_{8.9e-06}$
WFG8	$1.680864e-01_{2.2e-02}$	$1.731327e-01_{2.6e-02}$	$1.762630e-01_{2.7e-02}$
WFG9	$2.444569e-01_{1.5e-04}$	$2.439633e-01_{1.5e-04}$	$2.447730e-01_{1.6e-04}$
DTLZ1	$7.910644e-01_{2.2e-03}$	$7.848877e-01_{2.4e-04}$	$7.993489e-01_{8.5e-04}$
DTLZ2	$4.200114e-01_{2.3e-03}$	$4.184426e-01_{8.9e-04}$	$4.275582e-01_{8.8e-04}$
DTLZ3	$4.221261e-01_{2.2e-03}$	$4.190582e-01_{1.0e-03}$	$4.282988e-01_{8.1e-04}$
DTLZ4	$4.145095e-01_{2.2e-03}$	$4.159602e-01_{1.1e-03}$	$4.241378e-01_{7.9e-04}$
DTLZ5	$9.562550e-02_{6.0e-06}$	$9.472234e-02_{6.1e-06}$	$9.574149e-02_{1.1e-05}$
DTLZ6	$9.656628e-02_{1.0e-05}$	$9.566211e-02_{5.6e-07}$	$9.676292e-02_{7.1e-06}$
DTLZ7	$2.407067e-01_{3.4e-02}$	$2.339938e-01_{2.6e-02}$	$2.462745e-01_{3.2e-02}$

problems achieve the best state before 100 iterations. For IGD, WFG1 needs 1000 iterations to reach a state close to other problems, and with this trend, if the number of iterations is increased, WFG1 can achieve even better IGD.

For the UF dataset (Figures 2.16, 2.17), the results with HV are more diverse than for other problems. While UF2, UF4, and UF6 reach a balanced state, other problems still have a tendency to improve. For IGD, except for UF5, which continues to improve at 1000 iterations, other problems achieve a balanced state before 50 iterations.

From these charts, it can be seen that with the current number of iterations, most problems have achieved their best state, with only a few still having the potential to improve further.

Table 2.17: Performance comparisons between DPPCP with DPPCP-Ap and DPPCP-Ad using the IGD metric. The metric value with the highest mean is emphasized by being displayed in bold font with a gray background.

	DPPCP-Ap	DPPCP-Ad	DPPCP
ZDT1	$4.485633e-05_{2.2e-07}$	$5.530559e-05_{1.6e-07}$	$3.187123e-05_{6.9e-07}$
ZDT2	$4.583104e-05_{2.6e-07}$	$4.662961e-05_{7.6e-09}$	$3.175499e-05_{5.7e-07}$
ZDT3	$3.215338e-05_{5.4e-07}$	$8.807805e-05_{2.4e-07}$	$2.664059e-05_{4.9e-07}$
ZDT4	$4.468264e-05_{2.6e-07}$	$5.901455e-05_{3.2e-07}$	$3.138571e-05_{4.9e-07}$
ZDT6	$4.397428e-05_{1.8e-07}$	$4.627621e-05_{1.6e-09}$	$3.142337e-05_{7.2e-07}$
UF1	$5.465684e-05_{1.3e-06}$	$6.231947e-05_{2.6e-06}$	$3.484854e-04_{1.3e-03}$
UF2	$2.682161e-04_{7.3e-05}$	$2.575379e-04_{5.1e-05}$	$3.626042e-04_{1.1e-04}$
UF3	$3.703678e-04_{4.5e-04}$	$3.300163e-04_{2.7e-04}$	$7.198866e-04_{1.1e-03}$
UF4	$1.728170e-03_{8.3e-05}$	$1.719490e-03_{1.2e-04}$	$1.792191e-03_{9.6e-05}$
UF5	$9.393395e-02_{4.9e-02}$	$9.658430e-02_{5.1e-02}$	$9.217221e-02_{4.4e-02}$
UF6	$1.702495e-02_{4.9e-03}$	$1.887109e-02_{7.1e-03}$	$1.715472e-02_{5.8e-03}$
UF7	$1.034829e-03_{3.2e-03}$	$3.049188e-04_{3.6e-04}$	$4.723165e-04_{7.4e-04}$
UF8	$1.068130e-03_{1.6e-04}$	$1.199492e-03_{4.7e-04}$	$1.427242e-03_{1.6e-03}$
UF9	$2.177674e-03_{5.4e-04}$	$2.194444e-03_{1.0e-04}$	$1.860454e-03_{6.2e-04}$
UF10	$5.692229e-03_{7.7e-04}$	$4.920016e-03_{7.8e-04}$	$4.756647e-03_{8.1e-04}$
WFG1	$2.378909e-04_{1.5e-05}$	$2.564765e-04_{2.0e-05}$	$2.642254e-04_{9.2e-06}$
WFG2	$9.553416e-05_{4.9e-06}$	$6.077109e-04_{1.0e-06}$	$8.645268e-05_{5.0e-06}$
WFG3	$4.854589e-05_{8.6e-07}$	$5.473958e-05_{5.1e-08}$	$3.496741e-05_{5.4e-07}$
WFG4	$4.186748e-05_{5.1e-07}$	$6.152974e-05_{2.5e-07}$	$3.327079e-05_{6.0e-07}$
WFG5	$9.194347e-04_{5.2e-05}$	$8.987365e-04_{8.3e-05}$	$9.190031e-04_{5.1e-05}$
WFG6	$7.328672e-05_{3.7e-06}$	$9.137049e-05_{4.6e-08}$	$5.268868e-05_{2.4e-06}$
WFG7	$2.842715e-05_{2.2e-07}$	$4.053810e-05_{1.8e-08}$	$2.202683e-05_{3.0e-07}$
WFG8	$2.972886e-03_{1.2e-03}$	$2.690646e-03_{1.3e-03}$	$2.447889e-03_{1.3e-03}$
WFG9	$2.997811e-05_{2.6e-07}$	$4.058226e-05_{1.7e-07}$	$2.311125e-05_{6.5e-07}$
DTLZ1	$3.199129e-04_{1.3e-05}$	$3.470590e-04_{9.9e-07}$	$2.666947e-04_{5.0e-06}$
DTLZ2	$4.126831e-04_{8.9e-06}$	$4.307759e-04_{2.0e-06}$	$3.401370e-04_{5.6e-06}$
DTLZ3	$6.701955e-04_{1.9e-05}$	$7.209664e-04_{3.7e-06}$	$5.485917e-04_{8.3e-06}$
DTLZ4	$7.959802e-04_{5.7e-05}$	$8.381398e-04_{6.4e-05}$	$5.614028e-04_{5.8e-05}$
DTLZ5	$4.629690e-06_{7.8e-08}$	$1.514716e-05_{5.3e-08}$	$3.817541e-06_{1.1e-07}$
DTLZ6	$1.127554e-05_{2.9e-07}$	$3.453401e-05_{3.1e-08}$	$9.029561e-06_{3.2e-07}$
DTLZ7	$1.923883e-02_{7.1e-03}$	$2.005448e-02_{6.9e-03}$	$1.836414e-02_{6.8e-03}$

2.7.8. CPU time comparison

To compare the runtime of the algorithms, this study analyzed the CPU time cost of the proposed algorithm (DPPCP) with two baselines (NSGA-II and MOEA/D) and the co-evolutionary DPP algorithm. This study conducted comparisons across 31 problems. To get the most accurate assessment, all algorithms are implemented in *jMetal5* (an integrated Java framework). It can be downloaded from `http://jmetal.github.io/jMetal/`. The author runs multithreading with 8 cores on computers configured as Intel Xeon E5-2620, 16 GB Ram. Experimental results are shown in Figure.2.18-2.20. This study experimented with two different generation parameters, which are 10 (Figure.2.18-2.19) and 1000 (Figure.2.20).

Suppose that M is the objective number, N is the population size, and T is the neighbor size. The time complexity of MOEA/D in one genera-

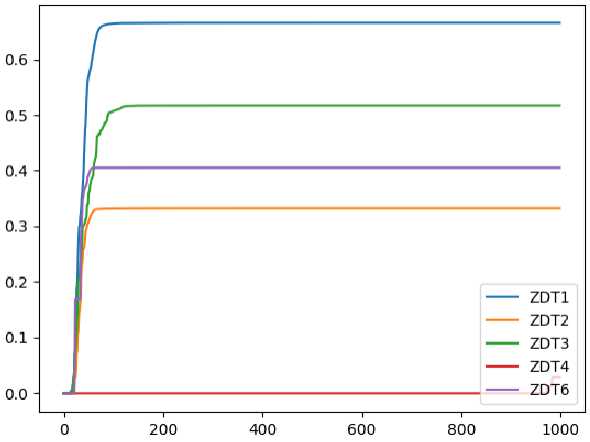


Figure 2.10: The HV values of ZDT problems in different stages of evolution

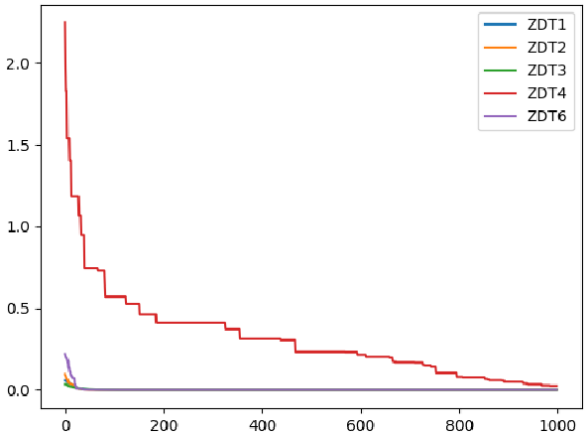


Figure 2.11: The IGD values of ZDT problems in different stages of evolution

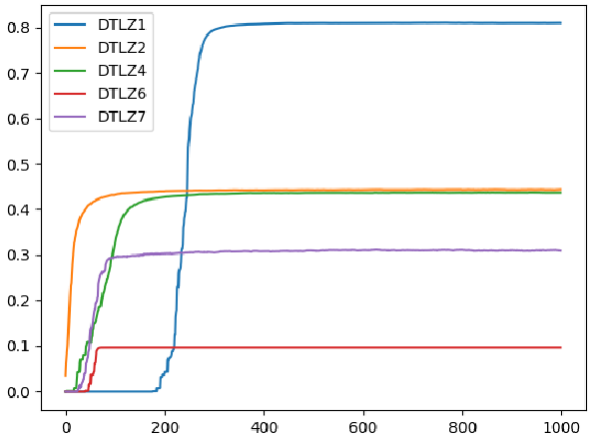


Figure 2.12: The HV values of DTLZ problems in different stages of evolution

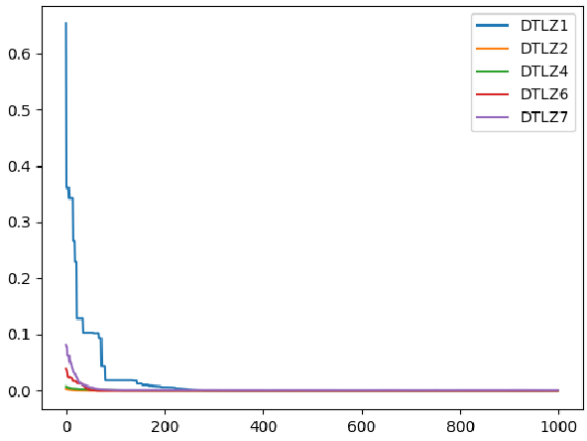


Figure 2.13: The IGD values of DTLZ problems in different stages of evolution

tion (iteration) is only $O(NTM)$, where M, T≪N. Meanwhile, $O(MN^2)$ is the time complexity of the NSGA-II algorithm. The DPPCP and DPP algorithms maintain two coevolving populations. The main running steps of these two algorithms are similar to those of the MOEA/D algorithm. Thus, the complexity of these main steps is still $O(NTM)$. However, small steps in the algorithms often have to be processed twice for two populations, so the calculation time will be longer than the baseline algorithms. Results with CPU time show this clearly. As you can see in Figure.2.18, the MOEA/D algorithm uses the least CPU time, followed by the NSGA-II algorithm. These two baseline algorithms take

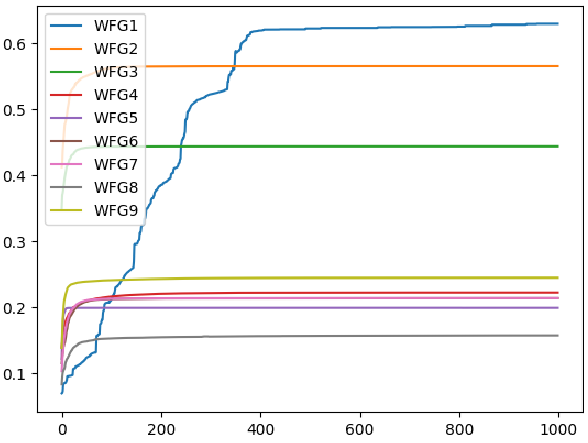


Figure 2.14: The HV values of WFG problems in different stages of evolution

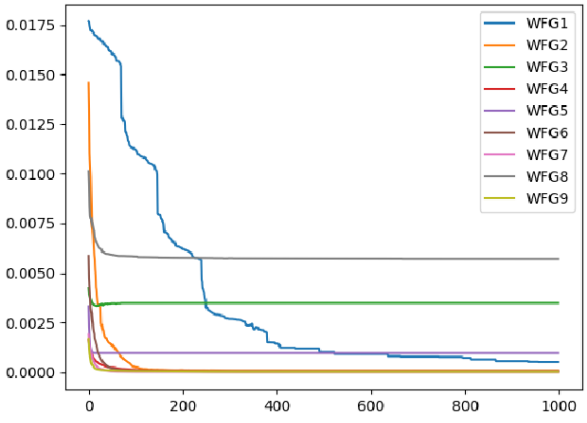


Figure 2.15: The IGD values of WFG problems in different stages of evolution

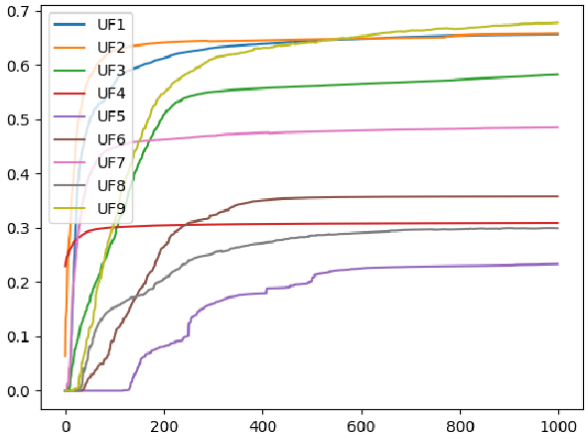


Figure 2.16: The HV values of UF problems in different stages of evolution

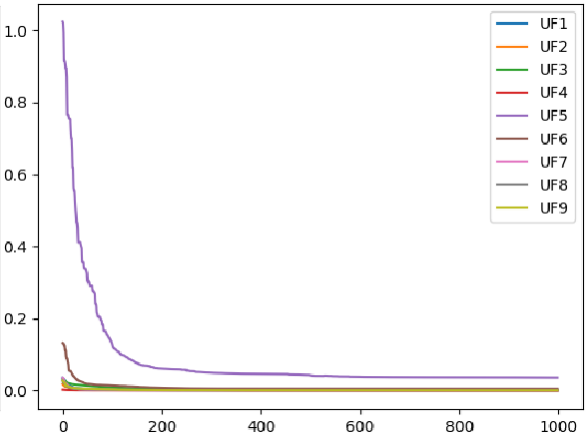


Figure 2.17: The IGD values of UF problems in different stages of evolution

less time than the two DPPCP and DPP algorithms. Figure.2.19 shows the comparison of the two co-evolution algorithms. The white boxes indicate that the DPPCP algorithm runs faster, and vice versa with the black boxes. It is evident that with a loop count of 10, the DPPCP algorithm runs faster than DPP in most test cases. The result is similar for these two algorithms when the number of iterations increases to 1000, as shown in Figure.2.20. The explanation for this result may be in the step of updating the child solution in the A_d population. While DPPCP updates with a limited number ($K < T$) of neighborhood solutions (the time complexity is $O(K)$), in DPP the author proceeds through all sub-

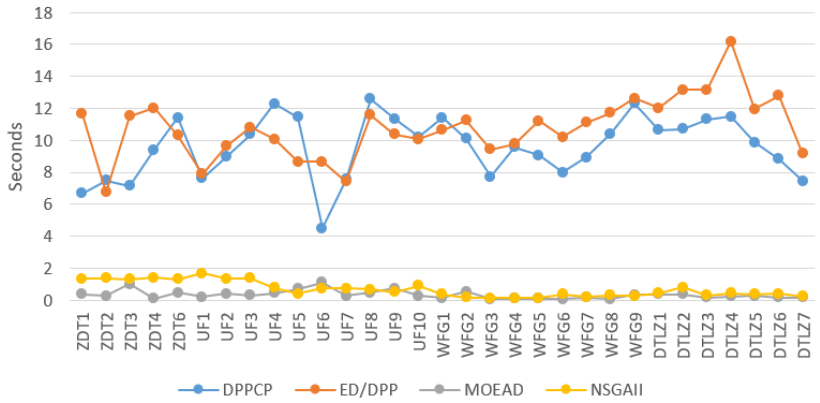


Figure 2.18: CPU time comparisons between algorithms on different test instances (with the number of generations is 10)

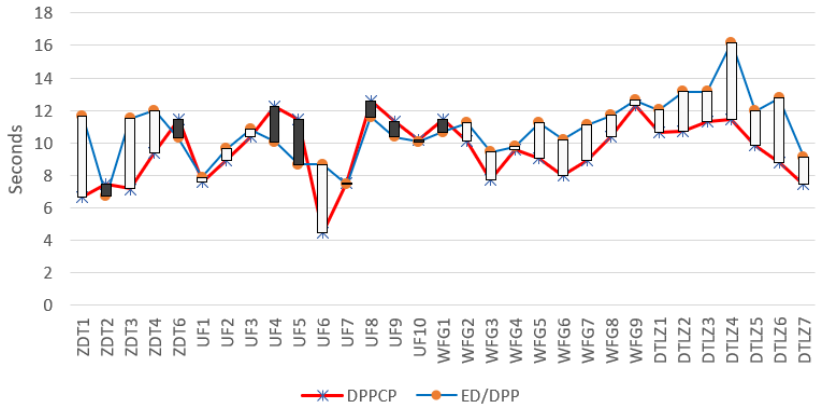


Figure 2.19: CPU time comparisons between DPPCP and ED/DPP for algorithms on different test instances (with the number of generations is 10)

regions to calculate distances, find the nearest sub-region, and compare the child solution with a solution in this sub-region to update (the time complexity is $O(NM)$). This is the main reason for the difference in CPU time between these two algorithms.

2.8. Summary

In this chapter, the author presents two DPP-based algorithms for balancing convergence and diversity in MOEAs. Specifically, a co-operative co-evolutionary algorithm (DPP2) and a competitive co-evolutionary (named DPPCP) algorithm are presented. The novelties of these algorithms include a new restricted competitive mating selection mechanism

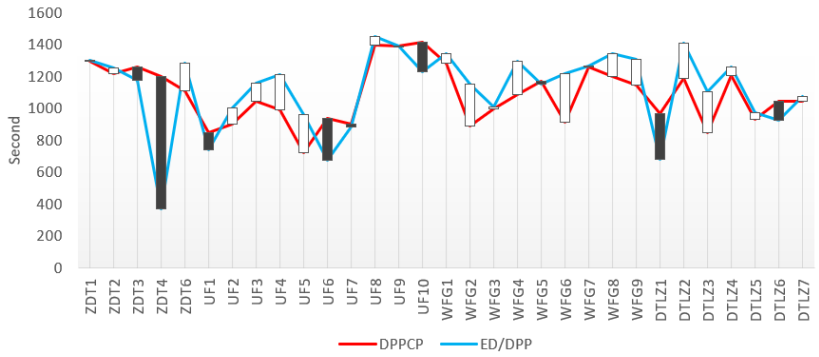


Figure 2.20: CPU time comparisons between DPPCP and ED/DPP on different test instances (with the number of generations is 1000)

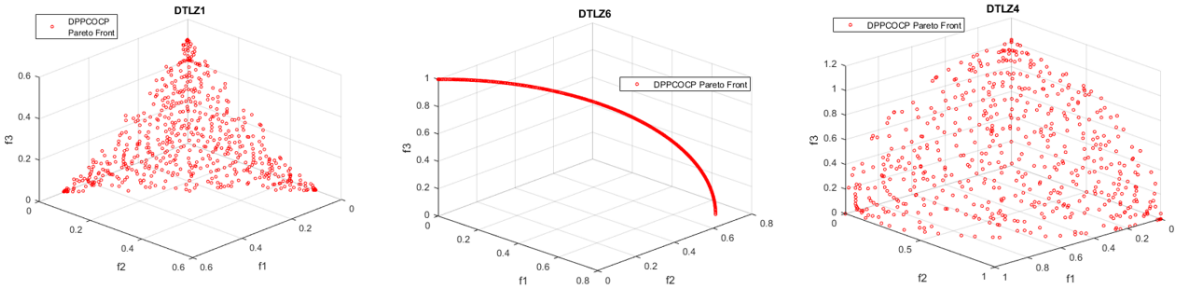


Figure 2.21: Plots of final solutions found by the DPPCP algorithm on DTLZ test instances

(RMS2); a new neighborhood-based selection mechanism (NBSMS) for choosing individuals for each population; a co-operative and competitive mechanism to force two offspring to associate with one another. By comparing DPPCP and DPP2 with baseline algorithms, the empirical results pointed out the efficacy of the co-evolutionary methods in balancing diversity and convergence for solving MOPs.

The proposed methods in this chapter were published in the IEEE Access journal (SCIE, Q1) [J1], and the Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES) conference (Scopus) [C1].

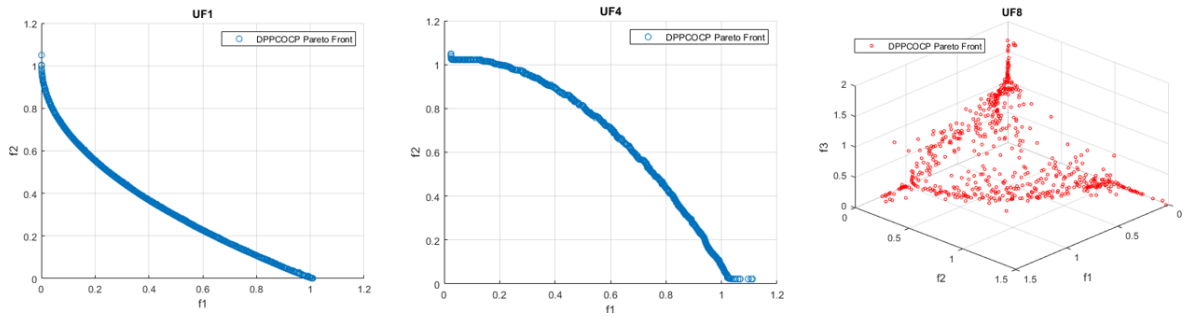


Figure 2.22: Plots of final solutions found by the DPPCP algorithm on UF test instances

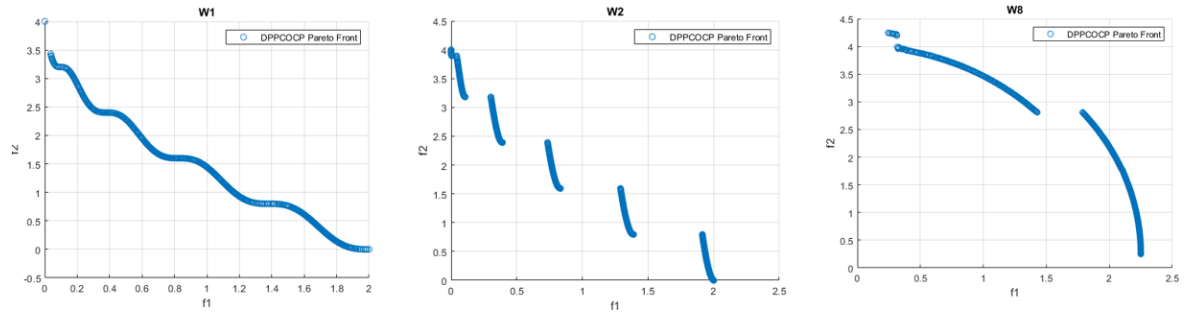


Figure 2.23: Plots of final solutions found by the DPPCP algorithm on WFG test instances

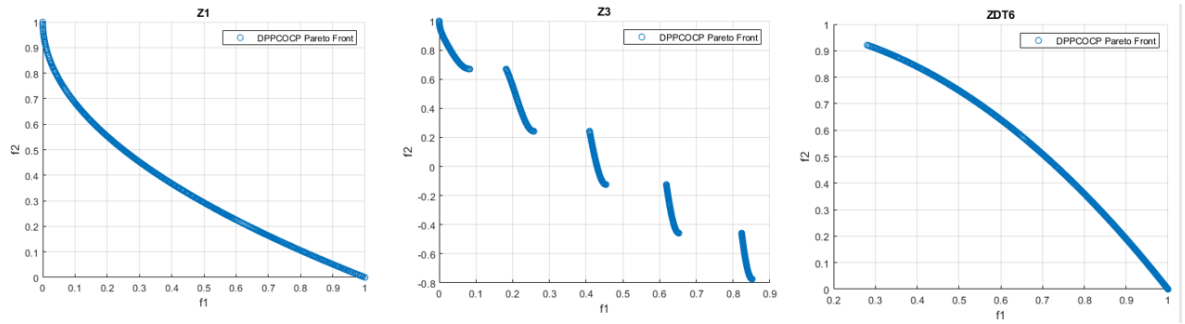


Figure 2.24: Plots of final solutions found by DPPCP algorithm on ZDT test instancess

Chapter 3

THE APPLICATION OF MULTI-OBJECTIVE CO-EVOLUTIONARY OPTIMIZATION METHODS FOR CLASSIFICATION PROBLEMS

In Chapter 2, a full explanation of dual-population methods for solving standard multi-objective optimization problems is presented. The highlight of DPP-based methods is the ability to find solution sets that have both convergence and diversity factors. From the success of these algorithms for basic multi-objective optimization problems, in this chapter, the author continues to expand these algorithms into solving particular machine learning challenges. In particular, a dual-population competitive co-evolutionary algorithm (i.e., DPPCP) with the capability of generating optimal solutions that meet both convergence and diversity criteria will be combined with an ensemble learning algorithm along with resampling methods to make an algorithm named IBDPPCP for the classification of imbalanced data. In addition, another new algorithm (named IBMCCA) is proposed for the imbalanced data classification. The idea of IBMCCA is inherited from the studies published in [J2] and [C2, C3]. These two proposed methods are also improved versions of the method published in [C6, C7]. The details of these algorithms are presented in the following sections:

3.1. Introduction

The difficulty of imbalanced datasets for traditional machine learning algorithms is that these algorithms often focus on optimizing accuracy

as well as generalization. This makes them biased toward the majority-class examples, and the minority ones are not well modeled into the final system. As a result, the classification results for the minority classes are often low. However, in reality, the minority classes often play vital roles and are the most meaningful to explore. This can be seen clearly in medical applications [113], finance [12], etc. The important point to note here is that the imbalanced ratio is not the only factor affecting the classification results; another factor that needs to be mentioned is the problem of overlapping between classes. This overlap problem is related to the number of redundant features in a dataset. When the dataset has more redundant and irrelevant features, the overlap becomes more complex, and the classification results are worse. Therefore, it is necessary to handle both problems: ***imbalance and overlap between classes.***

The problem of overlap between classes can be effectively solved by ***feature selection (FS)*** methods [3,32]. The purpose of FS is to make overlapping areas simpler, hence making it easier to generate rules that distinguish between classes. Meanwhile, by removing the instances of the majority class, a mechanism of ***instance selection (IS)*** [83] is well suited to tackle the imbalance problem. Additionally, the IS has extra benefits. *First*, it helps to eliminate redundant, irrelevant, noisy, and borderline instances that could affect the classification results [49]. *Second*, this method makes the training process more effective when dealing with large problems, and the final model may also be simpler [80]. However, it must be emphasized that FS and IS cannot resolve all issues relating to data imbalances. Instead, this data still needs to go through the pre-processing step (e.g., upper or under-resampling).

Recent studies have demonstrated the effectiveness of ***Multi-Objective Evolutionary Optimization (MOEA)*** [2] and ***ensemble learn-***

ing [48] for imbalanced classification. The non-dominated solutions of the Pareto front from the MOEA can be directly combined into an ensemble of classifiers. This is a key point that makes the combination of MOEA and ensemble learning effective in solving imbalanced classification problems. In [42], Fernández et al. proposed a multi-objective evolutionary algorithm (named ***EFIS-MOEA***) that combines ensemble learning, instance selection, and feature selection problems. This is one of the typical case studies for using ensemble learning algorithms to solve classification problems with imbalanced data utilizing a multi-objective optimization approach. In this study, the authors proposed an method to solve both problems at the same time: feature selection and instance selection. By using a Pareto-based multi-objective optimization algorithm (i.e., NSGA-II), a number of output optimal solutions are selected to form a set of weak learners in ensemble learning. The base algorithm used as the weak learner here is the decision tree algorithm C4.5 [98]. This algorithm is not only capable of selecting features based on the intrinsic characteristics of the data but has also been proven to be a good choice when developing ensemble learning systems [101]. ***Inspired by the success of this algorithm, this thesis proposes an algorithm named IBDPPCP as an extension and further development of this algorithm***. The proposed algorithm differs from the original work (i.e., EFIS-MOEA) in three key points:

- In the data preprocessing step, EFIS-MOEA only uses SMOTE to generate more samples for the minority class. The noisy data in the overlapping area will hopefully be eliminated through an instance selection solution. This method has no control over how much of the noise is removed. To overcome this phenomenon, in IBDPPCP, the author uses a combination of upper and lower sampling techniques (i.e., SMOTE-ENN). By using ENN, these noises will be actively

eliminated with more certainty.

- In IBDPPCP, the author still uses the individual encryption solution to solve both feature selection and instance selection problems, as in EFIS-MOEA. However, the difference here is in the individual encoding. While EFIS-MOEA utilizes binary encoding, IBDPPCP uses real encoding, with each bit value representing the probability that the bit will be selected. This encoding type is consistent with the decomposition-based algorithm that IBDPPCP uses.
- Both EFIS-MOEA and IBDPPCP use a multi-objective optimization algorithm to find optimal solutions that act as base learners for the ensemble learning algorithm. However, there is a difference here. In EFIS-MOEA, the multi-objective algorithm used is NSGA-II. This algorithm prioritizes convergence over diversity, whereas the IBDPPCP algorithm uses a DPP-based competitive co-evolutionary algorithm (i.e., DPPCP). This algorithm has been proven to produce solutions that ensure both convergence and diversity. They are the two decisive factors that help the ensemble learning algorithm execute most effectively.

The co-operative co-evolution method is an efficient evolutionary algorithm for solving the FS and IS in classification problems. In [111], the authors used a single-objective co-evolutionary method (named COCEA) to solve both feature selection and optimization problems of ANNs simultaneously. In this study, two populations were employed: one for selecting features and another for optimizing ANNs. In the feature population, the authors use a binary string to encode each individual and a genetic algorithm (GA) to evolve, whereas the second population utilizes real number encoding and the differential evolution algorithm (DE) to evolve. The COCEA algorithm is tested on four different datasets. The results show that the COCEA algorithm has high classification accu-

racy. ***The weakness of the COCEA lies in its single-objective function***. Although the COCEA uses two different populations, one for evolving the classifier and the other for evolving the feature set, both populations use the same fitness function. This is the classification accuracy (MSE). This means both populations only focus on optimizing this goal, and the criteria for feature selection are still secondary. This cannot guarantee that the number of features selected will be reduced. This is the main reason why the number of features retained by the COCEA is often large and uncontrollable. An extended version of the COCEA algorithm (named E-SOCA) is proposed in [109]. In this, the authors have combined the co-evolutionary method with an ensemble learning algorithm; the best individuals in each generation are stored in a list called *BestList* during the co-evolutionary process. These individuals are then used as weak learners for the Adaboost algorithm. *Although the classification results are better than COCEA, the shortcomings of COCEA are still unresolved since both use a single-objective function.* This problem can be handled by applying a multi-objective optimization algorithm instead of a single-objective algorithm. In [110], the authors presented a multi-objective co-operative co-evolutionary algorithm with dual populations for designing the artificial neural network (named MCCA) with feature selection. In this research, the authors used a multi-objective optimization algorithm with classification accuracy (ACC) and feature subset size (FS) as the two main objectives to evaluate the fitness of individuals. The authors have run experiments with three different remote sensing image datasets from the UCI and Statlog Repository [46]. Experimental results show that by using a multi-objective optimization method, MCCA outperforms SOCA not only in classification accuracy but also in the selected number of features. ***Inspired by the work in [111] and [109], in this study, the author pro-***

poses a new multi-objective co-operative co-evolution method with dual population for classification with imbalanced data (named IBMCCA). In comparison with the COCEA and E-SOCA algorithms, the proposed algorithm has the following similarities and differences:

+ *The feature individual encoding:* Both IBMCCA and COCEA use binary encoding to encode the individual of the feature population. Meanwhile, MCCA uses real strings to represent an individual, then uses a method to convert it to a binary string in order to calculate its fitness as well as to know which features will be chosen.

+ *The co-evolutionary algorithms*: In COCEA, the authors used two different single-objective evolutionary algorithms (the GA and DE algorithms) to evolve two populations during a co-evolution process. The objective function is the mean squared error (MSE) metric. In MCCA and IBMCCA, the author utilizes a multi-objective evolutionary algorithm (NSGA-II algorithms) to evolve each population. The difference between IBMCCA and MCCA is in the objective functions used in each population. While MCCA uses (ACC and FS) for the feature population and (MSE and ACC) for the remaining population. IBMCCA utilizes (AUC and FS) for the feature population and (AUC and IS) for the other population.

+ *The ensemble learning*: While E-SOCA and IBMCCA utilize ensemble learning, MCCA does not. The difference between E-SOCA and IBMCCA is that S-COCA uses a boosting algorithm (i.e., Adaboost), while IBMCCA utilizes a voting mechanism.

It can be seen that ***the co-evolutionary model in IBMCCA is a combination of the multi-objective co-evolutionary model in MCCA and the ensemble learning mechanism in E-SOCA***. Details of this proposed algorithm are presented in the following sections.

3.2. A multi-objective competitive co-evolutionary method for classification with imbalanced data (IBDPPCP)

The proposed algorithm model is shown in Figure 3.1 and the pseudo-code for this algorithm is presented in Algorithm 14. There are three main phases: *Data pre-processing, the co-evolutionary process, and ensemble-based decision-making*. The general idea of the algorithm is as follows: Encoding each individual into a couple of feature sets (FS) and instance sets (IS) with the hope of finding the optimal ones that have both key features as well as important instances to help solve the imbalanced dataset problem. The author's idea is to use a multi-objective competitive co-evolutionary algorithm (i.e., DPPCP) to find the set of optimal individuals, then combine these individuals with an ensemble learning algorithm. It should be noted that to boost the performance of the ensemble learning algorithm, the weak learners should satisfy two criteria: *having diversity as well as good classification performance*. The multi-objective optimization algorithm helps generate weak learners that satisfy both of these two criteria. After the evolutionary process, all individuals in the final population are used as weak learners in the ensemble learning algorithm. A voting mechanism is used to determine the final result. This study uses a DPP-based algorithm (named IBDPPCP) as the multi-objective optimization algorithm and utilizes the C4.5 algorithm as the base learner to solve this problem.

3.2.1. Individual encoding

An individual stands for a problem-solving strategy. As stated at the beginning of this section, we need to find solutions to solve two issues: The first is class overlap, and the second is unequal class distributions. Finding the boundary to separate becomes increasingly challenging for the first issue as the number of features and dimensions rises. There

are frequently redundant and unrelated features, and not all characteristics accurately describe the problem's data domain. The overlapping phenomenon is further fueled by these data. The effective solution to this problem is feature selection [32]. Therefore, this feature selection problem must be solved by individual encoding.

In the second issue, the data imbalance is solved by either increasing or decreasing the data until the proper ratio is achieved. There are two straightforward strategies for this issue: The first way is to use a combination with a sampling technique (such as SMOTE, ENN, etc.), and the second one involves using the instance selection (IS) method. A question arises here: if the sampling method has been used to bring the data to equilibrium, is there a need to use more IS? The essence of IS is to select different groups of data. However, this selection takes into account the selection probability of the samples. That is, the samples that give good classification results will be kept, and the other samples will be discarded. IS facilitates two tasks for us: In addition to assisting in the creation of more diverse sub-datasets, it also aids in the elimination of patterns in the overlapping region. This will enhance the performance of the ensemble learning method. As a result, even after doing data sampling, IS should be implemented.

From the above arguments, this study shows that FS and IS are two problems that need to be solved simultaneously. The author decides to encode an instance as a sequence of two substrings (Figure 3.2):

- *FS Substring*: The representation of a feature set
- *IS Substring*: The representation of an instance set.

One point to note here is that instead of encoding IS and FS as binary strings as traditional methods often do, these two strings in this situation both contain real numbers between [0, 1]. These numbers show the likelihood that a feature or sample will be chosen or not. The author

chooses a threshold of 0.5 to assess whether a gene is kept or removed. This implies that positions with values between 0.5 and 1 will be chosen, and positions with values lower than 0.5 will be rejected. A new dataset will be created for each of the encodings, which is then utilized to build a decision tree to determine the classification outcome. The Figure 3.3 illustrates how to create a decision tree from an individual.

In Figure 3.3, X is the original training dataset with S rows (or samples) and D columns (or features); X^F is the selected training dataset (with S rows and F columns) after removing some columns (i.e., positions with a value of '0' on the FS substring); X_I^F is X^F dataset (with I rows and F columns) after eliminating some rows (i.e., positions with a value of 0 on the IS substring).

3.2.2. Objective functions

There are two key objectives the author wishes to accomplish with this study. Increasing identification across all data classes (including minority and majority) is the first one. The second is to minimize the number of samples selected, or, in other words, to maximize the removal of bad samples.

To solve the first goal, it is necessary to have a metric that can evaluate the classifier's ability in all classes, not just favor samples in the majority or minority class. In common classification problems, the classification accuracy (ACC) metric is often chosen to evaluate the results. However, the disadvantage of this measurement is that it only evaluates the whole data set and does not represent the classification results for each class. Because of this feature, the ACC is not suitable for problems involving imbalanced classification. Instead, the AUC index is often used. The AUC [44] is calculated based on the ROC (receiving operating curve) to assess how well the model's classification ability is. The larger the AUC value, the better the algorithm's performance is. Therefore, here

the author chooses AUC as an objective function and takes the opposite sign to return to the problem of finding the minimum.

In the second objective, the removal of bad patterns not only makes the execution speed faster but also better. This has a great effect when solving problems with a large number of samples. However, removing too many samples can have the opposite effect on the learning ability of the model. How to delete to get the best execution performance? This is not a trivial answer to the fact that the two objectives selected here were in conflict. Through the above analysis, the author decided to choose two objective functions as follows:

$$\begin{cases} OBJ_1 = \quad AUC \\ OBJ_2 = \sum_{i=0}^{N-1} IS_i \end{cases} \tag{3.1}$$

where N is the number of samples of the training dataset; IS_i is a binary value (0 or 1) converted from the probability of selection. The task now is to minimize these two functions. This implies that for any solution, the smaller the IS and the higher the AUC, the better it is.

3.2.3. The IBDPPCP algorithm

The pseudocode of the algorithm is presented in Algorithm 14. There are three main steps: *Data preprocessing, co-evolutionary processes, and ensemble learning*. In the data preprocessing step, the original data goes through a process of re-sampling and cleaning. Since the original data is unbalanced, the SMOTE-ENN algorithm helps generate more data for the minority classes to make the data more balanced. After the sampling step, duplicate data may appear. Therefore, it is necessary to clean up these data so that they do not affect the training results. The data after preprocessing is the input for the co-evolutionary process. The **DPPCP (Algorithm.10)** is used to find a set of individuals (denoted

Algorithm 14: The IBDPPCP algorithm

input : Dataset
output: FinalResult

```
1  BestArchive ← Θ
2  #Step 1: Data preprocessing
3  Dataset ← DataSampling(DataSet);
4  Dataset ← DuplicateRemoving(DataSet);
5  #Step 2: The Co-evolutionary process
6  BestArchive= DPPCP (Dataset);
7  #Step 3: Ensemble learning
8  Classifiers = BuildTrees(BestArchive, Dataset)
9  FinalResult = EnsembleLearning(Classifiers, Dataset)
10 Return FinalResult;
```

BestArchive) that satisfy both convergence and diversity criteria. Each individual is encoded as a couple of IS and FS substrings. Corresponding to an individual is a new dataset that is generated from the input dataset (Figure 3.3). A decision tree (i.e., C4.5) is constructed using these new datasets. As a result, IBDPPCP generates a collection of *Classifiers*. Additionally, these classifiers will be used as input for the ensemble learning method to make the final decision using the voting mechanism.

As mentioned above, there are two important factors determining the success of ensemble learning algorithms: *predictive performance and diversity*. The first requires that classifiers (or weak learners) achieve a certain level of performance rather than being randomly initialized. By using an objective function of AUC, this factor can be guaranteed. With the second factor, the classifiers are required to have a certain difference to avoid premature convergence to local extremes as well as help make the most diverse decisions. These factors help the voting method produce better results. In this study, the author encodes each individual as a set of IS and FS. This is similar to creating subsets that differ in both the number of rows and the number of columns of data. Thereby, diversity is guaranteed.

A highlight of the DPP-based solution, as mentioned in Chapter 2, is the ability to find a set of solutions that ensure both convergence and diversity. This gives the author the hypothesis that it may be more suitable than the solution using NSGA-II for finding a set of individuals (or models) that satisfy both criteria that an ensemble learning algorithm requires. This is the reason to choose a DPP-based algorim for solving the imbalanced data classification problem.

3.3. A multi-objective co-operative co-evolutionary method for classification with imbalanced data (IBMCCA)

The IBDPPCP algorithm utilizes the DPPCP to find an optimal solution set. DPPCP helps to solve both FS and IS problems simultaneously. There is another algorithm using dual populations to solve this problem that will be introduced in this section (i.e., *IBMCCA*). There are two main differences between these two algorithms. The first one is individual encoding. In IBDPPCP, the individuals in the two populations use the same encoding (i.e., the two substrings FS and IS). In IBMCCA, each individual is a separate substring. Objective functions are the second difference. Because the role of each population is different, in IBMCCA, each of them uses different objective functions, whereas, in IBDPPCP, both populations use the same objective functions. The flowchart of the proposed method is shown in Figure 3.4. There are two populations that evolve simultaneously to solve two tasks: feature selection and instance selection. The first population (called the feature population or population 1) includes individuals representing different ways of selecting features. The second one (called the instance population or population 2) contains individuals that each represents a subset of the original dataset. In the process of co-evolution, in order to calculate the fitness value, each individual in the first population needs to be

associated with the individual in the second population, and vice versa. Undergoing a co-evolutionary process, the output is a combination of the best individuals from two populations. A more detailed explanation of the IBMCCA method is described in the following sections.

Algorithm 15: The IBMCCA algorithm

input : DataSet
output: FinalResult

#***Step 1: Data preprocessing***
DataSet $\leftarrow PreProcessing(DataSet)$;
#***Step 2: Population initiation***
$P1 \leftarrow InitializePopulation1(N)$
$P2 \leftarrow InitializePopulation2(N)$
$Elite1 \leftarrow [11\ldots1]$
$ElitePool \leftarrow \Theta; BestArchive \leftarrow \Theta$
ElitePool.Add(Elite1);
CalculateFitness(P2, Elite1, DataSet);
Eltite2 $\leftarrow Sort(P2)$;
Pool.Add(Elite2);
#***Step 3: The Co-evolutionary process***
while *Stop condition false* **do**
    Reproduction(P1, Eltite2;
    Eltite1 $\leftarrow Sort(P1)$;
    Pool.Update(Elite1);
    Reproduction(P2, Eltite1);
    Eltite2 $\leftarrow Sort(P2)$;
    Pool.Update(Elite2);
#***Step 4: Ensemble learning***
BestArchive= **Union** ($\mathbf{Front0}_{P1}$, ($\mathbf{Front0}_{P2}$));
Classifiers = **BuildTrees**(BestArchive, Dataset)
FinalResult = **EnsembleLearning**(Classifiers, Dataset)
Return FinalResult;

3.3.1. Individual encoding

Two populations have the same coding strategy. It is supposed that the feature population has N individuals ($Ind1_1, Ind1_2, ..., Ind1_N$). Each individual $Ind1_i$ is encoded as a binary string. In particular, the binary value is used to determine whether a feature is selected or not. If a bit value is "1", the corresponding feature is selected, and vice versa.

For example, with individual $Ind1_1 = [011000]$, the second and third features are selected, and the other ones are eliminated.

3.3.2. Objective functions

The objective functions in the evolutionary algorithms are designed to calculate fitness values, which can be used to evaluate individuals. In this study, the author utilizes the multi-objective evolutionary algorithm to evolve two populations. Because of the different purposes of each population, the objective functions are varied. The purpose of the feature population is to find individuals that have not only the least number of selected features but also the highest AUC value. Therefore, two chosen objectives for this population are the AUC and the number of selected features. FS stands for feature set. FS_i is a binary value (0 or 1) converted from the probability of selection of a feature. Meanwhile, the second population (i.e., the instance population) tries to find individuals with the least number of selected instances as well as the highest AUC value. Therefore, AUC and the number of selected samples are chosen as two objectives for this population. IS stands for instance set, IS_i is a binary value (0 or 1) converted from the probability of selection of an instance. Suppose D is the number of features and S is the number of samples. The formula for the objective functions of the two populations is as follows:

$$Population1 : \begin{cases} OBJ_1 = -AUC \\ OBJ_2 = \sum_{i=0}^{D-1} FS_i \end{cases} \tag{3.2}$$

$$Population2 : \begin{cases} OBJ_1 = -AUC \\ OBJ_2 = \sum_{i=0}^{S-1} IS_i \end{cases} \tag{3.3}$$

3.3.3. The IBMCCA algorithm

The implementation of the whole procedure can be divided into the following four main steps (Algorithm 15).

Step 1: Data preprocessing

Similar to the IBDPPCP algorithm, IBMCCA also uses a data resampling method (i.e., SMOTE/ENN) as a preprocessing step to make the data more balanced. This data is then cleaned by removing duplicate records.

Step 2: Population initiation

In general, each individual in two populations is randomly assigned an array of binary values. It is supposed that both populations are the same size as N. The total number of features is D, and the size of the samples is M. Then, N individuals with the length of D are randomly generated with binary values to form the initial feature population, and N individuals with the length of M are randomly generated with binary values to create the initial instance population. It is noted that this initialization process has to ensure two conditions: First, each individual in both populations must contain at least one bit with the value "1" (to ensure there is always at least one feature and one sample selected), and both populations must always have an individual whose all values are "1" (to ensure that all features and samples are selected). To begin, an individual in population 1 with all of the values set to "1" will be used to calculate the fitness values of the entire population 2. After that, a ranking based on the fitness value is performed, and the best individual is chosen as an elite (named *Elite2*) and put in a place that the author named the *Elite Pool* (this pool contains the two best individuals found from each population during the co-evolutionary process). Following that, *Elite2* is utilized to determine the fitness values of the individuals in Population 1. After being ranked, the best one (named *Elite1*) is put

into the Elite Pool.

Step 3: Co-evolutionary process

After the initialization step, Elite Pool consists of $< Elite1, Elite2 >$. These elites are inputs for the process of co-evolution. This is an iterative process, and basically, the steps are similar to the initialization step when each population evolves and selects the best one to update to the Elite pool. The elite in this population is then used as input for the other population in the next iteration. The elitist selection mechanism is used in both populations in order to keep the best solutions during the whole co-evolution process. If there is an individual (denoted by Elite1t at the loop t) dominating the corresponding elite in the pool (i.e., *Elite1*), the *Elite1* is substituted by $Elite1^t$ and the fitness of the new *Elite1* is updated as well. There are two main methods in this step: elite pool selection and updating mechanisms.

+ *Elite selection mechanism*: In this study, the authors use the multi-objective optimization algorithm NSGA-II to evolve populations. It is difficult to find exactly the best individual, like in a single-objective optimization algorithm. Instead, a set of non-dominated individuals (called Pareto-optimal solutions) will be selected. In this study, the set of solutions belonging to front 0 will be selected as candidates to find the best individual to put into the Elite Pool. The author prioritizes the AUC because this is the criteria used to evaluate the final classification results. Then, the ACC is used as a criterion to find the best individual from front 0.

+ *Updating Elite pool mechanism:* After selecting the best individual of each generation, this individual will be compared to the elite in the pool to check whether it will be updated or not. With population 1, in order to be updated, the best individual must be better than *Elite1* in terms of the AUC and SF criteria. Whereas in population 2, the

condition is that the best individual is better than *Elite2* in terms of the AUC and SI criteria. In order to update *Elite1*, both the AUC and the SF criteria must be considered. If the SF is the only one considered, many individuals may have very low AUC values. This means the final classification result will be poor. Similarly, if the authors only use the AUC, there will be many individuals with good AUC values but large SF. This does not guarantee that the result will always be good in terms of the SF value (this is a drawback of the COCEA algorithm [111] and the E-SOCA algorithm [109]).

Step 4: The ensemble learning

After the co-evolutionary process has ended, a list of the best individuals is found in the first two fronts (i.e., *Front0*) of two populations and combined in the *BestArchive* list. A note here is that the individuals on each *Front0* of the two populations have been guaranteed diversity (based on *Crowding distance* method) and convergence. The combination of two *Fronts* in two different populations helps to ensure these two factors. Therefore, it can be seen that this selection mechanism helps ***IBMCCA can find individuals with both convergence and diversity criteria***. The question is, which individuals are selected to calculate the final result? It is usually not easy to find a single individual that gives good and stable results with many different data sets. Because each individual is only best suited to certain types of data, this is the reason why the author chose the ensemble learning solution. The ensemble learning phase is fed by a set of individuals stored in the *BestArchive*. Each individual in the *BestArchive* is used to create a subset from the Dataset (Figure 3.3). Each subset is then used to build a decision tree (i.e., C4.5), and each decision tree serves as a *Classifier* in ensemble learning. Here, the author combines the *Classifiers*' results with the hard voting method to make the final result.

3.4. Experimental results

3.4.1. Experimental datasets

The experimental datasets consist of 42 standard imbalanced datasets. These data sets are divided into two groups: imbalance ratios lower than 9 and imbalance ratios higher than 9. All these datasets can be downloaded from the KEEL dataset repository. The details of the dataset parameters are described in Tables 3.2, 3.3. In which the *#Ex* parameter represents the number of samples, *#Atts* represents the number of features, *#IR* represents the imbalanced rate, *#FD* represents the degree of overlap between the classes. The smaller the *FD* value is, the greater the overlap is and the harder it is to separate. *FD* and *IR* are two parameters that represent the difficulty of classification. These datasets can be further split into two smaller groups based on the values of the *FD* metric: (1) a set of 22 problems with little overlap when $FD > 1.5$; and (2) a set of 20 problems with a lot of overlap when $FD < 1.5$. The hardest issues to solve are found in this second group. In this study, each *five-fold cross-validation* is conducted twenty-five times to reflect the stochastic character of the learning techniques. As a result, experimental results are calculated using the average of *125 runs* for each algorithm and dataset.

3.4.2. Parameter setting

The details of the initial parameters for the algorithms are shown in Table 3.1. These values are common for all test cases. They were selected according to the recommendations of the corresponding authors, and it is also the default setting of the parameters of the libraries.

Table 3.1: Initial parameters

Algorithm	Parameters	Value
IBDPPCP and IBMCCA	Population size	100
	The number of evaluations	1000
	Crossover rate	0.8
	Mutation rate	0.025
C4.5	Pruning	True
	Confidence	0.25
	Instances Per Leaf	2
SMOTE	Neighbors	5
ENN	Neighbors	3

Table 3.3: Imbalance ratio higher than 9

No	Name	#Attributes (R/I/N)		#Examples	IR	FD
Imbalance ratio higher than 9 - Part I						
1	yeast-2_vs_4	**8**	(8/0/0)	514	9.08	1.5790
2	yeast-0-5-6-7-9_vs_4	**8**	(8/0/0)	528	9.35	1.0510
3	vowel0	**13**	(10/3/0)	988	9.98	2.4580
4	glass-0-1-6_vs_2	**9**	(9/0/0)	192	10.29	0.2692
5	glass2	**9**	(9/0/0)	214	11.59	0.3952
6	shuttle-c0-vs-c4	**9**	(0/9/0)	1829	13.87	0.3534
7	yeast-1_vs_7	**7**	(7/0/0)	459	14.3	12.970
8	glass4	**9**	(9/0/0)	214	15.47	1.4690
9	ecoli4	**7**	(7/0/0)	336	15.8	3.2470
10	abalone9-18	**8**	(7/0/1)	731	16.4	0.6320
11	glass-0-1-6_vs_5	**9**	(9/0/0)	184	19.44	1.8510
12	shuttle-c2-vs-c4	**9**	(0/9/0)	129	20.5	12.130
13	yeast-1-4-5-8_vs_7	**8**	(8/0/0)	693	22.1	0.1757
14	glass5	**9**	(9/0/0)	214	22.78	1.0190
15	yeast-2_vs_8	**8**	(8/0/0)	482	23.1	1.1420
16	yeast4	**8**	(8/0/0)	1484	28.1	0.7412
17	yeast-1-2-8-9_vs_7	**8**	(8/0/0)	947	30.57	0.3660
18	yeast5	**8**	(8/0/0)	1484	32.73	4.1980
19	ecoli-0-1-3-7_vs_2-6	**7**	(7/0/0)	281	39.14	1.9670
20	yeast6	**8**	(8/0/0)	1484	41.4	2.3020
21	abalone19	**8**	(7/0/1)	4174	129.44	0.5295

In this study, the author compares the two proposed algorithms (i.e., **IBDPPCP** and **IBMCCA**) against the state-of-the-art algorithms, conventional machine learning algorithms, ensemble machine learning methods, and evolutionary computation algorithms in order to assess the performance of the suggested algorithm. Except for the proposed algorithm, which is built in Java using the *JMetal* library [36], all of these algorithms are written in the Python programming language. En-

Table 3.2: Imbalance ratio lower than 9

No	Name	#Attributes (R/I/N)		#Examples	IR	FD
	Imbalance ratio between 1.5 and 9					
1	glass1	**9**	(9/0/0)	214	1.82	0.1897
2	ecoli-0_vs_1	**7**	(7/0/0)	220	1.86	9.7520
3	wisconsin	**9**	(0/9/0)	683	1.86	3.568
4	pima	**8**	(8/0/0)	768	1.87	0.5760
5	iris0	**4**	(4/0/0)	150	2	16.8200
6	glass0	**9**	(9/0/0)	214	2.06	0.6492
7	yeast1	**8**	(8/0/0)	1484	2.46	0.2422
8	haberman	**3**	(0/3/0)	306	2.78	0.1850
9	vehicle2	**18**	(0/18/0)	846	2.88	0.1691
10	vehicle1	**18**	(0/18/0)	846	2.9	0.3805
11	vehicle3	**18**	(0/18/0)	846	2.99	0.1855
12	glass-0-1-2-3_vs_4-5-6	**9**	(9/0/0)	214	3.2	3.3240
13	vehicle0	**18**	(0/18/0)	846	3.25	1.1240
14	ecoli1	**7**	(7/0/0)	336	3.36	2.6500
15	new-thyroid1	**5**	(4/1/0)	215	5.14	3.5790
16	new-thyroid2	**5**	(4/1/0)	215	5.14	3.5790
17	ecoli2	**7**	(7/0/0)	336	5.46	2.6500
18	segment0	**19**	(19/0/0)	2308	6.02	1.7980
19	glass6	**9**	(9/0/0)	214	6.38	2.3910
20	yeast3	**8**	(8/0/0)	1484	8.1	2.7510
21	ecoli3	**7**	(7/0/0)	336	8.6	1.5790

semble learning algorithms are found in the *imblearn* library, whereas machine learning methods are extracted from the *sklearn* library. In contrast, algorithms for evolutionary computation are drawn from the *DEAP* library [45]. The author uses default settings for machine learning methods. Whereas with evolutionary computation algorithms, the author sets up parameters like population size, number of evolutionary generations, etc. to be similar to the proposed method.

3.4.3. Test scenarios

To evaluate the performance of the two proposed algorithms, in this study, the author conducts some experiments as follows:

+Scenario 1: Compare the proposed algorithm with the baseline algorithms

In this part of the study, the two proposed algorithms are compared with five other algorithms.

+ **IBDPPCP2** is another version of IBDPPCP; the difference is that IBDPPCP2 does not use data sampling. The purpose of this comparison is to check the effectiveness of using the data sampling method as well as answer the question: *Is the IBDPPCP capable of solving the problem of imbalanced data if the sampling method is not used?.*

+ **IBDPP2** is another version of the IBDPPCP algorithm. The difference is that IBDPP2 uses DPP2 (Algorithm.5) instead of DPPCP in the co-evolutionary process. In chapter 2, DPPCP and DPP2 are two algorithms proposed for balancing diversity and convergence in multi-objective optimization problems. The purpose of this comparison is to check *how efficient the two algorithms are then applied to the imbalanced data classification problem?.*

+ **EFIS_MOEA** [42] is the premise algorithm chosen by the author for improvement. This is one of the latest studies using the method of solving classification problems with imbalanced data in a similar way to this study. Comparing the two proposed algorithms to determine *whether the proposed algorithms are superior to the premise research?.*

+ **DEMOA** [C6] is an algorithm that uses a decomposition mechanism for classification problems with unbalanced data like IBDPPCP. Their difference is that DEMOA uses only one population and utilizes MOEA/D as a decomposition algorithm. This comparison helps determine *whether a dual population method is better than a single population method?.*

+ **SMEN_C45** is an algorithm that uses only the data sampling method (i.e., SMOTE-ENN) to generate the balanced data without using the co-evolutionary process or ensemble learning. This comparison shows *the effect of using co-evolution in combination with ensemble learning.*

+ Scenario 2: Compare the proposed algorithms with machine learning algorithms

There are five traditional algorithms (SVM, ANN, KNN, Naive Bayes, and LDA) and a deep learning algorithm (CNN). This case study compares the performance of the proposed method against some of the most widely used machine learning algorithms, including conventional and deep learning methods.

+Scenario 3: Compare the proposed algorithm with the ensemble learning algorithms.

In the proposed algorithms, the author uses an ensemble learning method. The main difference between the proposed methods and common ensemble learning algorithms is the way subsets are generated. In the proposed algorithms, subsets are created from individuals found through the co-evolutionary process; in the ensemble learning algorithm, the subsets are created using sampling with replacement mechanisms. This comparison illustrates *how useful it is to use co-evolutionary solutions to find subsets or decision trees?*. There are two subgroups: *Basic Bagging*, *Basic AdaBoost*, and *Basic Randomforest* are three examples of traditional ensemble learning algorithms, and there are three other versions that have been enhanced with sampling methods to handle imbalanced data (i.e. *BalancedBagging*, *Balanced Randomforest* and *RUS AdaBoost*).

+Scenario 4: Compare the proposed algorithm with the evolutionary computational algorithms. It includes three single-objective algorithms (GA, DE, and PSO) and a multi-objective algorithm (NSGA-II).

The idea of this scenario is to compare how the performance of the proposed solution using the co-evolutionary solution compares with other CI algorithms. At the same time, since IBMCCA uses two populations using the same mechanism as NSGA-II, comparing them with each other can clearly show *the effect of the solution using multi-objective co-*

evolutionary methods.

3.4.4. Results and analysis

Scenario 1: Compare the proposed algorithm with the baseline algorithms

The experimental results are shown in Tables 3.7 and 3.8. The questions posed in the previous section will be analyzed in detail in this section.

+ Question 1: Is IBDPPCP algorithm capable of solving the problem of imbalanced data if the sampling method is not used?

Through the comparison of the two algorithms, IBDPPCP and IBDPPCP2, this answer will be clarified. Table 3.7 and Figure 3.5 show the comparison results between the two algorithms on the dataset with a low IR imbalance rate (less than 9). Better values are highlighted. It is easy to see that these two algorithms give better results than the rest. While IBDPPCP gave the best results in 16 of 21 datasets, IBDPPCP2 gave the best results in 5 of 21 datasets. However, looking at the average AUC results, it can be seen that the difference between these two algorithms is not significant. While IBDPPCP results in an average AUC of 0.9038, the value for IBDPPCP2 is 0.8999. The results from the second dataset in Table 3.8 and Figure 3.6 with a greater imbalance rate ($IR > 9$) demonstrate that IBDPPCP produces the best outcomes on 12 out of 21 data sets. Although IDPPCP2 produces the greatest results on nine of the 21 datasets, these two algorithms still give the best results. The average AUC result of IBDPPCP is better than that of IBDPPCP2, the corresponding values for the two algorithms are 0.8520 and 0.8365. *In summary, it can be said that the data sampling method enhances the outcomes. However, this difference is not significant. Even without data sampling, the IDPPCP algorithm is capable of handling imbalanced data.*

+Statistical test for comparing performance

To gain more insight into the reliability of comparing the proposed algorithm with other algorithms, the author conducted a comparative evaluation based on statistics. Two statistical methods used are Friedman and Wilcoxon tests. Both of them are non-parametric statistical tests. While the Friedman test is a rank-based test that determines whether there are any differences between the algorithms being compared, the Wilcoxon test is used to compare two algorithms. It tests the null hypothesis that the population median of the differences between paired algorithms is zero.

Table 3.4 shows the results of the Friedman statistical test. For each dataset, the Friedman test returns a chi-square statistic and a p-value. The p-value is calculated based on the chi-square statistic and the degrees of freedom. The significance level is 0.05. The null hypothesis (i.e., *Ho*) assumes that there is no significant difference between the rankings of the algorithms being compared. As can be seen, the p-value is less than the significance level (3.17150e-08 and 3.41412e-12), so we can reject *Ho* and conclude that there is evidence of a significant difference among the algorithms.

Next, the author uses the Wilcoxon signed-rank test as a post-hoc test to determine which algorithms differ significantly from each other. The results of this statistical test are presented in Tables 3.5 and 3.6 for the two datasets, respectively. Each value in the tables represents the p-value of the comparison between the proposed algorithm and the corresponding algorithm. It is easy to see that most of the p-values are smaller than the significance level (i.e., 0.05). This indicates that there is strong evidence against the null hypothesis and that the two algorithms are likely to be different.

+ *Question 2: How efficient the two algorithms (IDPPCP and IDPP2) are when applied to the imbalanced data classification problem?*

Table 3.4: The Friedman test results for IBDPPCP and the state- of-the-art algorithms on two datasets , *Chi2* is the Chi-square value

Dataset	**Chi2**	**P-value**
Higher9	45.84983	3.17150e-08
Lower9	65.49744	3.41412e-12

Table 3.5: Wilcoxon test at a 0.05 significance level between the proposed algorithm and the state-of-the-art algorithms on a dataset having an imbalance ratio lower than 9

Data	**IBDPP2**	**IBMCCA**	**EFIS_MOEA**	**DEMOA**	**SMEN_C45**
glass0123vs456	1.69544E-09	**0.20377**	**0.28154**	**0.29925**	1.33281E-09
yeast3	1.33281E-09	5.52429E-09	1.50351E-09	5.52429E-09	1.33281E-09
vehicle3	1.33281E-09	1.69544E-09	1.01408E-05	1.01408E-05	1.33281E-09
haberman	**0.22525**	1.2918E-06	5.25381E-08	2.66071E-05	1.33281E-09
segment0	2.4258E-09	8.09995E-08	1.50351E-09	1.33281E-09	1.33281E-09
new-thyroid2	1.33281E-09	1.50351E-09	1.33281E-09	1.33281E-09	1.33281E-09
ecoli1	7.27308E-08	0.002898167	3.03313E-08	4.37475E-09	1.33281E-09
glass0	1.33281E-09	4.3165E-07	0.00017	0.00027	1.33281E-09
ecoli2	1.33281E-09	2.71454E-08	6.20438E-09	6.52824E-08	1.33281E-09
glass1	1.33281E-09	0.00082	1.88302E-05	0.08956	1.33281E-09
pima	1.33281E-09	2.71454E-08	1.73513E-08	1.38013E-07	1.33281E-09
wisconsin	1.33281E-09	1.33281E-09	1.33281E-09	1.33281E-09	1.33281E-09
ecoli0vs1	1.69544E-09	0.00553	0.02905	0.00490	0.02905
Ecoli3	7.89176E-07	0.00032	6.44733E-06	5.28581E-07	1.33281E-09
iris0	2.73146E-09	0.00023	1.70367E-07	3.17669E-07	1.33281E-09
yeast1	1.33281E-09	1.23563E-08	9.01752E-08	4.3165E-07	1.33281E-09
glass6	3.07449E-09	1.33281E-09	1.91117E-09	1.33281E-09	1.33281E-09
vehicle2	1.33281E-09	1.33281E-09	4.71059E-08	3.89092E-09	1.33281E-09
vehicle1	1.33281E-09	2.15356E-09	8.71554E-07	1.01408E-05	1.33281E-09
vehicle0	1.33281E-09	5.25381E-08	3.69964E-06	3.17669E-07	1.33281E-09
new-thyroid1	1.33281E-09	3.01747E-07	1.50351E-09	1.91117E-09	1.33281E-09

Table 3.7 and Figure 3.7 show the comparison results between the two algorithms on the dataset with a low IR imbalance rate ($IR < 9$). It can be easily seen that in all these 21 experimental data sets, the IBDPPCP algorithm gives better results than the IBDPP2 algorithm. The average AUC results for the two algorithms, IBDPPCP and IBDPP2, are 0.9038 and 0.8672, respectively. There are several datasets where IBDPPCP gives much superior results to IBDPP2, such as Yeast3 (0.9548 vs. 0.8827), Glass0123vs456 (0.9551 vs. 0.9098). Table 3.8 and Figure 3.8 show the comparison results between the two algorithms on the dataset with a high IR imbalance rate ($IR > 9$). The results are similar to the first data set. On all 21 experimental datasets, the IBDPPCP algorithm

Table 3.6: Wilcoxon test at a 0.05 significance level between the proposed algorithm and the state-of-the-art algorithms on a dataset having an imbalance ratio higher than 9

Data	**IBDPP2**	**IBMCCA**	**EFIS_MOEA**	**DEMOA**	**SMEN_C45**
yeast4	2.66071E-05	4.91693E-09	1.33281E-09	1.73513E-08	1.33281E-09
yeast1289vs7	1.32497E-05	0.001726994	0.000274727	5.22129E-05	1.33281E-09
shuttlec0vsc4	6.15877E-06	**1**	**0.99226**	**0.08247**	1.33281E-09
ecoli0137vs26	3.73798E-05	2.42853E-08	4.1024E-07	1.33281E-09	1.33281E-09
yeast5	1.33281E-09	1.33281E-09	8.7702E-09	6.52824E-08	1.33281E-09
yeast1vs7	0.00022	0.00448	0.00015	0.00077	1.33281E-09
glass5	1.56928E-06	1.32497E-05	0.00025	0.00076	1.33281E-09
vowel0	1.33281E-09	1.55004E-08	4.37475E-09	2.15356E-09	1.33281E-09
yeast6	9.6218E-07	1.69544E-09	1.33281E-09	1.69544E-09	1.33281E-09
yeast2vs4	6.20438E-09	6.46337E-07	2.3302E-07	0.01198	1.33281E-09
glass4	2.71454E-08	1.50351E-09	1.33281E-09	1.33281E-09	1.33281E-09
yeast1458vs7	**0.09713**	3.17669E-07	8.7702E-09	2.30576E-06	1.33281E-09
glass016vs5	3.73798E-05	0.00031	4.54136E-07	7.2514E-05	1.33281E-09
glass016vs2	2.78884E-06	3.02451E-05	0.06251	**0.27296456**	1.33281E-09
abalone918	7.06371E-06	0.00050	1.00354E-07	1.24151E-07	2.42853E-08
abalone19	0.03362	**0.05353**	0.01337	**0.053535**	1.33281E-09
glass2	0.00010	0.00421	**0.12295**	0.00019	1.33281E-09
yeast2vs8	2.09998E-07	**0.105202641**	**0.45506**	**0.16535**	1.33281E-09
yeast05679vs4	1.72868E-06	1.55004E-08	6.96565E-09	1.38013E-07	1.33281E-09
ecoli4	4.71059E-08	1.33281E-09	1.33281E-09	1.33281E-09	1.33281E-09
shuttlec2vsc4	0.00010	**1**	**1**	**0.22525**	**1**

outperformed IBDPP2. The average AUC results for the two algorithms are 0.8520 and 0.8136, respectively. IBDPPCP outperforms IBDPP2 in some datasets, such as Ecoli4 (0.9534 vs. 0.8845) and Yeast2vs.8 (0.8412 vs. 0.7488). Through the above analysis, it can be concluded that *both algorithms demonstrate the ability to solve the problem. In which the IBDPPCP algorithm (using the DPPCP algorithm) proved to be more efficient than the IBDPP2 algorithm (using the DPP2 algorithm).*

+ Question 3: Are the proposed algorithms better than the premise research?

The results of the comparison between the three methods on the dataset with a low IR imbalance rate ($IR < 9$) are shown in Table 3.7 and Figure 3.9. It is clear that the IBDPPCP algorithm produces the best results in all 21 experimental data sets. The IBMCCA algorithm gives the second-best result. The average AUC and rank results for the three algorithms, IBDPPCP, IBMCCA, and EFIS_MOEA, are 0.9038

(1.29), 0.8908 (3.81), and 0.8890 (4.43), respectively. On datasets with an IR greater than 9, the same outcomes were obtained (Table 3.8 and Figure 3.10). The figures for IBDPPCP, IBMCCA, and EFIS_MOEA are 0.8520 (1.43), 0.8263 (3.76), and 0.8223 (4.38), respectively. Therefore, it can be said that *the two proposed algorithms are better than the baseline method in terms of performance. This shows that these two algorithms are capable of handling classification problems with imbalanced data.*

+ Question 4: Is the dual population method better than the single population method?

The comparison of the two algorithms on the dataset with a low IR imbalance rate is presented in Table 3.7 and Figure 3.11 and the results on the second dataset are shown in Table 3.8 and Figure 3.12. It is easy to see that the IBDPPCP algorithm beats the DEMOA algorithm on all 42 experimental datasets. The figures for the two algorithms on the two datasets are (0.9038 vs. 0.8906) and (0.8520 vs. 0.8253). This demonstrates that *the dual population co-evolution method is superior to the single population method for finding the optimal sets of individuals or subsets of data.*

+ Question 5: How is the effect of using co-evolution in combination with ensemble learning? This issue is clarified by the comparison of the two proposed algorithms with the SMEN_C45 algorithm. Table 3.7 and Figure 3.13 show the comparison results between the three algorithms on the dataset with a low IR imbalance rate ($IR < 9$). It can be easily seen that in all these 21 experimental data sets, the IBDPPCP algorithm gives the best results, and the IBMCCA gives better results than the SMEN_C45 algorithm on 19 of the 21 datasets except newthyroid1 (0.9727 vs. 0.9794) and iris0 (0.9869 vs. 0.9887). Two algorithms produce the same results on Yeast1458vs7 (0.9832). The average

AUC and rank results for the three algorithms IBDPPCP, IBDPP2, and SMEN_C45 are 0.9038 (1.29), 0.8914 (3.33), and 0.8695 (5.48), respectively. The results on the second dataset are similar. IBDPPCP gives the best results on all data. IBMCCA beats SMEN_C45 in 18/21 data. There are five data sets where SMEN_C45 gives better results: shuttlec0vsc4 (0.9989 vs. 0.9997), ecoli4 (0.9015 vs. 0.9084), ecoli0137vs26 (0.8211 vs. 0.8345), yeast1vs7 (0.7000 vs. 0.7572) and yeast1458vs7 (0.5566 vs. 0.5624). The average AUC and rank results for the three algorithms IBDPPCP, IBDPP2, and SMEN_C45 are 0.8520 (1.43), 0.8263 (3.76), and 0.8035 (5.48), respectively. These results show that *the utilization of co-evolution combined with ensemble learning is effective in improving classification results.*

Scenario 2: Compare the proposed algorithm with machine learning algorithms

There are five traditional algorithms (SVM, ANN, KNN, Naive Bayes, and LDA) and a deep learning algorithm (CNN). Experimental results with each dataset are shown in tables 3.9, 3.10 and Figures 3.15, 3.16. It can be easily seen that the two proposed algorithms give better results in most of the test cases, except for the case of datasets with IR 9 ¿ 9, where CNN gives a better result than IBMCCA. In the data set having $IR < 9$, *IBDPPCP* and *IBMCCA* give superior results compared to other algorithms. Meanwhile, the *ANN* algorithm gives the worst results with 0.7081. The *CNN* algorithm gives the third-best result with an AUC value of 0.7982. *SVM, KNN, and LDA* algorithms give results that are not too different; the corresponding figures are 0.7504, 0.7753, and 0.7477.

Switching to the more difficult datasets (with $IR > 9$). The results of most algorithms are reduced, except for the cases of *CNN* and *Naïve Bayers* where the results are even better than the previous data set. The

Table 3.7: Experimental results of the proposed algorithm and the baseline algorithms with IR less than 9. The values are presented in the form of mean ± standard deviation (rank)

Data	IBDPPCP	IBDPPCP2	IBDPP2	IBMCCA	EFIS_MOEA	DEMOA	SMEN_C45
yeast3	0.9548 ± 0.0022 (1)	0.9515 ± 0.0011 (2)	0.8827 ± 0.0317 (7)	0.9452 ± 0.0049 (3)	0.9395 ± 0.0055 (5)	0.9446 ± 0.0048 (4)	0.9326 ± 0.0000 (6)
yeast1	0.7543 ± 0.0039 (2)	0.7564 ± 0.0036 (1)	0.7096 ± 0.0126 (7)	0.7397 ± 0.0068 (5)	0.7439 ± 0.0055 (4)	0.7457 ± 0.0051 (3)	0.7138 ± 0.0000 (6)
wisconsin	0.9823 ± 0.0013 (1)	0.9815 ± 0.0017 (2)	0.9649 ± 0.0040 (6)	0.9709 ± 0.0032 (3)	0.9700 ± 0.0035 (4)	0.9699 ± 0.0022 (5)	0.9596 ± 0.0000 (7)
vehicle3	0.8205 ± 0.0049 (1)	0.8041 ± 0.0054 (4)	0.7580 ± 0.0124 (6)	0.7984 ± 0.0089 (5)	0.8095 ± 0.0079 (3)	0.8110 ± 0.0070 (2)	0.7452 ± 0.0000 (7)
vehicle2	0.9815 ± 0.0024 (1)	0.9801 ± 0.0012 (2)	0.9335 ± 0.0162 (7)	0.9689 ± 0.0075 (5)	0.9719 ± 0.0051 (3)	0.9712 ± 0.0052 (4)	0.9434 ± 0.0000 (6)
vehicle1	0.8054 ± 0.0036 (1)	0.7863 ± 0.0055 (4)	0.7581 ± 0.0147 (7)	0.7842 ± 0.0080 (5)	0.7969 ± 0.0056 (3)	0.7978 ± 0.0056 (2)	0.7591 ± 0.0000 (6)
vehicle0	0.9678 ± 0.0025 (2)	0.9687 ± 0.0022 (1)	0.9328 ± 0.0129 (6)	0.9597 ± 0.9597 (5)	0.9607 ± 0.0048 (4)	0.9619 ± 0.0037 (3)	0.9117 ± 0.0000 (7)
segment0	0.9958 ± 0.0007 (1)	0.9918 ± 0.0007 (5)	0.9893 ± 0.0020 (7)	0.9934 ± 0.0013 (2)	0.9906 ± 0.0017 (6)	0.9925 ± 0.0014 (3)	0.9920 ± 0.0000 (4)
pima	0.7889 ± 0.0037 (2)	0.7892 ± 0.0042 (1)	0.7562 ± 0.0126 (6)	0.7753 ± 0.0064 (4)	0.7746 ± 0.0070 (5)	0.7773 ± 0.0074 (3)	0.7376 ± 0.0000 (7)
newthyroid2	0.9885 ± 0.0031 (1)	0.9679 ± 0.0071 (4)	0.9511 ± 0.0144 (7)	0.9727 ± 0.0063 (3)	0.9564 ± 0.0107 (6)	0.9635 ± 0.0094 (5)	0.9794 ± 0.0000 (2)
newthyroid1	0.9953 ± 0.0034 (1)	0.9834 ± 0.0057 (4)	0.9613 ± 0.0094 (7)	0.9869 ± 0.0049 (3)	0.9741 ± 0.0128 (6)	0.9771 ± 0.0109 (5)	0.9887 ± 0.0000 (2)
iris0	0.9976 ± 0.0025 (1)	0.9965 ± 0.0024 (2)	0.9795 ± 0.0123 (7)	0.9908 ± 0.0066 (3)	0.9806 ± 0.0089 (6)	0.9842 ± 0.0101 (5)	0.9900 ± 0.0000 (4)
haberman	0.6394 ± 0.0112 (2)	0.6709 ± 0.0083 (1)	0.6324 ± 0.0237 (3)	0.6142 ± 0.0156 (5)	0.6131 ± 0.011 (6)	0.6219 ± 0.0131 (4)	0.5872 ± 0.0000 (7)
glass6	0.9500 ± 0.0048 (1)	0.9498 ± 0.0030 (2)	0.9148 ± 0.0163 (5)	0.9191 ± 0.0128 (3)	0.9085 ± 0.0213 (6)	0.8973 ± 0.0246 (7)	0.916 ± 0.0000 (4)
glass1	0.7966 ± 0.0105 (1)	0.7839 ± 0.0131 (3)	0.7183 ± 0.0265 (6)	0.7817 ± 0.0177 (4)	0.7810 ± 0.0112 (5)	0.7919 ± 0.0124 (2)	0.6987 ± 0.0000 (7)
glass0	0.8622 ± 0.0092 (2)	0.8644 ± 0.0067 (1)	0.8067 ± 0.0218 (7)	0.8391 ± 0.014 (5)	0.8471 ± 0.0159 (3)	0.8471 ± 0.0147 (3)	0.8140 ± 0.0000 (6)
glass0123vs4565	0.9551 ± 0.0063 (1)	0.9468 ± 0.0059 (5)	0.9098 ± 0.0202 (7)	0.9522 ± 0.0097 (4)	0.9529 ± 0.0071 (2)	0.9527 ± 0.0083 (3)	0.9184 ± 0.0000 (6)
ecoli3	0.9102 ± 0.0061 (1)	0.9013 ± 0.0047 (2)	0.8856 ± 0.0186 (7)	0.8996 ± 0.0107 (3)	0.8968 ± 0.0106 (4)	0.8905 ± 0.0144 (5)	0.8861 ± 0.0000 (6)
ecoli2	0.9249 ± 0.0052 (1)	0.9181 ± 0.0046 (2)	0.8932 ± 0.0117 (7)	0.9103 ± 0.0072 (5)	0.9104 ± 0.0067 (4)	0.9116 ± 0.0071 (3)	0.9000 ± 0.0000 (6)
ecoli1	0.9246 ± 0.0024 (1)	0.9216 ± 0.004 (3)	0.9016 ± 0.0224 (7)	0.9219 ± 0.0046 (2)	0.9071 ± 0.0096 (5)	0.9090 ± 0.0079 (4)	0.9034 ± 0.0000 (6)
ecoli0vs15	0.9841 ± 0.0012 (2)	0.9844 ± 0.0010 (1)	0.9720 ± 0.0090 (7)	0.9832 ± 0.001 (3)	0.9832 ± 0.0000 (3)	0.983 ± 0.0005 (6)	0.9832 ± 0.0000 (3)
AVERAGE	0.9038 (1.29)	0.8999 (2.48)	0.8672 (6.48)	0.8908 (3.81)	0.889 (4.43)	0.8906 (3.86)	0.8695 (5.48)

Table 3.8: Experimental results of the proposed algorithm and the baseline algorithms with IR higher than 9. The values are presented in the form of mean ± standard deviation (rank)

Dataset	IBDPPCP	IBDPPCP2	IBDPP2	IBMCCA	EFIS_MOEA	DEMOA	SMEN_C45
vowel0	**0.981 ± 0.0023 (1)**	**0.9688 ± 0.0023 (3)**	**0.9468 ± 0.0123 (6)**	**0.97 ± 0.0057 (2)**	**0.9645 ± 0.0083 (4)**	**0.9622 ± 0.0072 (5)**	**0.9461 ± 0.0000 (7)**
shuttlec2vsc4	**1.0000 ± 0.0000 (1)**	**1.0000 ± 0.0000 (1)**	0.995 ± 0.0045 (7)	**1.0000 ± 0.0000 (1)**	**1.0000 ± 0.0000 (1)**	0.9996 ± 0.0008 (6)	**1.0000 ± 0.0000 (1)**
shuttlec0vsc4	0.9999 ± 0.0001 (2)	**1.0000 ± 0.0000 (1)**	0.9989 ± 0.0019 (6)	0.9989 ± 0.0001 (6)	0.9998 ± 0.0003 (3)	0.9998 ± 0.0002 (3)	0.9997 ± 0.0000 (5)
glass5	0.911 ± 0.0115 (2)	**0.9431 ± 0.0087 (1)**	0.8591 ± 0.0374 (6)	0.8907 ± 0.0248 (3)	0.8807 ± 0.0312 (5)	0.8893 ± 0.0288 (4)	0.8195 ± 0.0000 (7)
glass4	**0.9483 ± 0.0099 (1)**	**0.8818 ± 0.0214 (4)**	0.8917 ± 0.0317 (2)	0.8857 ± 0.0181 (3)	0.8794 ± 0.0188 (5)	0.866 ± 0.0283 (6)	0.8646 ± 0.0000 (7)
glass2	**0.7921 ± 0.0207 (1)**	**0.6348 ± 0.0255 (7)**	0.7382 ± 0.0542 (6)	0.7646 ± 0.0361 (3)	0.7754 ± 0.0384 (2)	0.7595 ± 0.0366 (4)	0.7452 ± 0.0000 (5)
glass016vs5	**0.938 ± 0.028 (1)**	**0.934 ± 0.0145 (2)**	0.8905 ± 0.0433 (4)	0.8978 ± 0.0426 (3)	0.8817 ± 0.0313 (6)	0.8895 ± 0.0399 (5)	0.8657 ± 0.0000 (7)
glass016vs2	**0.7515 ± 0.029 (1)**	**0.597 ± 0.034 (7)**	0.6897 ± 0.0421 (5)	0.7068 ± 0.0363 (4)	0.738 ± 0.0306 (3)	0.7423 ± 0.0312 (2)	0.6575 ± 0.0000 (6)
ecoli4	**0.9534 ± 0.0056 (1)**	**0.927 ± 0.0028 (2)**	0.8845 ± 0.0335 (5)	0.9015 ± 0.0209 (4)	0.8727 ± 0.019 (7)	0.8779 ± 0.0214 (6)	0.9084 ± 0.0000 (3)
ecoli0137vs26	0.857 ± 0.0106 (2)	0.8886 ± 0.0067 (1)	0.8118 ± 0.0416 (7)	0.8211 ± 0.0162 (6)	0.8335 ± 0.0148 (4)	0.8312 ± 0.007 (5)	0.8345 ± 0.0000 (3)
abalone918	0.7488 ± 0.0107 (2)	**0.7752 ± 0.0073 (1)**	0.7105 ± 0.0374 (7)	0.7321 ± 0.0206 (3)	0.7186 ± 0.0143 (6)	0.7214 ± 0.015 (5)	0.7295 ± 0.0000 (4)
abalone19	0.6247 ± 0.0213 (2)	**0.7005 ± 0.009 (1)**	0.6183 ± 0.0054 (3)	0.6134 ± 0.0231 (5)	0.6147 ± 0.0166 (4)	0.6127 ± 0.0245 (6)	0.5159 ± 0.0000 (7)
yeast6	0.8894 ± 0.0045 (2)	**0.9013 ± 0.0036 (1)**	0.8623 ± 0.0209 (3)	0.8623 ± 0.0128 (3)	0.8091 ± 0.0187 (7)	0.8385 ± 0.0177 (5)	0.8148 ± 0.0000 (6)
yeast5	0.9745 ± 0.0029 (1)	0.9728 ± 0.0009 (2)	0.9547 ± 0.0178 (7)	0.9613 ± 0.0045 (5)	0.9627 ± 0.0074 (4)	0.9644 ± 0.006 (3)	0.956 ± 0.0000 (6)
yeast4	0.8414 ± 0.0075 (2)	**0.87 ± 0.0028 (1)**	0.8181 ± 0.0226 (3)	0.8064 ± 0.0138 (5)	0.7988 ± 0.0119 (6)	0.8176 ± 0.014 (4)	0.7636 ± 0.0000 (7)
yeast2vs8	**0.8412 ± 0.0125 (1)**	0.8066 ± 0.0153 (5)	0.7488 ± 0.0604 (7)	0.8334 ± 0.0174 (4)	0.839 ± 0.0181 (2)	0.8365 ± 0.0145 (3)	0.804 ± 0.0000 (6)
yeast2vs4	0.9392 ± 0.007 (2)	**0.9561 ± 0.0017 (1)**	0.9046 ± 0.0207 (6)	0.9225 ± 0.0131 (4)	0.9178 ± 0.0126 (5)	0.9301 ± 0.0125 (3)	0.9012 ± 0.0000 (7)
yeast1vs7	0.7258 ± 0.0154 (2)	0.7246 ± 0.0147 (3)	0.6855 ± 0.0399 (7)	0.7 ± 0.0379 (4)	0.6959 ± 0.029 (6)	0.6994 ± 0.0282 (5)	0.7572 ± 0.0000 (1)
yeast1458vs7	0.6054 ± 0.0116 (1)	0.5994 ± 0.0184 (2)	0.5923 ± 0.041 (3)	0.5566 ± 0.0288 (7)	0.5638 ± 0.0205 (5)	0.5751 ± 0.0207 (4)	0.5624 ± 0.0000 (6)
yeast1289vs7	0.7200 ± 0.0161 (1)	0.6635 ± 0.0271 (5)	0.6635 ± 0.0466 (5)	0.6992 ± 0.0247 (2)	0.6986 ± 0.0175 (3)	0.6908 ± 0.0273 (4)	0.6215 ± 0.0000 (7)
yeast05679vs4	0.8488 ± 0.0054 (1)	0.8226 ± 0.0048 (5)	0.8203 ± 0.0191 (6)	0.8275 ± 0.0099 (2)	0.8242 ± 0.0124 (4)	0.8274 ± 0.0109 (3)	0.806 ± 0.0000 (7)
AVERAGE	**0.8520 (1.43)**	0.8365 (2.67)	0.8136 (5.29)	0.8263 (3.76)	0.8223 (4.38)	0.8253 (4.33)	0.8035 (5.48)

Table 3.9: Experimental results of the proposed algorithm and machine learning algorithms on datasets with IR less than 9. The values are presented in the form of mean (rank)

Data	IBDPPCP	IBMCCA	SVM	ANN	KNN	Naive Bayes	LDA	CNN
yeast3	0.9548 (1)	0.9452 (2)	0.8377 (4)	0.6977 (7)	0.8300 (5)	0.6033 (8)	0.8257 (6)	0.9008 (3)
yeast1	0.7543 (1)	0.7397 (2)	0.6185 (6)	0.6013 (7)	0.6452 (4)	0.519 (8)	0.6286 (5)	0.7104 (3)
wisconsin	0.9823 (1)	0.9709 (2)	0.9211 (3)	0.9064 (5)	0.8573 (7)	0.8875 (6)	0.8458 (8)	0.9206 (4)
vehicle3	0.8205 (1)	0.7984 (2)	0.5000 (3)	0.5000 (3)	0.5000 (3)	0.4976 (8)	0.4992 (6)	0.4988 (7)
vehicle2	0.9815 (1)	0.9689 (2)	0.5000 (5)	0.5000 (5)	0.5000 (5)	0.5056 (3)	0.5000 (5)	0.5029 (4)
vehicle1	0.8054 (1)	0.7842 (2)	0.5061 (6)	0.5 (8)	0.5099 (4)	0.5039 (7)	0.5099 (4)	0.5125 (3)
vehicle0	0.9678 (1)	0.9597 (2)	0.5000 (5)	0.5000 (5)	0.5000 (5)	0.5054 (3)	0.5000 (5)	0.5032 (4)
segment0	0.9958 (1)	0.9934 (2)	0.9907 (4)	0.9544 (7)	0.9889 (5)	0.8936 (8)	0.9757 (6)	0.9909 (3)
pima	0.7889 (1)	0.7753 (2)	0.722 (4)	0.6412 (8)	0.6959 (7)	0.7204 (6)	0.7206 (5)	0.7568 (3)
newthyroid2	0.9885 (1)	0.9727 (2)	0.8802 (5)	0.8143 (7)	0.9516 (4)	0.8087 (8)	0.8286 (6)	0.9662 (3)
newthyroid1	0.9953 (1)	0.9869 (2)	0.8687 (5)	0.8229 (8)	0.9659 (4)	0.8258 (7)	0.8429 (6)	0.9667 (3)
iris0	0.9976 (7)	0.9908 (8)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)
haberman	0.6394 (1)	0.6142 (2)	0.5000 (4)	0.5000 (4)	0.4879 (8)	0.5000 (4)	0.5000 (4)	0.5118 (3)
glass6	0.950 (1)	0.9191 (2)	0.8406 (6)	0.8341 (8)	0.8379 (7)	0.8911 (4)	0.9086 (3)	0.8709 (5)
glass1	0.7966 (1)	0.7817 (2)	0.5591 (6)	0.5393 (8)	0.7488 (3)	0.6709 (5)	0.5430 (7)	0.7376 (4)
glass0	0.8622 (1)	0.8391 (2)	0.6085 (8)	0.6281 (7)	0.8027 (3)	0.7001 (5)	0.6980 (6)	0.7652 (4)
glass0123vs4565	0.9551 (1)	0.9522 (2)	0.8934 (5)	0.8676 (8)	0.9086 (4)	0.8689 (7)	0.8773 (6)	0.9392 (3)
ecoli3	0.9102 (1)	0.8996 (2)	0.7833 (6)	0.5508 (8)	0.7086 (7)	0.8436 (4)	0.8355 (5)	0.8742 (3)
ecoli2	0.9249 (2)	0.9103 (3)	0.9049 (4)	0.7630 (7)	0.9358 (1)	0.6074 (8)	0.8235 (6)	0.9020 (5)
ecoli1	0.9246 (1)	0.9219 (2)	0.8443 (6)	0.8046 (7)	0.885 (3)	0.7363 (8)	0.8481 (5)	0.8827 (4)
ecoli0vs15	0.9841 (1)	0.9832 (2)	0.9800 (4)	0.9702 (7)	0.9733 (5)	0.9077 (8)	0.9832 (2)	0.9721 (6)
AVERAGE	0.9038 (1.333)	0.8908 (2.333)	0.7504 (4.762)	0.7081 (6.429)	0.7753 (4.524)	0.7087 (6)	0.7477 (5.095)	0.7982 (3.714)

figures for this algorithm are 0.844 and 0.762, respectively. IBDPPCP still gives the highest result, with an AUC value of 0.8520. IBMCCA gives the 3^{rd} best result with 0.8263. *CNN* now takes second place with 0.8366. The *KNN* also gives a good result with 0.7542. The rest of the algorithms are pretty much the same, except that ANN continues to give the worst results with 0.5832.

Let's look at some data with a small FD in all the data. With *Yeast1458vs7*, the smallest *FD* index is 0.1757. It can be seen in Table 3.10 that the results of the algorithms with this set are relatively low, while *IBDPPCP* and *CNN* continue to give the two best results (respectively, 0.6054 and 0.6148), and the algorithms are still all giving results of approximately 0.5. With *Vehicle2* (FD = 0.1691) witnessed the two proposed algorithms give completely superior results compared to other algorithms. While the IBDPPCP and IBMCCA give 0.9815 and 0.9689, all remaining algorithms give results slightly better than random level (i.e. 0.5).

From the above experimental results, it can be concluded that the proposed algorithms and the CNN algorithm give more stable and better

Table 3.10: Experimental results of the proposed algorithm and machine learning algorithms on datasets with IR higher than 9. The values are presented in the form of mean (rank)

Data	IBDPPCP	IBMCCA	SVM	ANN	KNN	Naive Bayes	LDA	CNN
vowel0	0.9810 (4)	0.9700 (5)	0.9667 (6)	0.8555 (8)	0.9883 (2)	0.9883 (2)	0.8572 (7)	0.9894 (1)
shuttlec2vsc4	1.0000 (1)	1.0000 (1)	0.9500 (3)	0.7987 (8)	0.9500 (3)	0.9500 (3)	0.9418 (7)	0.9500 (3)
shuttlec0vsc4	0.9999 (1)	0.9989 (2)	0.9960 (3)	0.9960 (3)	0.9960 (3)	0.9960 (3)	0.9960 (3)	0.9960 (3)
glass5	0.9110 (2)	0.8907 (5)	0.5000 (7)	0.5000 (7)	0.8927 (3)	0.8927 (3)	0.6427 (6)	0.9126 (1)
glass4	0.9483 (1)	0.8857 (3)	0.5667 (7)	0.5667 (7)	0.7659 (4)	0.7659 (4)	0.5851 (6)	0.9214 (2)
glass2	0.7921 (1)	0.7646 (2)	0.5000 (6)	0.5000 (6)	0.5712 (4)	0.5712 (4)	0.4949 (8)	0.6472 (3)
glass016vs5	0.9380 (1)	0.8978 (3)	0.5000 (7)	0.5000 (7)	0.8386 (4)	0.8386 (4)	0.5914 (6)	0.9267 (2)
glass016vs2	0.7515 (1)	0.7068 (2)	0.5000 (6)	0.5000 (6)	0.5690 (4)	0.569 (4)	0.4943 (8)	0.6113 (3)
ecoli4	0.9534 (1)	0.9015 (4)	0.9000 (5)	0.6333 (8)	0.8484 (6)	0.8484 (6)	0.9187 (2)	0.9018 (3)
ecoli0137vs26	0.8570 (1)	0.8211 (6)	0.8500 (2)	0.5000 (8)	0.8445 (3)	0.8445 (3)	0.8445 (3)	0.8063 (7)
yeast6	0.8894 (1)	0.8623 (3)	0.5000 (7)	0.5000 (7)	0.7520 (4)	0.7520 (4)	0.6955 (6)	0.8764 (2)
yeast5	0.9745 (1)	0.9613 (2)	0.6229 (7)	0.5064 (8)	0.8479 (4)	0.8479 (4)	0.7993 (6)	0.9519 (3)
yeast4	0.8414 (1)	0.8064 (3)	0.5000 (7)	0.5000 (7)	0.5846 (5)	0.5846 (5)	0.6088 (4)	0.8202 (2)
yeast2vs8	0.8412 (1)	0.8334 (2)	0.7739 (4)	0.725 (8)	0.7739 (4)	0.7739 (4)	0.7739 (4)	0.8224 (3)
yeast2vs4	0.9392 (1)	0.9225 (2)	0.7933 (7)	0.5000 (8)	0.8395 (4)	0.8395 (4)	0.8282 (6)	0.8831 (3)
yeast1vs7	0.7258 (2)	0.7000 (3)	0.5000 (7)	0.5000 (7)	0.5574 (5)	0.5574 (5)	0.5608 (4)	0.7557 (1)
yeast1458vs7	0.6054 (2)	0.5566 (3)	0.5000 (4)	0.5000 (4)	0.4947 (7)	0.4947 (7)	0.5000 (4)	0.6148 (1)
yeast1289vs7	0.7200 (1)	0.6992 (3)	0.5000 (7)	0.5000 (7)	0.5317 (5)	0.5317 (5)	0.5328 (4)	0.7101 (2)
yeast05679vs4	0.8488 (1)	0.8275 (2)	0.5600 (7)	0.5000 (8)	0.6829 (5)	0.6829 (5)	0.7309 (4)	0.7973 (3)
AVERAGE	0.852 (1.32)	0.8263 (2.95)	0.6568 (5.74)	0.5832 (6.95)	0.7542 (4.16)	0.7542 (4.16)	0.7051 (5.16)	0.8366 (2.53)

classification results than other algorithms on the datasets. However, in terms of the overlapping factor, the proposed algorithms give superior results to all. Thereby proving that *the proposed algorithms are capable of handling the overlapping phenomenon of imbalanced data.* **Scenario 3: Compare the proposed algorithm with ensemble learning algorithms**

The detailed experimental results are shown in tables 3.11, 3.12 and Figures 3.17, 3.18. There are two different groups. The first one includes traditional ensemble algorithms such as bagging, boosting, and random forest. The second one includes three algorithms that combine the traditional ensemble algorithms with sampling techniques: *BalancedRandomForest* is a random forest algorithm that applies random undersampling to balance the different bootstraps. *RUSAdaboost* is an AdaBoost classifier where each bootstrap is balanced using random undersampling at each round of boosting. *Balancedbagging* is a bagging classifier applying random-under sampling to balance.

Observing the results in Figure 3.17, it is clear that the algorithms using the sampling solution give better results than the traditional algorithms. The difference is significant. The figures for bagging, Random-

Table 3.11: Experimental results of the proposed algorithm and ensemble learning algorithms on datasets with an IR lower than 9. The values are presented in the form of mean (rank)

Data	IBDPPCP	IBMCCA	Basic Bagging	BalancedBagging	Basic RF	Balanced RF	Basic AdaBoost	RUS AdaBoost
yeast3	0.9548 (1)	0.9452 (2)	0.8396 (7)	0.9198 (5)	0.7983 (8)	0.9292 (3)	0.8603 (6)	0.9227 (4)
yeast1	0.7543 (1)	0.7397 (2)	0.6622 (7)	0.7147 (4)	0.6525 (8)	0.7107 (5)	0.6803 (6)	0.7317 (3)
wisconsin	0.9823 (1)	0.9709 (2)	0.9106 (8)	0.9168 (6)	0.9127 (7)	0.9211 (3)	0.9211 (3)	0.9211 (3)
vehicle3	0.8205 (1)	0.7984 (2)	0.4992 (3)	0.4961 (8)	0.4992 (3)	0.4977 (7)	0.4992 (3)	0.4987 (6)
vehicle2	0.9815 (1)	0.9689 (2)	0.5000 (6)	0.5032 (4)	0.5000 (6)	0.5032 (4)	0.5000 (6)	0.5051 (3)
vehicle1	0.8054 (1)	0.7842 (2)	0.5076 (6)	0.5099 (3)	0.5076 (6)	0.5099 (3)	0.5053 (8)	0.5099 (3)
vehicle0	0.9678 (1)	0.9597 (2)	0.5000 (6)	0.5054 (3)	0.5000 (6)	0.5039 (5)	0.5000 (6)	0.5049 (4)
segment0	0.9958 (1)	0.9934 (3)	0.9833 (8)	0.9919 (6)	0.9921 (5)	0.9927 (4)	0.9911 (7)	0.9935 (2)
pima	0.7889 (1)	0.7753 (2)	0.6885 (7)	0.7321 (4)	0.6877 (8)	0.7246 (5)	0.7115 (6)	0.7479 (3)
newthyroid2	0.9885 (1)	0.9727 (2)	0.9401 (7)	0.9575 (5)	0.9401 (7)	0.9631 (4)	0.9544 (6)	0.9635 (3)
newthyroid1	0.9953 (1)	0.9869 (2)	0.9032 (8)	0.9548 (6)	0.9202 (7)	0.9718 (5)	0.9774 (4)	0.9775 (3)
iris0	0.9976 (7)	0.9908 (8)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)
haberman	0.6394 (1)	0.6142 (2)	0.5000 (5)	0.5094 (3)	0.5000 (5)	0.4831 (8)	0.5000 (5)	0.5078 (4)
glass6	0.9500 (1)	0.9191 (6)	0.8692 (8)	0.9311 (2)	0.8892 (7)	0.9225 (4)	0.9252 (3)	0.9224 (5)
glass1	0.7966 (1)	0.7817 (5)	0.7396 (6)	0.7874 (2)	0.7819 (4)	0.785 (3)	0.6916 (8)	0.7343 (7)
glass0	0.8622 (1)	0.8391 (2)	0.7979 (6)	0.8060 (5)	0.7797 (7)	0.8244 (3)	0.7594 (8)	0.8190 (4)
glass0123vs4565	0.9551 (1)	0.9522 (2)	0.8717 (7)	0.9033 (5)	0.8793 (6)	0.937 (4)	0.8566 (8)	0.9441 (3)
ecoli3	0.9102 (1)	0.8996 (2)	0.712 (7)	0.8628 (5)	0.7169 (6)	0.8748 (3)	0.6785 (8)	0.8693 (4)
ecoli2	0.9249 (1)	0.9103 (3)	0.8605 (7)	0.9120 (2)	0.8729 (6)	0.8967 (4)	0.8461 (8)	0.8950 (5)
ecoli1	0.9246 (1)	0.9219 (2)	0.8457 (8)	0.9095 (3)	0.8633 (7)	0.8873 (4)	0.8787 (5)	0.8752 (6)
ecoli0vs15	0.9841 (1)	0.9832 (2)	0.9795 (5)	0.9796 (4)	0.9831 (3)	0.9657 (8)	0.9766 (6)	0.9686 (7)
AVERAGE	0.9038 (1.286)	0.8908 (2.714)	0.7726 (6.333)	0.8063 (4.095)	0.7759 (5.857)	0.8064 (4.286)	0.7739 (5.762)	0.8051 (3.952)

forest, and Adaboost are (0.77263 vs. 0.8063), (0.7759 vs. 0.8064), and (0.7739 vs. 0.8051), respectively. Looking at this figure also shows the complete superiority of the proposed algorithms over the rest of the algorithms when they reach the values of 0.9038 and 0.8908 for IBDPPCP and IBMCCA, respectively. This is the result of a small dataset with an imbalanced ratio.

Next, observe the results in Table 3.12 and Figure 3.18 to see how the experimental results with a more difficult data set will be. This data set witnessed the fairly evenly good classification ability of the algorithms. While IBDPPCP still gives the best results, the ensemble learning algorithms combined with the sampling method give slightly better results than the IBMCCA algorithm. The *BalanceRF* algorithm reaches a value of 0.8438, while the figure for IBMCCA is 0.8263. Algorithms that do not use sampling solutions also give relatively good results, reaching approximately 0.74 with the two algorithms *Bagging* and *Adaboost*. The *Randomforest* algorithm gave the worst results with a value of 0.6946.

Through this experimental scenario, the following conclusions can be drawn: Algorithms using sampling solutions give better and more stable results than conventional algorithms. The proposed algorithm still gives

Table 3.12: Experimental results of the proposed algorithm and ensemble learning algorithms on datasets with an IR higher than 9. The values are presented in the form of mean (rank)

Data	IBDPPCP	IBMCCA	Basic Bagging	BalancedBagging	Basic RF	Balanced RF	Basic AdaBoost	RUS AdaBoost
vowel0	0.9810 (1)	0.9700 (2)	0.9589 (5)	0.9488 (6)	0.9389 (7)	0.9666 (4)	0.9033 (8)	0.9668 (3)
shuttlec2vsc4	1.0000 (1)	1.0000 (1)	0.9500 (7)	0.9500 (7)	1.0000 (1)	1.0000 (1)	1.0000 (1)	0.9833 (6)
shuttlec0vsc4	0.9999 (7)	0.9989 (8)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)
glass5	0.9110 (3)	0.8907 (5)	0.7976 (7)	0.8939 (4)	0.6951 (8)	0.9220 (2)	0.8476 (6)	0.9374 (1)
glass4	0.9483 (1)	0.8857 (4)	0.6542 (8)	0.9175 (2)	0.6808 (7)	0.8518 (5)	0.7950 (6)	0.8967 (3)
glass2	0.7921 (1)	0.7646 (2)	0.4847 (8)	0.6460 (5)	0.5205 (7)	0.7094 (3)	0.6155 (6)	0.6889 (4)
glass016vs5	0.9380 (1)	0.8978 (3)	0.8443 (6)	0.8929 (4)	0.6971 (8)	0.8914 (5)	0.8305 (7)	0.9371 (2)
glass016vs2	0.7515 (1)	0.7068 (3)	0.5412 (7)	0.6564 (5)	0.5276 (8)	0.7340 (2)	0.5660 (6)	0.6776 (4)
ecoli4	0.9534 (1)	0.9015 (4)	0.8405 (6)	0.8747 (5)	0.8234 (7)	0.9339 (3)	0.8171 (8)	0.9434 (2)
ecoli0137vs26	0.8570 (3)	0.8211 (4)	0.7391 (5)	0.9252 (1)	0.65 (7)	0.8663 (2)	0.6409 (8)	0.6983 (6)
yeast6	0.8894 (1)	0.8623 (3)	0.7115 (8)	0.8701 (2)	0.7122 (6)	0.8505 (5)	0.7119 (7)	0.8568 (4)
yeast5	0.9745 (1)	0.9613 (2)	0.8483 (6)	0.9597 (3)	0.7573 (8)	0.9563 (4)	0.8368 (7)	0.9500 (5)
yeast4	0.8414 (1)	0.8064 (5)	0.6429 (6)	0.8129 (4)	0.5566 (8)	0.8296 (2)	0.6194 (7)	0.8187 (3)
yeast2vs8	0.8412 (1)	0.8334 (2)	0.7478 (5)	0.7545 (4)	0.6250 (8)	0.7407 (6)	0.6978 (7)	0.7808 (3)
yeast2vs4	0.9392 (2)	0.9225 (4)	0.8328 (7)	0.949 (1)	0.8091 (8)	0.9190 (5)	0.8439 (6)	0.9343 (3)
yeast1vs7	0.7258 (4)	0.7000 (5)	0.6608 (6)	0.7494 (3)	0.5453 (8)	0.76600 (2)	0.5751 (7)	0.7661 (1)
yeast1458vs7	0.6054 (3)	0.5566 (4)	0.5152 (6)	0.5535 (5)	0.5000 (7)	0.6721 (1)	0.4985 (8)	0.6342 (2)
yeast1289vs7	0.7200 (2)	0.6992 (3)	0.6257 (6)	0.7209 (1)	0.5478 (8)	0.6391 (5)	0.5645 (7)	0.6820 (4)
yeast05679vs4	0.8488 (1)	0.8275 (2)	0.7020 (6)	0.7860 (3)	0.6098 (8)	0.7841 (4)	0.6908 (7)	0.7673 (5)
AVERAGE	0.8520 (1.89)	0.8263 (3.47)	0.742 (6.11)	0.8348 (3.47)	0.6946 (6.84)	0.8438 (3.26)	0.7397 (6.32)	0.8379 (3.26)

the best results when compared to these ensemble learning algorithms. This proves that *selecting subsets from the original dataset by using the co-evolutionary methods is better than the sampling with replacement mechanism that is commonly used by ensemble machine learning algorithms.*

Scenario 4: Compare the proposed algorithms with the evolutionary computational algorithms(ECAs)

These algorithms can be divided into two groups: single-objective algorithms (i.e., GA, DE, and PSO) and multi-objective algorithms (i.e., NSGA-II). Detailed experimental results with the data are shown in tables 3.13, 3.14, and figures 3.19, 3.20.

The first data set ($IR < 9$) witnesses a complete superiority of the proposed algorithms compared to the remaining algorithms. While ECAs give fairly uniform results, fluctuating around 0.77, the proposed algorithms reach 0.9038 and 0.8908. Looking at this chart, it can be seen that in the case of the small imbalanced rate data, the single-objective and multi-objective algorithms do not have much difference.

In the second dataset ($IR > 9$), the results are similar to the first dataset. There is still a big difference between the proposed algorithms and the rest. The DE algorithm gives the worst result with a value of

Table 3.13: Experimental results of the proposed algorithm and evolutionary computation learning algorithms on datasets with an IR lower than 9. The values are presented in the form of mean (rank)

Data	IBDPPCP	IBMCCA	GA	DE	PSO	NSGA-II
yeast3	0.9548 (1)	0.9452 (2)	0.8471 (6)	0.8526 (4)	0.8478 (5)	0.8567 (3)
yeast1	0.7543 (1)	0.7397 (2)	0.6340 (5)	0.6279 (6)	0.6451 (4)	0.6469 (3)
wisconsin	0.9823 (1)	0.9709 (2)	0.9135 (5)	0.9127 (6)	0.9194 (3)	0.9160 (4)
vehicle3	0.8205 (1)	0.7984 (2)	0.4997 (6)	0.5008 (5)	0.5016 (3)	0.5016 (3)
vehicle2	0.9815 (1)	0.9689 (2)	0.5000 (3)	0.5000 (3)	0.5000 (3)	0.5000 (3)
vehicle1	0.8054 (1)	0.7842 (2)	0.5085 (4)	0.5080 (5)	0.5080 (5)	0.5099 (3)
vehicle0	0.9678 (1)	0.9597 (2)	0.5000 (3)	0.5000 (3)	0.5000 (3)	0.5000 (3)
segment0	0.9958 (1)	0.9934 (2)	0.9854 (5)	0.9833 (6)	0.9862 (3)	0.9859 (4)
pima	0.7889 (1)	0.7753 (2)	0.6517 (6)	0.6565 (5)	0.6751 (3)	0.6625 (4)
newthyroid2	0.9885 (1)	0.9727 (2)	0.9409 (3)	0.9340 (5)	0.8998 (6)	0.9346 (4)
newthyroid1	0.9953 (1)	0.9869 (2)	0.9192 (5)	0.9222 (4)	0.9273 (3)	0.9119 (6)
iris0	0.9976 (5)	0.9908 (6)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)
haberman	0.6394 (1)	0.6142 (2)	0.4997 (5)	0.4993 (6)	0.5001 (4)	0.5098 (3)
glass6	0.9500 (1)	0.9191 (2)	0.8755 (3)	0.8475 (5)	0.8588 (4)	0.8248 (6)
glass1	0.7966 (1)	0.7817 (2)	0.7477 (3)	0.7062 (6)	0.7253 (5)	0.7414 (4)
glass0	0.8622 (1)	0.8391 (2)	0.7664 (6)	0.7852 (4)	0.7767 (5)	0.7878 (3)
glass0123vs4565	0.9551 (1)	0.9522 (2)	0.8860 (6)	0.8975 (5)	0.9014 (3)	0.9002 (4)
ecoli3	0.9102 (1)	0.8996 (2)	0.7768 (3)	0.7694 (4)	0.7550 (5)	0.7539 (6)
ecoli2	0.9249 (1)	0.9103 (2)	0.8577 (4)	0.8596 (3)	0.8508 (5)	0.8417 (6)
ecoli1	0.9246 (1)	0.9219 (2)	0.8668 (3)	0.8500 (4)	0.8443 (5)	0.8297 (6)
ecoli0vs15	0.9841 (1)	0.9832 (2)	0.9694 (5)	0.9706 (4)	0.9692 (6)	0.9732 (3)
AVERAGE	0.9038 (1.19)	0.8908 (2.19)	0.7745 (4.286)	0.7713 (4.476)	0.7719 (4)	0.772 (3.905)

0.7713; the difference with the ICDPPCP algorithm is 13.25%. This result shows that using *the co-evolution methods give much better results than using other single EAs.*

3.5. Summary

In this chapter, the authors propose a competitive co-evolutionary algorithm named IBDPPCP and a co-operative co-evolutionary algorithm (named IBMCCA) for imbalanced data classification. The proposed algorithms take advantage of their strengths in creating sets of individuals that have both convergence and diversity factors to generate a collection of subsets of data that are used to generate classifiers in ensemble learning algorithms. Combined with hybrid data sampling solutions, IBDPPCP and IBMCCA have shown a good ability to handle problems related to imbalanced data. Experimental results on 42 datasets and comparisons with many other algorithms have clearly demonstrated

Table 3.14: Experimental results of the proposed algorithm and evolutionary computation learning algorithms on datasets with IR higher than 9. The values are presented in the form of mean (rank)

Data	**IBDPPCP**	**IBMCCA**	**GA**	**DE**	**PSO**	**NSGA-II**
vowel0	0.9810 (1)	0.9700 (2)	0.9586 (4)	0.9600 (3)	0.9427 (7)	0.9458 (5)
shuttlec2vsc4	1.0000 (1)	1.0000 (1)	0.9833 (5)	0.9987 (3)	0.9167 (7)	0.9667 (6)
shuttlec0vsc4	0.9999 (6)	0.9989 (7)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)
glass5	0.9110 (3)	0.8907 (5)	0.9476 (1)	0.7776 (7)	0.8943 (4)	0.9142 (2)
glass4	0.9483 (1)	0.8857 (2)	0.8033 (5)	0.8597 (3)	0.7889 (7)	0.8034 (4)
glass2	0.7921 (1)	0.7646 (2)	0.5667 (6)	0.5883 (4)	0.6413 (3)	0.5772 (5)
glass016vs5	0.9380 (1)	0.8978 (3)	0.8681 (4)	0.8238 (7)	0.8581 (5)	0.8329 (6)
glass016vs2	0.7515 (1)	0.7068 (2)	0.5906 (6)	0.6056 (3)	0.5840 (7)	0.5913 (5)
ecoli4	0.9534 (1)	0.9015 (2)	0.8118 (6)	0.7618 (7)	0.8389 (4)	0.8290 (5)
ecoli0137vs26	0.8570 (1)	0.8211 (5)	0.7903 (6)	0.7712 (7)	0.8372 (3)	0.8378 (2)
yeast6	0.8894 (1)	0.8623 (2)	0.6982 (7)	0.7324 (4)	0.7438 (3)	0.7223 (5)
yeast5	0.9745 (1)	0.9613 (2)	0.8485 (7)	0.8633 (4)	0.8660 (3)	0.8519 (5)
yeast4	0.8414 (1)	0.8064 (2)	0.6611 (7)	0.6798 (3)	0.6663 (5)	0.6677 (4)
yeast2vs8	0.8412 (1)	0.8334 (2)	0.7318 (5)	0.7311 (6)	0.7346 (4)	0.7311 (6)
yeast2vs4	0.9392 (1)	0.9225 (2)	0.8358 (5)	0.8293 (7)	0.8346 (6)	0.8528 (4)
yeast1vs7	0.7258 (1)	0.7000 (2)	0.647 (4)	0.6266 (6)	0.6733 (3)	0.6290 (5)
yeast1458vs7	0.6054 (1)	0.5566 (3)	0.5523 (4)	0.5456 (5)	0.5584 (2)	0.5159 (7)
yeast1289vs7	0.7200 (1)	0.6992 (2)	0.5892 (7)	0.6182 (6)	0.6347 (3)	0.6209 (5)
yeast05679vs4	0.8488 (1)	0.8275 (2)	0.6820 (5)	0.6666 (6)	0.7226 (3)	0.6603 (7)
AVERAGE	0.852 (1.33)	0.8263 (2.57)	0.7666 (4.81)	0.76 (4.67)	0.7756 (4.1)	0.7658 (4.52)

this statement. The novelties of these algorithms include an algorithm that combines a DPP-based approach (i.e. DPPCP) and an ensemble learning that allows solving both feature selection and instance selection problems simultaneously, as well as the proposal of a multi-objective co-operative co-evolutionary algorithm with dual population. Both of these methods are capable of discovering a collection of classifiers that have both convergence and diversity aspects for ensemble learning algorithms.

The proposed methods in this chapter were published in the journal of Research and Development on Information and Communication Technology [J2], NAFOSTED Conference on Information and Computer Science [C2], the 11th International Conference on Knowledge and Systems Engineering (KSE)(Scopus- C3), the 15th KSE [C6] and the National Science Workshop 2021 - Some selected issues of Information and Communication Technology [C7].

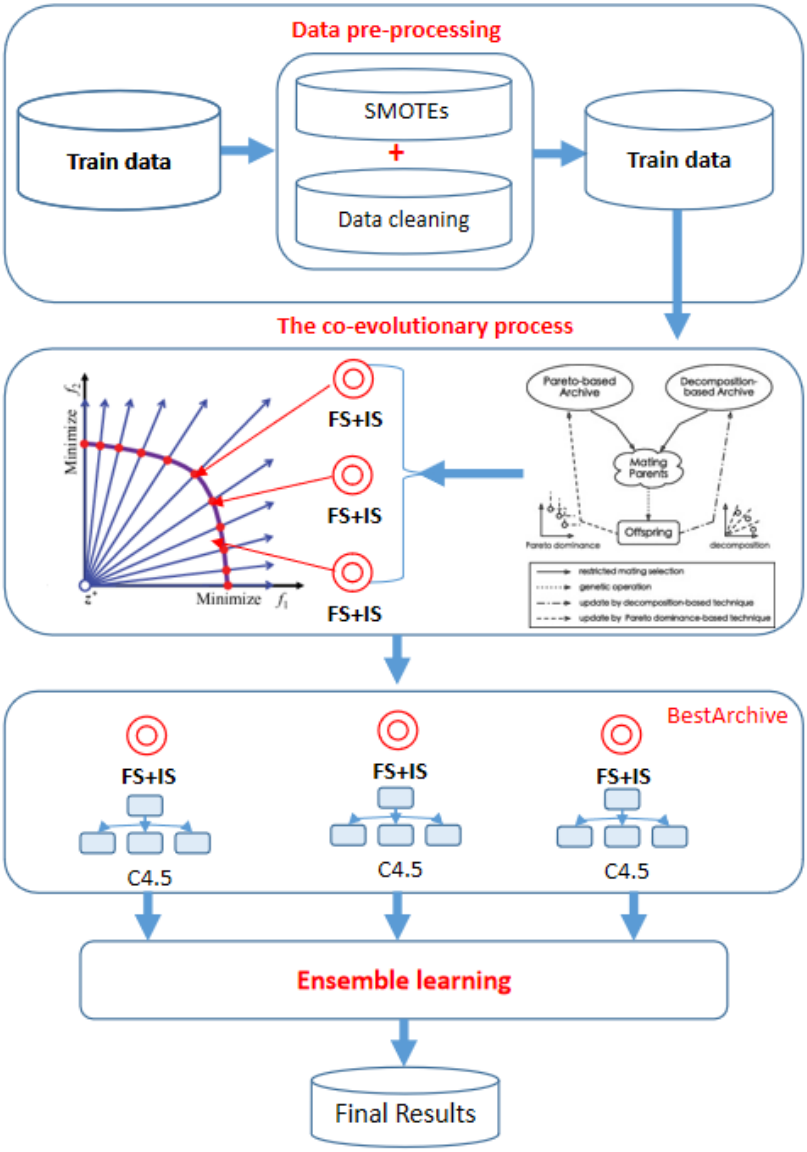


Figure 3.1: The general model of the proposed method. There are three main phases: Data pre-processing; the co-evolutionary process; and ensemble-based decision-making

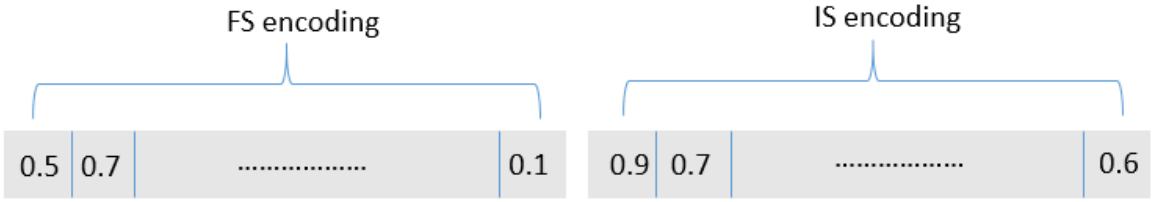


Figure 3.2: Individual encoding. Each individual is encoded as a sequence of real-valued numbers representing the probability of being selected. There are two sub-sequences, one representing the FS set and the other representing the IS set.

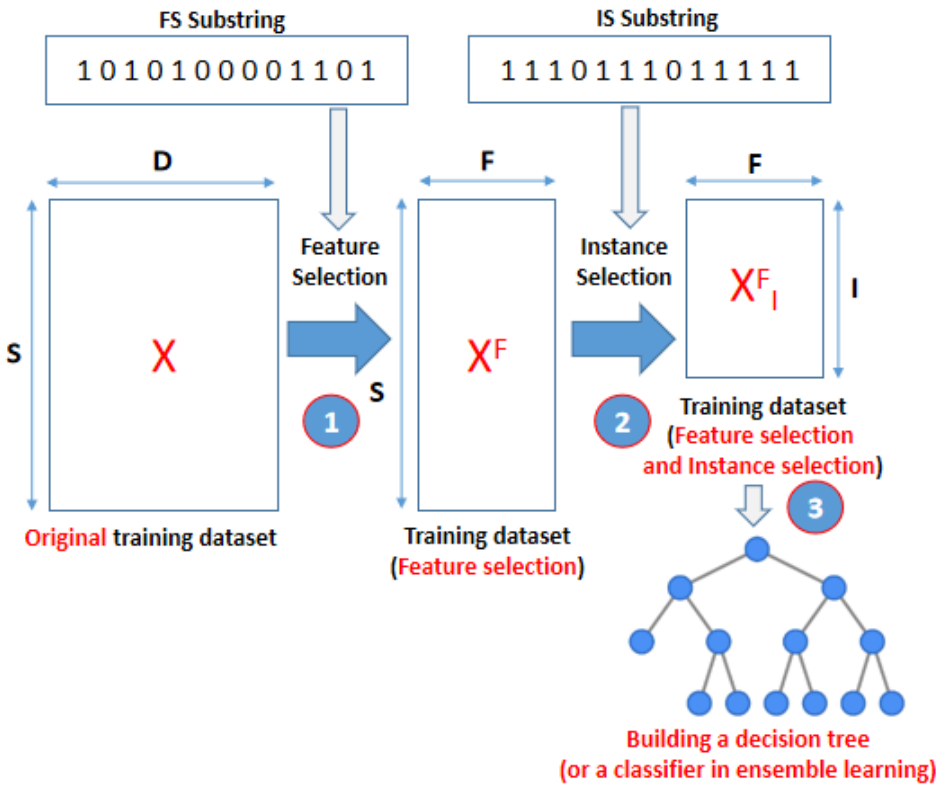


Figure 3.3: The way to build a decision tree from an individual. From the original dataset, use the FS encoding string to eliminate columns corresponding to bits with a probability of selection less than 0.5, and use the IS encoding string to eliminate rows corresponding to bits with a probability of selection less than 0.5.

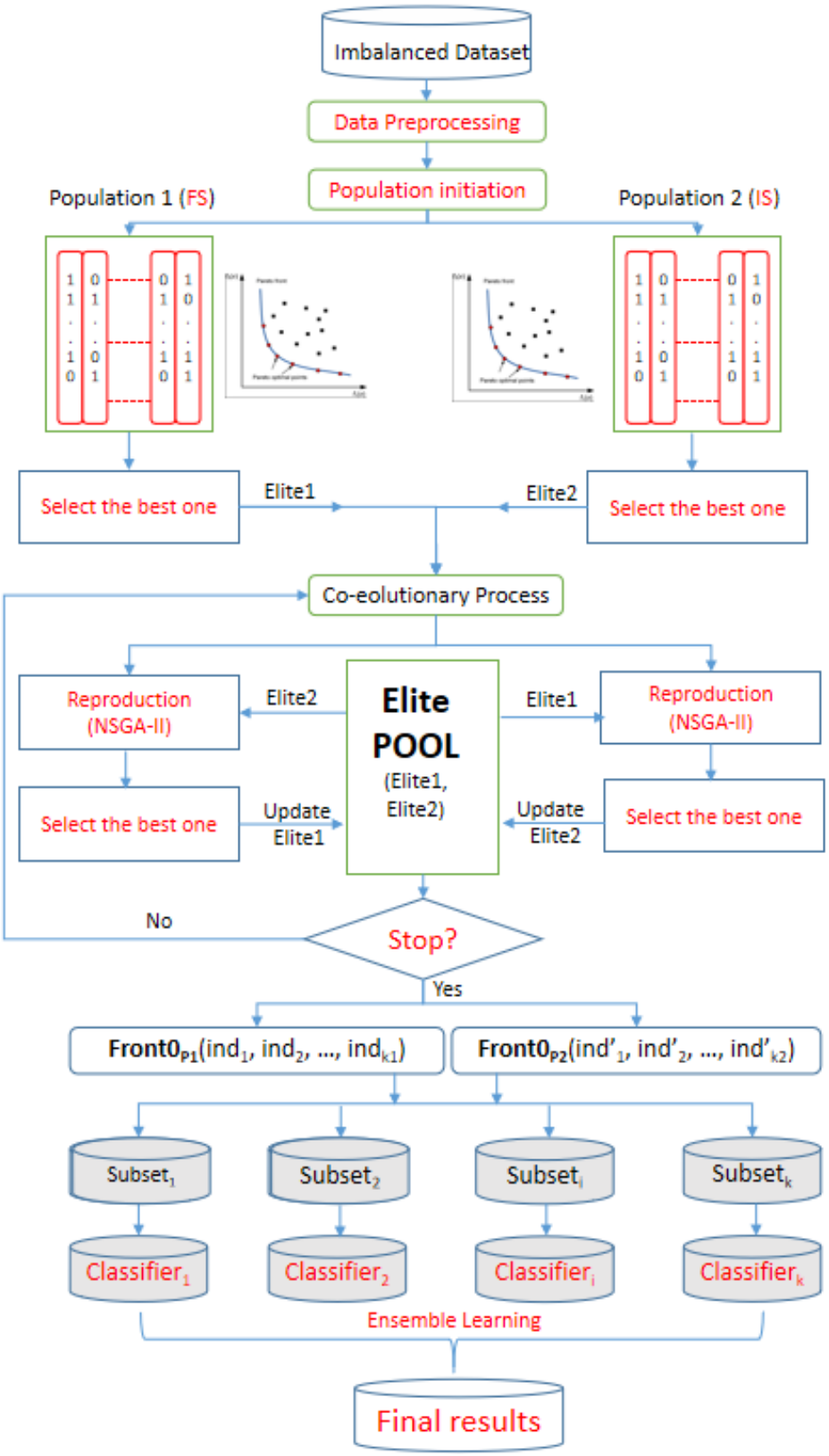


Figure 3.4: The multi-objective co-operative co-evolutionary method for classification with imbalanced data

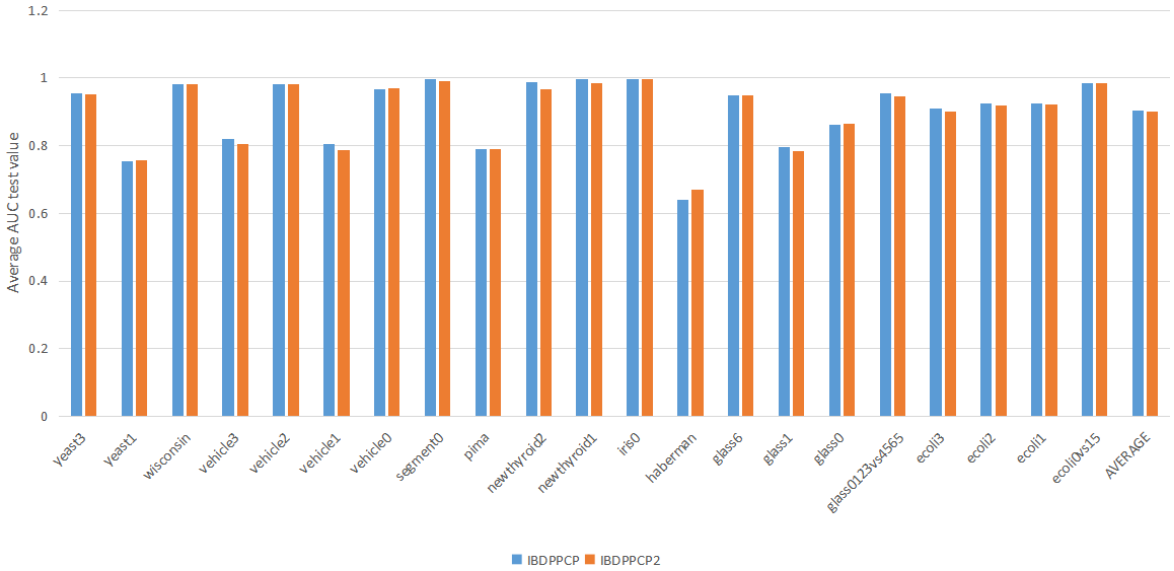


Figure 3.5: Experimental results of the IBDPPCP and IBDPPCP2 on datasets with IR less than 9. For each pair, the column that has a higher value is considered better.

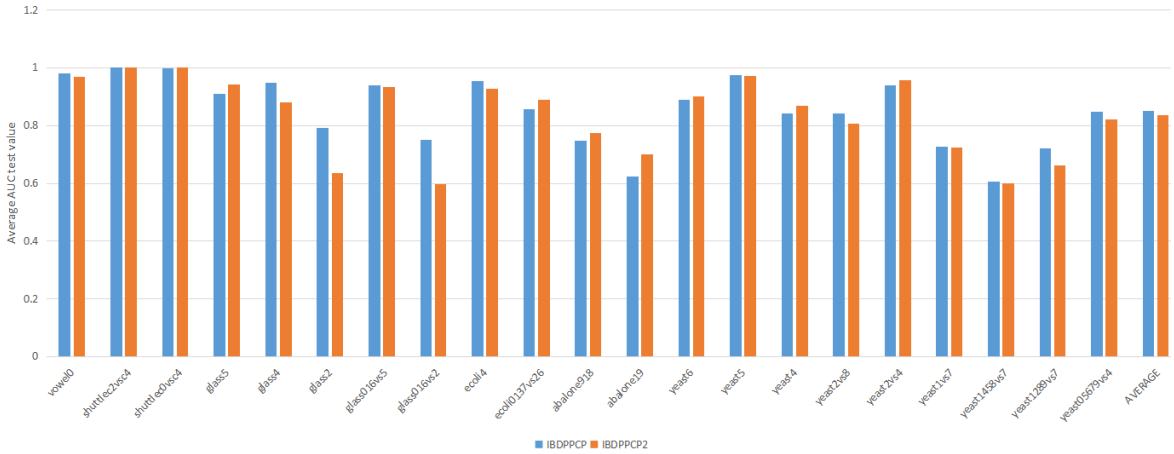


Figure 3.6: Experimental results of the IBDPPCP and IBDPPCP2 on datasets with IR higher than 9. For each pair, the column that has a higher value is considered better.

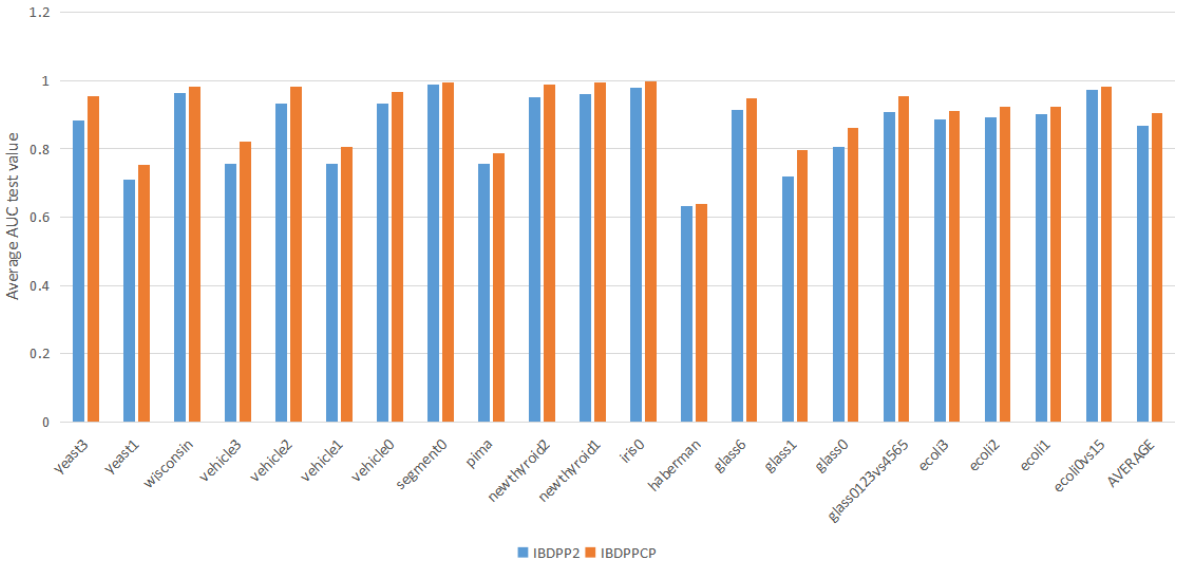


Figure 3.7: Experimental results of the IBDPPCP and IBDPP2 on datasets with IR less than 9. For each pair, the column that has a higher value is considered better.

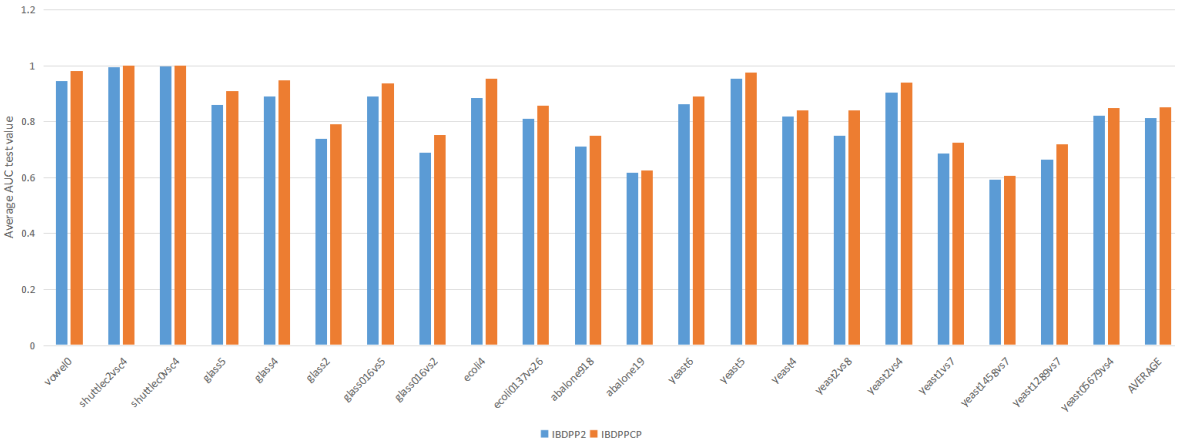


Figure 3.8: Experimental results of the IBDPPCP and IBDPP2 on datasets with IR higher than 9. For each pair, the column that has a higher value is considered better.

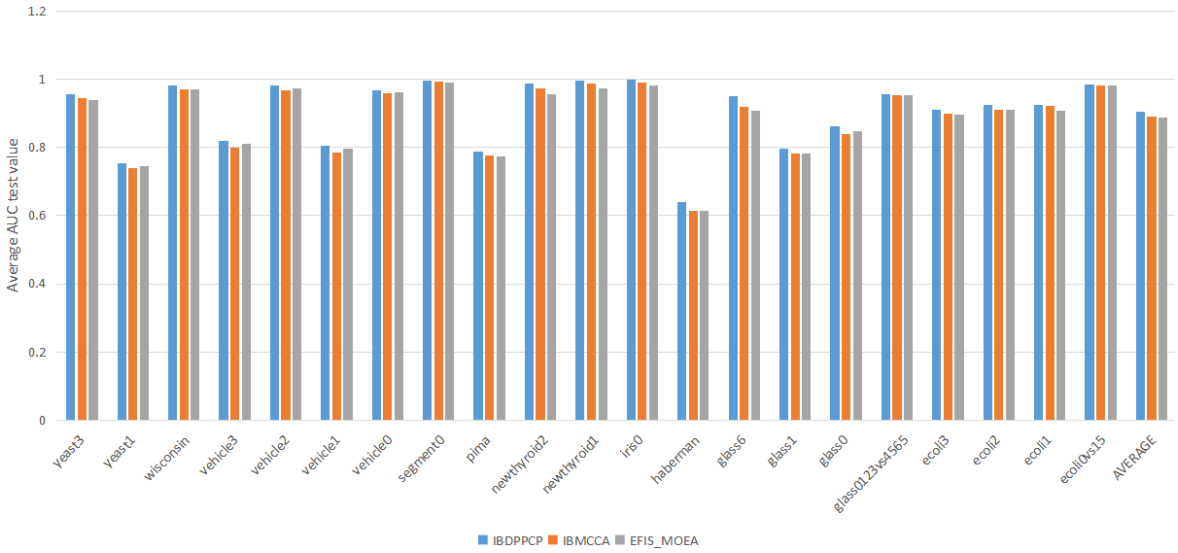


Figure 3.9: Experimental results of the two proposed methods and the premise research on datasets with IR less than 9. The column that has a higher value is considered better.

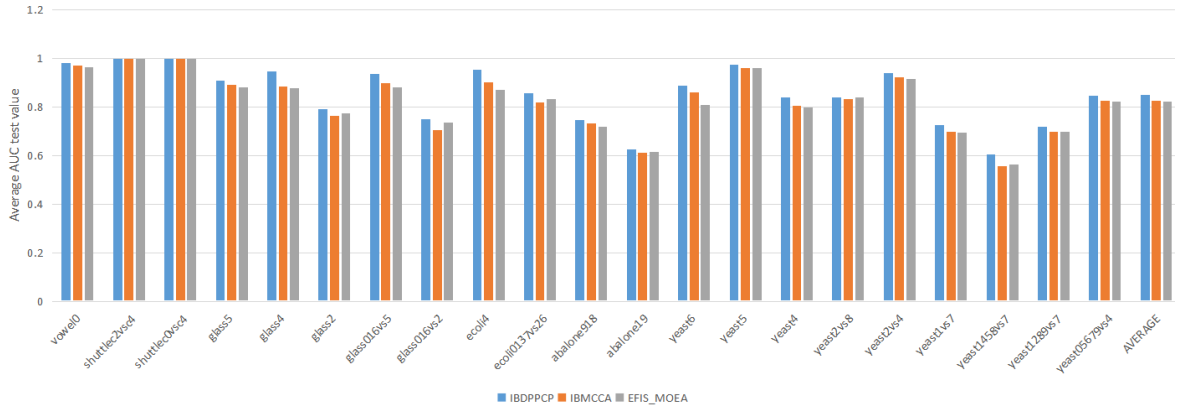


Figure 3.10: Experimental results of the the two proposed methods and the premise research on datasets with IR higher than 9. The column that has a higher value is considered better.

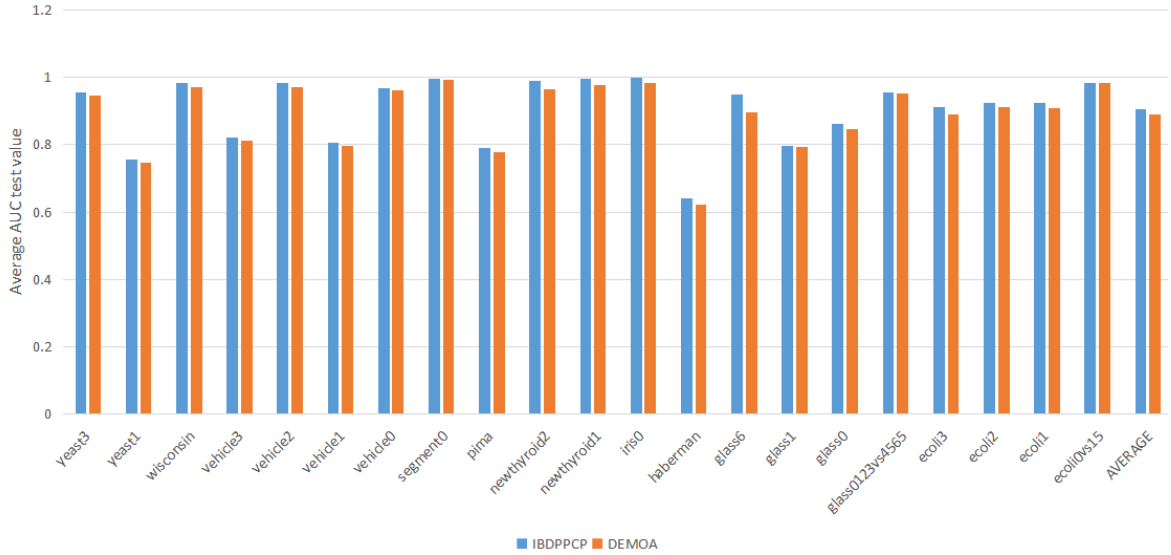


Figure 3.11: Experimental results of IBDPPCP and DEMOA on datasets with IR less than 9

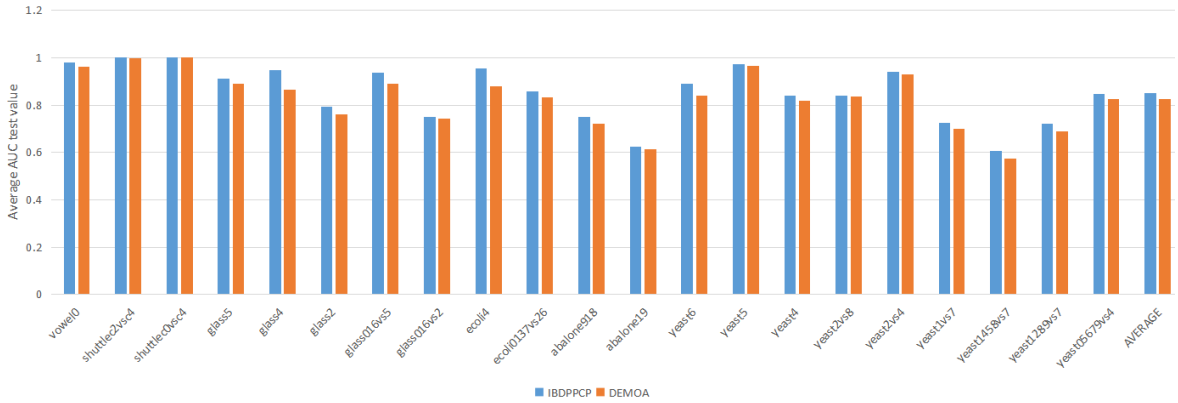


Figure 3.12: Experimental results of IBDPPCP and DEMOA on datasets with IR higher than 9

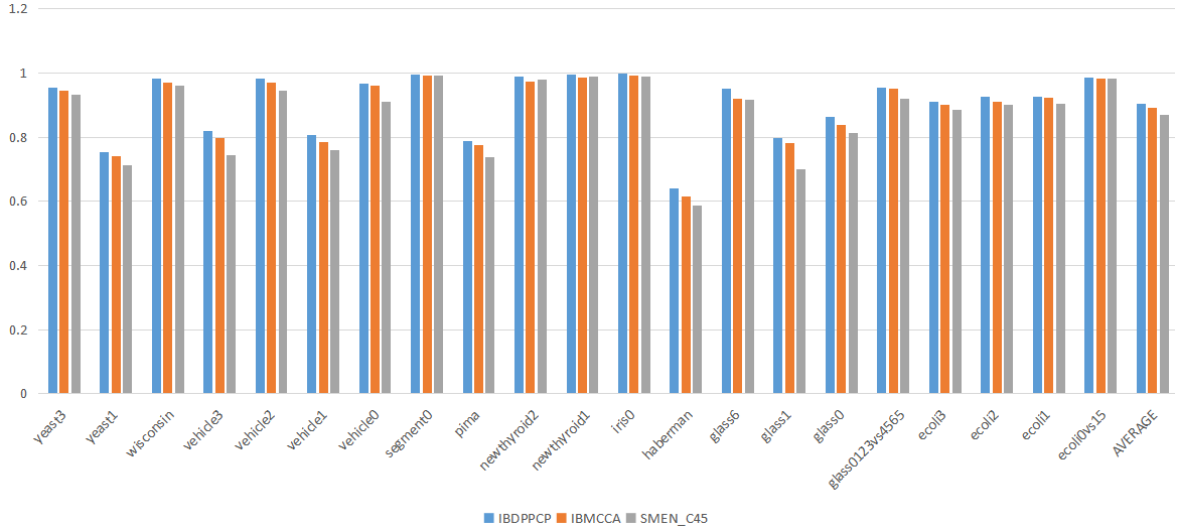


Figure 3.13: Experimental results of the proposed methods with SMEN_C45 on datasets with IR less than 9

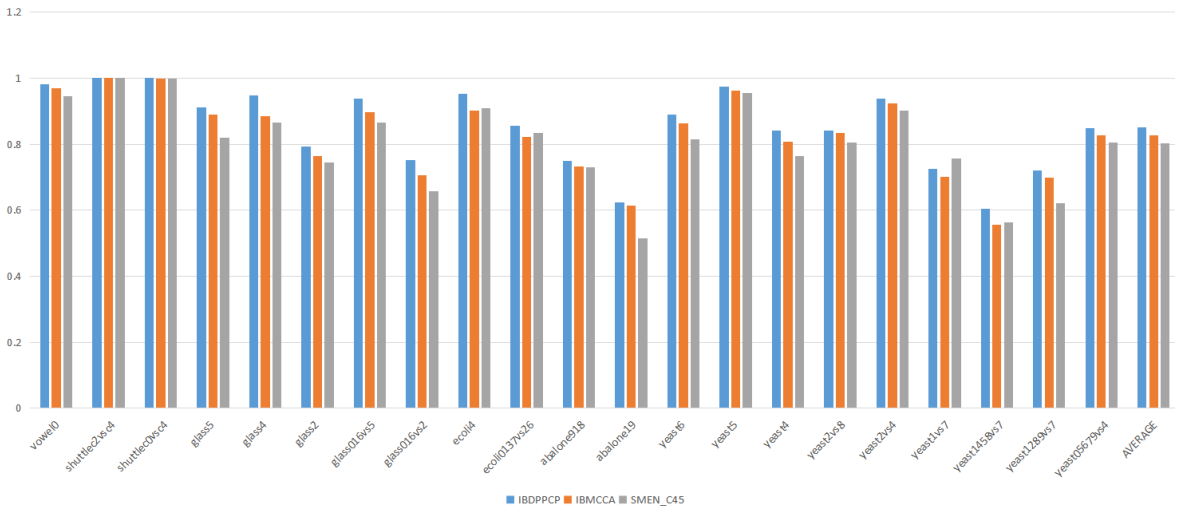


Figure 3.14: Experimental results of the proposed methods with SMEN_C45 on datasets with IR higher than 9

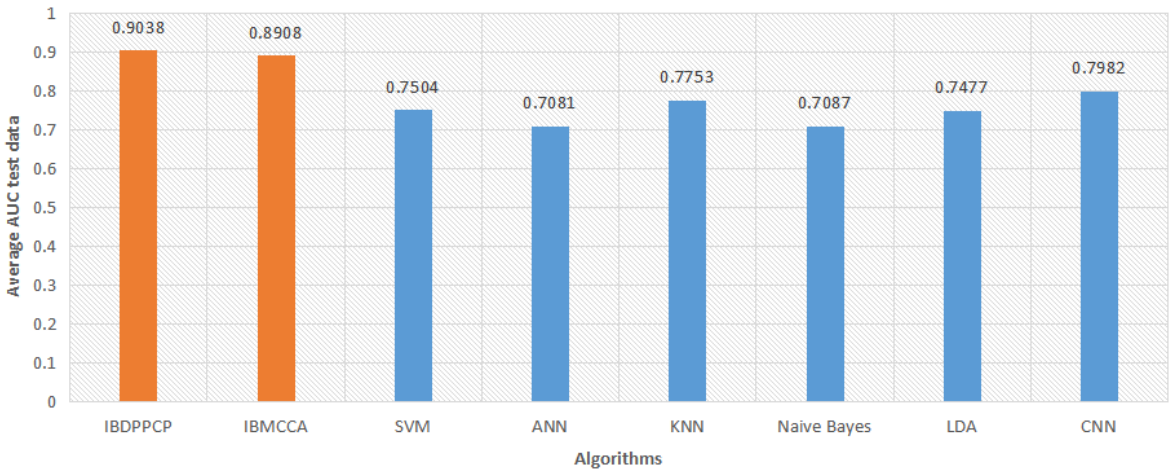


Figure 3.15: Experimental results of the proposed algorithm and machine learning algorithms on datasets with IR less than 9. The column that has a higher value is considered better.

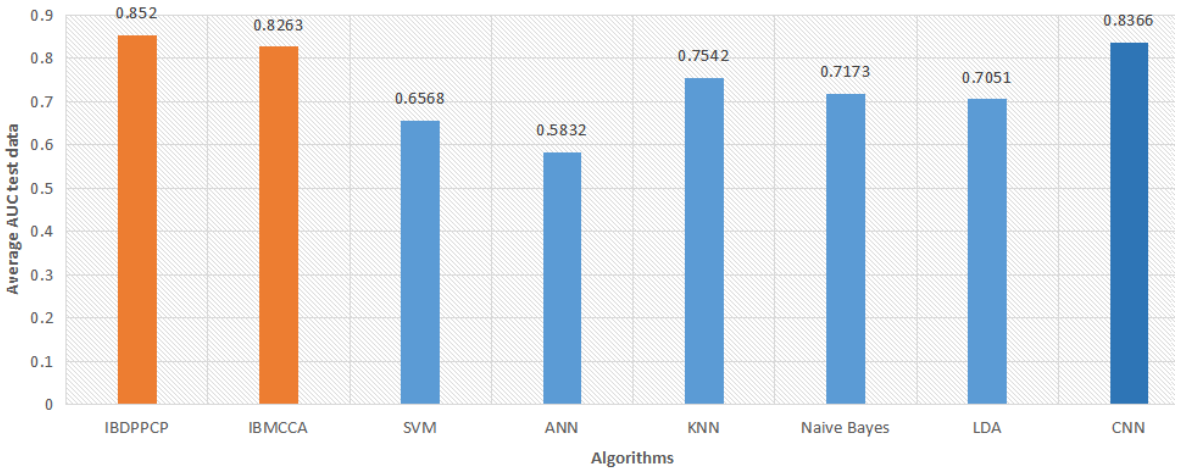


Figure 3.16: Experimental results of the proposed algorithm and machine learning algorithms on datasets with IR higher than 9. The column that has a higher value is considered better.

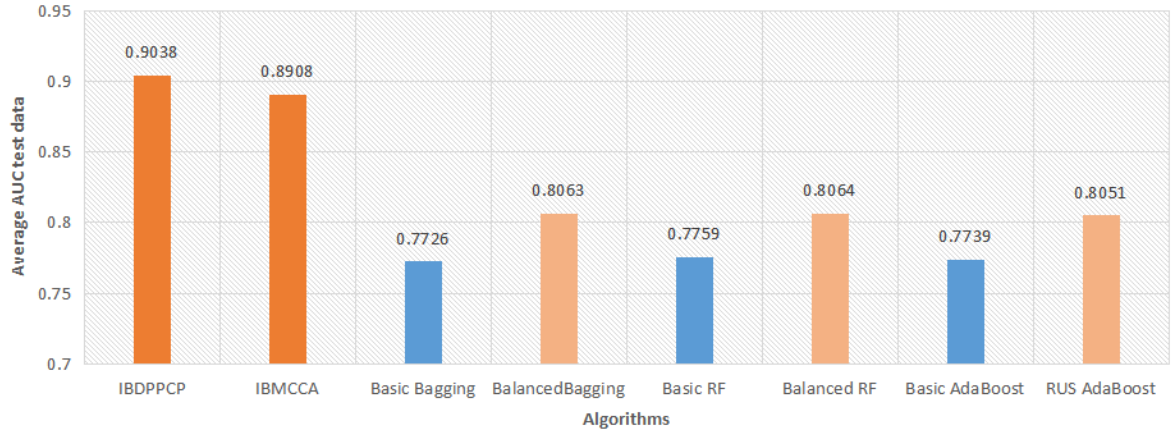


Figure 3.17: Experimental results of the proposed algorithm and ensemble learning algorithms on datasets with IR lower than 9. The column that has a higher value is considered better.

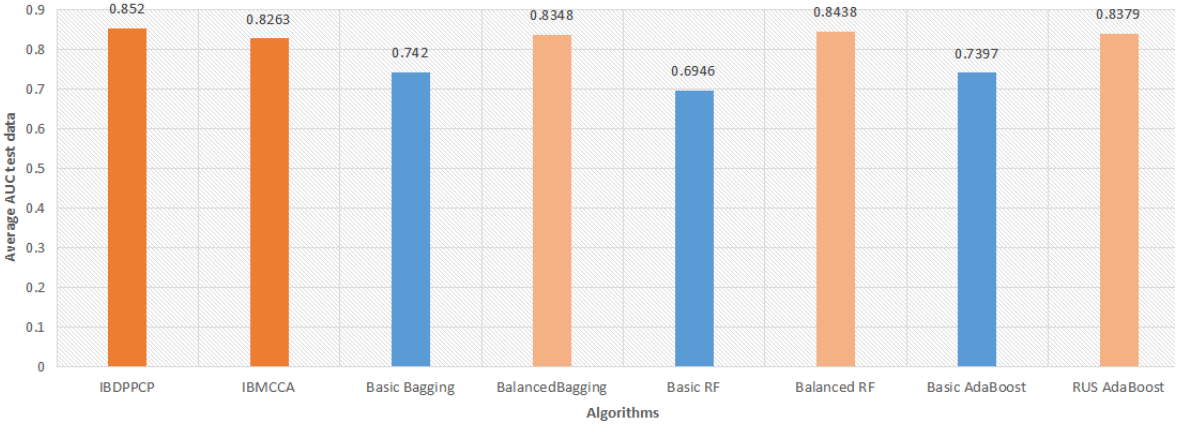


Figure 3.18: Experimental results of the proposed algorithm and ensemble learning algorithms on datasets with IR higher than 9. The column that has a higher value is considered better.

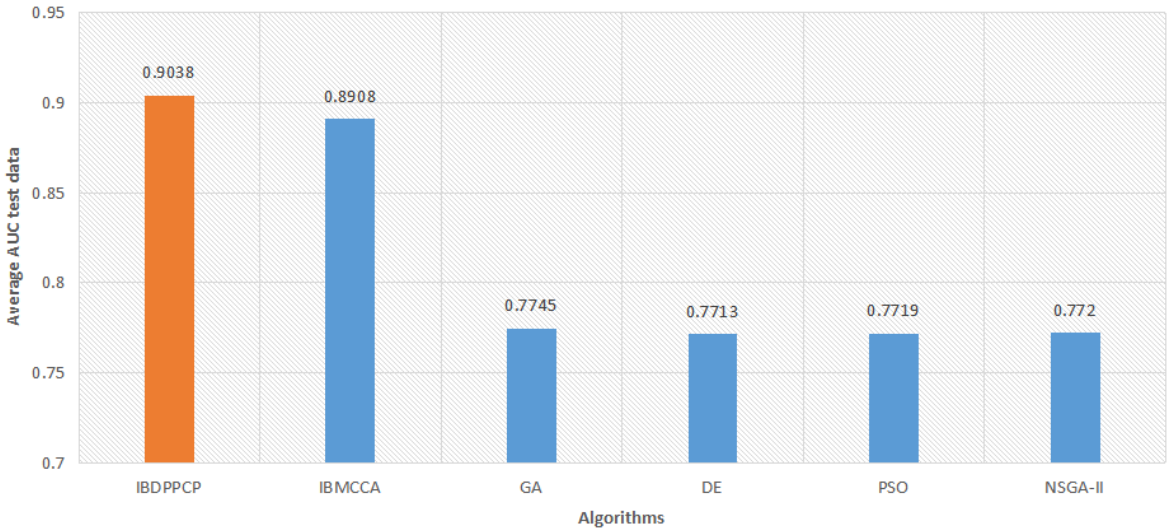


Figure 3.19: Experimental results of the proposed algorithm and Evolutional computation learning algorithms on datasets with IR lower than 9. The column that has a higher value is considered better.

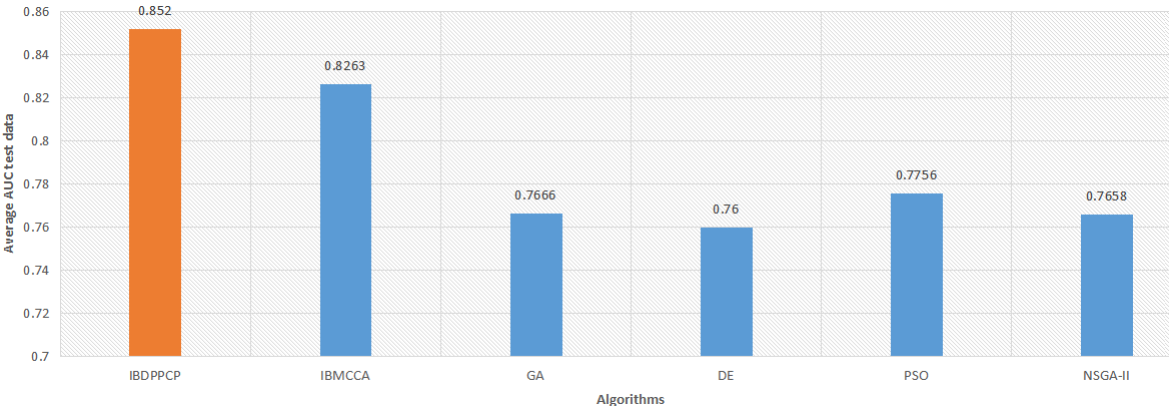


Figure 3.20: Experimental results of the proposed algorithm and Evolutional computation learning algorithms on datasets with IR higher than 9. The column that has a higher value is considered better.

CONCLUSIONS AND SUGGESTIONS FOR FUTURE STUDIES

This section summarizes the contributions of the thesis and presents some open problems for future studies.

A. Conclusions

In this thesis, the author discussed the essential theory of co-evolutionary algorithms, multi-objective optimization, as well as some of its current applications. This thesis focuses on addressing the following two major issues. The first involves developing algorithms to improve in balancing convergence and diversity in multi-objective optimization problems, and the second is using those methods to resolve classification issues. Following is a summary of the major contributions made in this thesis.

(*) Balancing convergence and diversity in multi-objective optimization problems

- Proposing a DPP-based co-operative co-evolutionary approach for balancing the convergence and diversity. Key improvements of the algorithm include a new restricted mating selection mechanism (named RMS2); a new solution-alternative selection strategy as well as a new mechanism for instance updates[C1].
- Proposing a DPP-based competitive co-evolutionary approach for balancing the convergence and diversity. The novelties of this study include: Proposing a mechanism for selecting individuals for each population (named NBSM selection) and proposing a competitive co-evolutionary mechanism to make two offspring interact with each

other instead of using the co-operative co-evolutionary mechanism [J1].

(*) Applying multi-objective co-evolutionary methods for classification with imbalanced data

- Proposing a multi-objective competitive co-evolutionary approach for imbalanced dataset classification (named IBDPPCP). The main contribution of this study is to propose an approach that combines a DPP-based method (i.e.DPPCP) and an ensemble learning that allows solving both feature selection and instance selection problems simultaneously. The DPPCP algorithm helps to find a set of solutions (corresponding to different sub-data sets) to serve as the basis for building classifiers of ensemble learning that satisfy both convergence and diversity criteria. This approach helps to solve both challengers of imbalanced classification problems: imbalance and overlapping problems.[C7].
- Proposing a multi-objective co-operative co-evolutionary approach (named IBMCCA) for solving classification with imbalanced data. The main contribution of this study is to propose an multi-objective cooperative co-evolutionnary model with dual population. This model can find a set of individuals (or sub-datasets) that have both convergence and diversity factors to solve both of instance and feature selection in imbalanced dataset classification.[J2, C2, C3, C6].

Along with the models mentioned in the thesis, the author has also developed a few additional co-evolution models such as: a new multi-objective competitive co-evolutionary approach with a Prey-Predator model for solving with classification problems [C3];a multi-population co-evolutionary approach for software defect prediction [C5]; a multi-swarm co-evolutionary approaches time series forecasting problems [C8].

B. Future Studies

Although the Co-evolution was studied widely in the literature, there are still several possible open problems which require further investigations in order to have a full understanding about their applicability as follows.

- Developing the DPP-based models using both of co-operative and competitive for multi-objective optimization problems.

- Developing a multi-objective multi-population for machine learning problems.

- Developing Spatial-based co-evolutionary algorithms for solving spatial challenges such as spatial forest planning, groundwater management, etc.

3.6. PUBLICATIONS

1. Related to thesis

Journal articles:

[J1] **Van Truong VU**, Lam Thu BUI, and Trung Thanh NGUYEN. "*A competitive co-evolutionary approach for the multi-objective evolutionary algorithms.*" IEEE Access 8 (2020): 56927-56947, **SCIE, Q1, IF: 4.64**.

[J2] **Van Truong VU**, Lam Thu BUI, and Trung Thanh NGUYEN. "*An Ensemble Co-Evolutionary based Algorithm for Classification Problems.*" Journal of Research and Development on Information and Communication Technology 2019.1, 10/2019.

Conference papers:

[C1] **Van Truong VU**, Lam Thu Bui, Trung Thanh Nguyen, "*A modified dual-population approach for solving multi-objective problems.*" 2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES). IEEE, 11/2017 [**SCOPUS Conference**].

[C2] **Van Truong VU**, Lam Thu Bui, Trung Thanh Nguyen, *A Coevolutionary approach for classification problems: Preliminary results.* In: 5th NAFOSTED Conference on Information and Computer Science, NICS 2018, 23 November 2018 through 24 November 2018 [**SCOPUS Conference**].

[C3] **Van Truong VU**, Lam Thu Bui, Trung Thanh Nguyen, *A multi-objective cooperative coevolutionary approach for remote sensing image classification*, 2019 11th International Conference on Knowledge and Systems Engineering (KSE), 10/2019 [**SCOPUS Conference**].

[C4] **Van Truong VU**; Lam Thu BUI; Trung Thanh NGUYEN, *A multi-objective competitive co-evolutionary approach for classification problems*, The NAFOSTED Conference on Information and Computer Science (NICS), 08/2019 [**SCOPUS Conference**].

[C5] Lam Thu Bui, **Van Truong VU**, Bich Van Pham, Viet Anh Phan,

A multi-population coevolutionary approach for Software defect prediction with imbalanced data, 2022 14th International Conference on Knowledge and Systems Engineering (KSE), 10/2022.

[C6] Lam Thu Bui, **Van Truong VU**, Huong Dinh Thi Thu, Hang Le Minh, *A multi-objective co-operative co-evolutionary method for classification with imbalanced data*, 2023 15th International Conference on Knowledge and Systems Engineering (KSE), 10/2023 [**Accepted Paper**].

[C7] ***Van Truong VU***, *Lam Thu Bui, A decomposition-based ensemble multi-objective optimization algorithm for imbalanced dataset classification problems*, Hội thảo quốc gia lần thứ XXIV "Một số vấn đề chọn lọc của công nghệ thông tin và truyền thông, 12/2021.

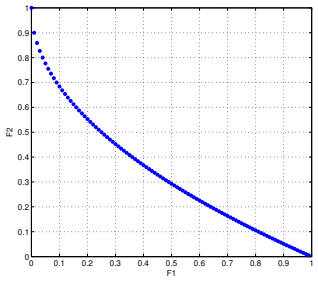
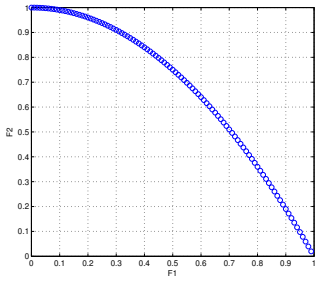
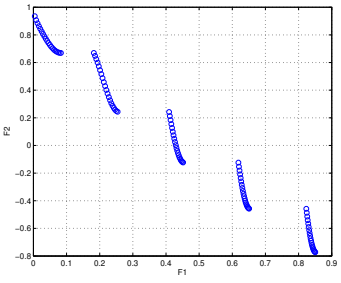
2. Other publications

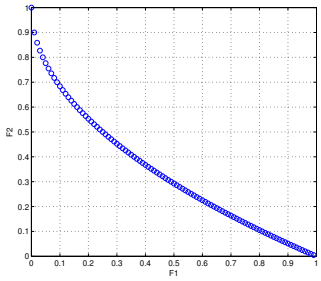
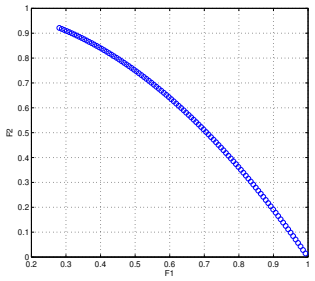
[C8]. Thi Thu Huong Dinh **, Van Truong VU**, Lam Thu Bui, *An ensemble multi-objective particle swarm optimization approach for exchange rates forecasting problem*, The 4th International Conference on Machine Learning and Soft Computing (ICMLSC 2020), 1/2020 [**SCOPUS Conference**].

Appendix 4

Benchmark test problems

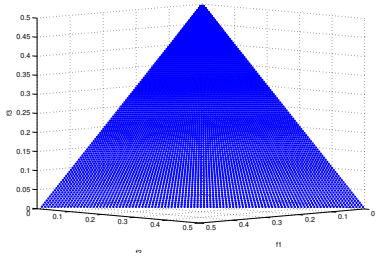
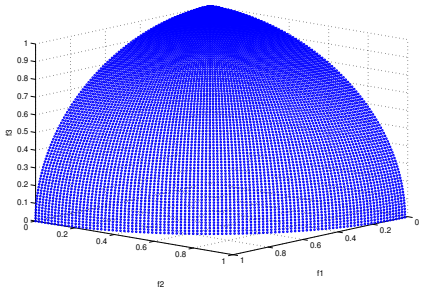
Table 4.1: ZDT Problems. Two objectives $f_1(\overrightarrow{x})$ and $f_2(\overrightarrow{x})$ have to be minimize. The function $g(\overrightarrow{x})$ can be thought of as the function for convergence.

MOP	POF
ZDT1 : $f_1(\overrightarrow{x}) = x_1,$ $f_2(\overrightarrow{x}, g) = g(\overrightarrow{x}).(1 - \sqrt{\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}}),$ $g(\overrightarrow{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i.$ where n = 30, and $x_i \in [0, 1]$; $g(\overrightarrow{x}) = 1$. The Pareto-optimal front (POF) is *convex*.	
ZDT2 : $f_1(\overrightarrow{x}) = x_1,$ $f_2(\overrightarrow{x}, g) = g(\overrightarrow{x}).(1 - (\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})})^2),$ $g(\overrightarrow{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i.$ where n = 30, and $x_i \in [0, 1]$; $g(\overrightarrow{x}) = 1$. The POF is *non-convex*.	
ZDT3 : $f_1(\overrightarrow{x}) = x_1,$ $f_2(\overrightarrow{x}, g) = g(\overrightarrow{x}).(1 - \sqrt{\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}} - \frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}.\sin(10\pi f_1(\overrightarrow{x}))),$ $g(\overrightarrow{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i.$ where $n = 30$, and $x_i \in [0, 1]$; $g(\overrightarrow{x}) = 1$. The POF is *disconnected*.	

MOP	POF
ZDT4 : $f_1(\overrightarrow{x}) = x_1,$ $f_2(\overrightarrow{x}, g) = g(\overrightarrow{x}).(1 - \sqrt{\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}}),$ $g(rrightarrowx) = 1 + 10.(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i)).$ where $n = 10$, $x_1 \in [0,1]$ and $x_2, ..., x_n \in [-5,5]$; $g(\overrightarrow{x}) = 1$. The POF is *convex*. This problem has a large number of local Pareto-optimal solutions (As a result, algorithms are prone to become trapped in a local optimum).	
ZDT6 : $f_1(\overrightarrow{x}) = 1 - exp(-4x_1).\sin^6(6\pi x_1),$ $f_2(\overrightarrow{x}, g) = g(\overrightarrow{x}).(1 - (\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})})^2),$ $g(\overrightarrow{x}) = 1 + 9(\frac{1}{9}.\sum_{i=2}^{n}(x_i)).$ where $n = 10$, $x_i \in [0,1]$; $g(\overrightarrow{x}) = 1$. The POF is *non-convex*. The Pareto-optimal region has non-uniform distribution of solution density.	

natbib

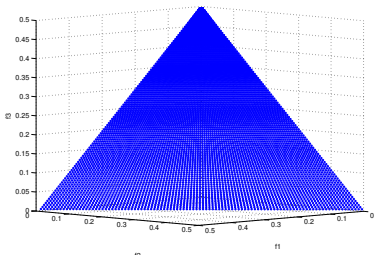
Table 4.2: DTLZ Problems

MOP	POF
DTLZ1 : $f_1(\overrightarrow{x}) = \frac{1}{2}x_1x_2...x_{M-1}(1+g(\overrightarrow{x}_M))$, $f2(\overrightarrow{x}) = \frac{1}{2}x_1x_2...(1-x_{M-1})(1+g(\overrightarrow{x}_M))$, ..., $f_{M-1}(\overrightarrow{x}) = \frac{1}{2}x1(1-x_2)(1+g(\overrightarrow{x}_M))$, $f_M(\overrightarrow{x}) = \frac{1}{2}(1-x_1)(1+g(\overrightarrow{x}_M))$. subject to $0 \leq x_i \leq 1, \forall i = 1,2,...,n$ where: $\overrightarrow{x}_M = x_M, x_{M+1}, ..., x_n$ and $g(\overrightarrow{x}_M) = 100[\lvert\overrightarrow{x}_M\rvert + \sum_{x_i \in \overrightarrow{x}_M}(x_i - 0.5)^2 - cos(20\pi(x_i - 0.5))]$. The Pareto-optimal solution corresponds to $\overrightarrow{x}^*_M = 0.5$ and the objective function values on the linear hyper-plane: $\sum_{m=1}^{M} f_i = 0.5$. Convergence to the Pareto-optimal hyper-plane is the problem's only challenge. There are $(11^k - 1)$ local POFs in the search space where a MOEA may be drawn before reaching the global POF. The POF is linear, *separable* and *multi-modal*.	
DTLZ2 : $f_1(\overrightarrow{x}) = (1+g(\overrightarrow{x}_M))\cos(x_1\frac{\pi}{2})\cos(x_2\frac{\pi}{2})...$ $\cos(x_{M-2}\frac{\pi}{2})\cos(x_{M-1}\frac{\pi}{2})$, $f_2(\overrightarrow{x}) = (1+g(\overrightarrow{x}_M))\cos(x_1\frac{\pi}{2})\cos(x_2\frac{\pi}{2})...$ $\cos(x_{M-2}\frac{\pi}{2})\sin(x_{M-1}\frac{\pi}{2})$, $f_3(\overrightarrow{x}) = (1+g(\overrightarrow{x}_M))\cos(x_1\frac{\pi}{2})cos(x_2\frac{\pi}{2})...sin(x_{M-2}\frac{\pi}{2})$, ..., $f_{M-1}(\overrightarrow{x}) = (1+g(\overrightarrow{x}_M))\cos(x_1\frac{\pi}{2})sin(x_2\frac{\pi}{2})$, $f_M(\overrightarrow{x}) = (1+g(\overrightarrow{x}_M))sin(x_1\frac{\pi}{2})$. subject to $0 \leq x_i \leq 1, \forall i = 1,2,...,n$ where: $\overrightarrow{x}_M = x_M, x_{M+1}, ..., x_n$ and $g(\overrightarrow{x}_M) = \sum_{x_i \in \overrightarrow{x}_M}(x_i - 0.5)^2$ The Pareto-optimal solutions corresponds to $x_i = 0.5$ for all $x_i \in \overrightarrow{x}_M, \forall i = M, M+1, ..., n$ and all objective function values must satisfy: $\sum_{i=1}^{M}(f_i)^2 = 1$. The POF is *concave*, *scalable* and *multi-modal*.	

MOP POF

DTLZ3 :

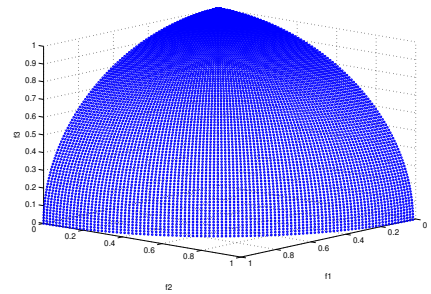
$$f_1(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(x_1 \frac{\pi}{2}) \cos(x_2 \frac{\pi}{2})...$$
$$\cos(x_{M-2} \frac{\pi}{2}) \cos(x_{M-1} \frac{\pi}{2}),$$
$$f_2(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(x_1 \frac{\pi}{2}) \cos(x_2 \frac{\pi}{2})...$$
$$\cos(x_{M-2} \frac{\pi}{2}) \sin(x_{M-1} \frac{\pi}{2}),$$
$$f_3(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(x_1 \frac{\pi}{2}) cos(x_2 \frac{\pi}{2})...$$
$$sin(x_{M-2} \frac{\pi}{2}),$$
$$...,$$
$$f_{M-1}(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(x_1 \frac{\pi}{2}) sin(x_2 \frac{\pi}{2}),$$
$$f_M(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) sin(x_1 \frac{\pi}{2}).$$



subject to $0 \leq x_i \leq 1, \forall i = 1, 2, ..., n$ where: $\overrightarrow{x}_M = x_M, x_{M+1}, ..., x_n$ and $g(\overrightarrow{x}_M) = 100[|\overrightarrow{x}_M| + \sum_{x_i \in \overrightarrow{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]$
A global POF and $(3^k - 1)$ local POFs are introduced by the g function. An MOEA can become stuck at any of the local POFs that are parallel to the global POF before convergent reaching the global POF .

DTLZ4 :

$$f_1(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(x_1^\alpha \frac{\pi}{2}) \cos(x_2^\alpha \frac{\pi}{2})...$$
$$\cos(x_{M-2}^\alpha \frac{\pi}{2}) \cos(x_{M-2}^\alpha \frac{\pi}{2}),$$
$$f_2(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(x_1^\alpha \frac{\pi}{2}) \cos(x_2^\alpha \frac{\pi}{2})...$$
$$\cos(x_{M-2}^\alpha \frac{\pi}{2}) \sin(x_{M-1}^\alpha \frac{\pi}{2}),$$
$$f_3(x) = (1 + g(\overrightarrow{x}_M)) \cos(x_1^\alpha \frac{\pi}{2}) \cos(x_2^\alpha \frac{\pi}{2})...$$
$$\sin(x_{M-2}^\alpha \frac{\pi}{2}),$$
$$...,$$
$$f_{M-1}(x) = (1 + g(\overrightarrow{x}_M)) \cos(x_1^\alpha \frac{\pi}{2}) \sin(x_2^\alpha \frac{\pi}{2}),$$
$$fM(x) = (1 + g(\overrightarrow{x}_M)) \sin(x_1^\alpha \frac{\pi}{2}).$$



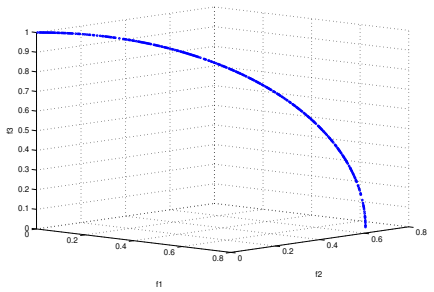
subject to $0 \leq x_i \leq 1, \forall i = 1, 2, ..., n$ where: $\overrightarrow{x}_M = x_M, x_{M+1}, ..., x_n$ and $g(\overrightarrow{x}_M) = \sum_{x_i \in \overrightarrow{x}_M} (x_i - 0.5)^2$.
This is another version of the DTLZ2 problem with a modified parametric variable mapping to examine a MOEA's capability to preserve a good distribution of solutions. Here, the parameter $\alpha = 100$ is recommended.

MOP	POF

DTLZ5 :

$$f_1(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) \cos(\theta_2 \frac{\pi}{2}) ...$$
$$\cos(\theta_{M-2} \frac{\pi}{2}) \cos(\theta_{M-1} \frac{\pi}{2}),$$
$$f_2(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) \cos(\theta_2 \frac{\pi}{2}) ...$$
$$\cos(\theta_{M-2} \frac{\pi}{2}) \sin(\theta_{M-1} \frac{\pi}{2}),$$
$$f_3(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) cos(\theta_2 \frac{\pi}{2}) ...$$
$$sin(\theta_{M-2} \frac{\pi}{2}),$$
$$...,$$
$$f_{M-1}(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) sin(\theta_2 \frac{\pi}{2}),$$
$$f_M(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) sin(\theta_1 \frac{\pi}{2}).$$

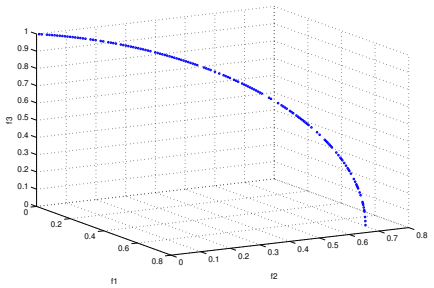
subject to $0 \leq x_i \leq 1, \forall i = 1, 2, ..., n$ where: $\theta_i = \frac{\pi}{4(1+g(\overrightarrow{x}_M))}(1 + 2g(\overrightarrow{x}_M) x_i)$, for $i = 2, 3, ..., (M - 1)$ and $g(\overrightarrow{x}_M) = \sum_{x_i \in \overrightarrow{x}_M} (x_i - 0.5)^2$.
An MOEA's capacity to converge on a curve will be examined in this problem. This challenge may be simple for an algorithm to handle because there is a natural bias toward solutions that are near to this Pareto-optimal curve.



DTLZ6 :

$$f_1(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) \cos(\theta_2 \frac{\pi}{2}) ...$$
$$\cos(\theta_{M-2} \frac{\pi}{2}) \cos(\theta_{M-1} \frac{\pi}{2}),$$
$$f_2(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) \cos(\theta_2 \frac{\pi}{2}) ...$$
$$\cos(\theta_{M-2} \frac{\pi}{2}) \sin(\theta_{M-1} \frac{\pi}{2}),$$
$$f_3(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) cos(\theta_2 \frac{\pi}{2}) ...$$
$$\sin(\theta_{M-2} \frac{\pi}{2}),$$
$$...,$$
$$f_{M-1}(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) \cos(\theta_1 \frac{\pi}{2}) sin(\theta_2 \frac{\pi}{2}),$$
$$f_M(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)) sin(\theta_1 \frac{\pi}{2}).$$

subject to $0 \leq x_i \leq 1, \forall i = 1, 2, ..., n$ where: $\theta_i = \frac{\pi}{4(1+g(\overrightarrow{x}_M))}(1 + 2g(\overrightarrow{x}_M) x_i)$, for $i = 2, 3, ..., (M - 1)$ and $g(\overrightarrow{x}_M) = \sum_{x_i \in \overrightarrow{x}_M} (x_i)^{0.1}$.
The size of the x_M vector is set to 10, and the overall number of variables is the same as in the DTLZ5 problem. Because of the aforementioned modification to the problem, some MOEAs find it challenging to reach the same true POF as DTLZ5.



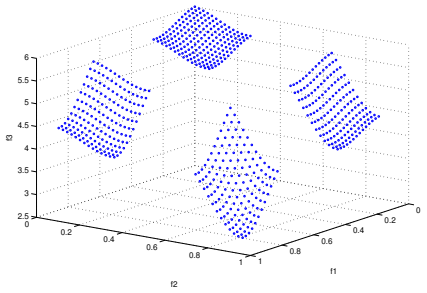
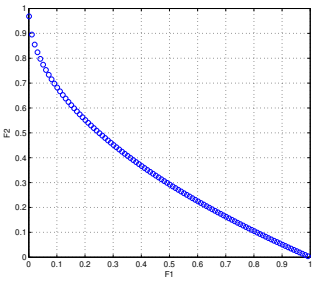
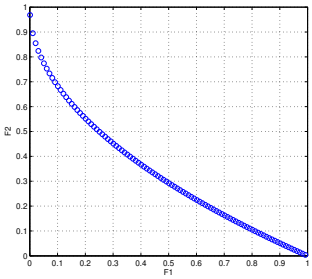
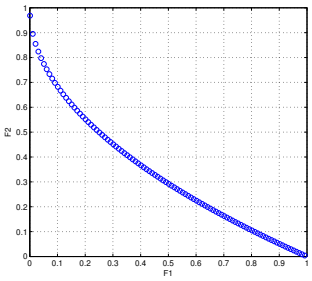
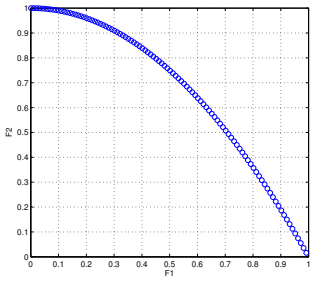
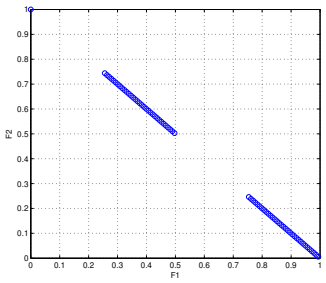
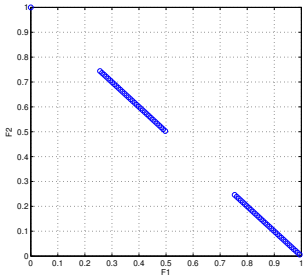
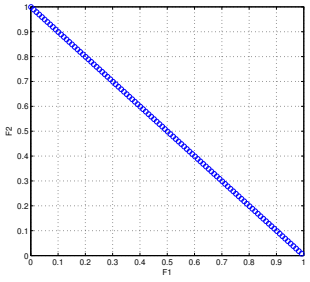
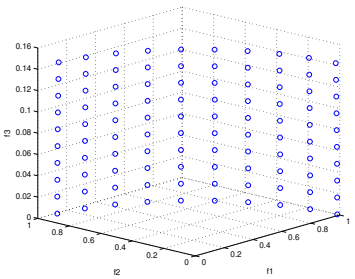
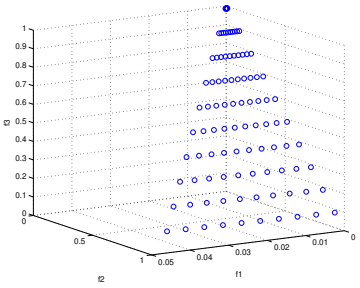
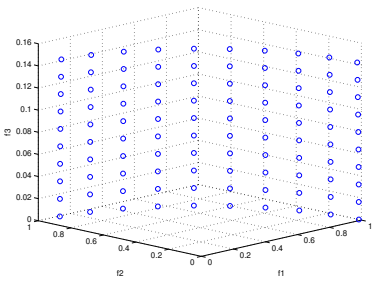
MOP	POF
DTLZ7 : $f_1(\overrightarrow{x}) = x_1,$ $f_2(\overrightarrow{x}) = x_2,$ ..., $f_{M-1}(\overrightarrow{x}) = \overrightarrow{x}_{M-1}$ $f_M(\overrightarrow{x}) = (1 + g(\overrightarrow{x}_M)).h(f_1, f_2, ..., f_{M-1}g(x))$ subject to $0 \leq x_i \leq 1, \forall i = 1, 2, ..., n$ where: $g(x) = 1 + \frac{9}{\mid \overrightarrow{x}_M \mid} \cdot \sum_{x_i \in \overrightarrow{x}_M} x_i$, $h(f_1, f_2, ..., f_{M-1}, g) = M - \sum_{i=1}^{M-1} \frac{f_i}{1+g(x)}(1 + sin(3\pi f_i))$. There are 2^{M-1} unconnected Pareto-optimal regions in the search space for this test problem. The maintenance of sub-populations in various Pareto-optimal locations will be tested by this task.	

Table 4.3: UF Problems

MOP	POF
UF1 : $f_1(\overrightarrow{x}) = x_1 + \frac{2}{\mid J_1 \mid} \sum_{j \in J_1} [x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2,$ $f_2(\overrightarrow{x}) = 1 - \sqrt{x_1} + \frac{2}{\mid J_2 \mid} \sum_{j \in J_2} [x_j - \sin(6\pi x_1 + \frac{j\pi}{n}]^2$ where $J_1 = \{j \mid j$ is odd and $(2 \leq j \leq n\}$ and $J_2 = \{j \mid j$ is even and $2 \leq j \leq n\}$. The search space is $[0, 1] \times [-1, 1]^{n-1}$.	
UF2 : $f_1(\overrightarrow{x}) = x_1 + \frac{2}{\mid J_1 \mid} \sum_{j \in J_1} y_j^2,$ $f_2(\overrightarrow{x}) = 1 - \sqrt{x_1} + \frac{2}{\mid J2 \mid} \sum_{j \in J_2} y_j^2$ where $J_1 = \{j \mid j$ is odd and $(2 \leq j \leq n\}$ and $J_2 = \{j \mid j$ is even and $2 \leq j \leq n\}$ and $y_j = \begin{cases} x_j - [0.3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1] \cos(6\pi x_1 + \frac{j\pi}{n}) j \in J_1 \\ x_j - [0.3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1] \sin(6\pi x_1 + \frac{j\pi}{n}) j \in J_2 \end{cases}$ The search space is $[0, 1] \times [-1, 1]^{n-1}$.	
UF3 : $f_1(\overrightarrow{x}) = x_1 + \frac{2}{\mid J_1 \mid}(4 \sum_{j \in J_1} y_j^2 - 2 \prod_{j \in J_1} \cos(\frac{20y_j\pi}{\sqrt{j}}) + 2),$ $f_2(\overrightarrow{x}) = 1 - \sqrt{x_1} + \frac{2}{\mid J_2 \mid}(4 \sum_{j \in J_2} y_j^2 - 2 \prod_{j \in J_2} \cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$ where J_1 and J_2 are the same as those of $UF1$, and $y_j = x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})}, j = 2, ..., n,.$ The search space is $[0, 1]^n$.	

MOP	POF
$$f_1(\overrightarrow{x}) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h(y_j),$$ $$f_2(\overrightarrow{x}) = 1 - x_1^2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h(y_j)$$ UF4 : where $J_1 = \{j \lvert j$ is odd and $(2 \leq j \leq n\}$ and $J_2 = \{j \lvert j$ is even and $2 \leq j \leq n\}$ $y_i = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, ..., n$ and $h(t) = \frac{\lvert t\rvert}{1+e^{2\lvert t\rvert}}$. The search space is $[0, 1] \times [-2, 2]^{n-1}$.	
$$f_1(\overrightarrow{x}) = x_1 + (\frac{1}{2N} + \epsilon)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} h(y_j),$$ $$f_2(\overrightarrow{x}) = 1 - x_1 + (\frac{1}{2N} + \epsilon)\lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} h(y_j)$$ UF5 : where $J_1 = \{j \lvert j$ is odd and $(2 \leq j \leq n\}$ and $J_2 = \{j \lvert j$ is even and $2 \leq j \leq n\}$ $y_i = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, ..., n$ and $h(t) = 2t^2 - \cos(4\pi t) + 1$. The search space is $[0, 1] \times [-1, 1]^{n-1}$.	

MOP	POF
UF6 : $f_1(\overrightarrow{x}) = x_1 + \max\{0, 2(\frac{1}{2N} + \epsilon)\sin(2N\pi x_1)\} + \frac{2}{\lvert J_1\rvert}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$, $f_2(\overrightarrow{x}) = 1 - x_1 + \max\{0, 2(\frac{1}{2N} + \epsilon)\sin(2N\pi x_1)\} + \frac{2}{\lvert J_2\rvert}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$ where $J_1 = \{j \mid j \text{ is odd and } (2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$ $y_i = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, ..., n$. The search space is $[0,1] \times [-1,1]^{n-1}$.	
UF7 : $f_1(\overrightarrow{x}) = \sqrt[5]{x_1} + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1} y_j^2$, $f_2(\overrightarrow{x}) = 1 - \sqrt[5]{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2} y_j^2$ where $J_1 = \{j \mid j \text{ is odd and } (2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$ $y_i = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, ..., n$. The search space is $[0,1] \times [-1,1]^{n-1}$.	
UF8 : $f_1(\overrightarrow{x}) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$, $f_2(\overrightarrow{x}) = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$, $f_3(\overrightarrow{x}) = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$, where $J_1 = \{j \mid 3 \le j \le n$, and $j-1$ is a multiplication of $3\}$, $J_2 = \{j \mid 3 \le j \le n$, and $j-2$ is a multiplication of $3\}$, $J_3 = \{j \mid 3 \le j \le n$, and j is a multiplication of $3\}$. The search space is $[0,1]^2 \times [-2,2]^{n-2}$.	

MOP	POF
UF9 : $f_1(\overrightarrow{x}) = 0.5[\max\{0,(1+\epsilon)(1-4(2x_1-1)^2)\}+2x_1]x_2 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1+\frac{j\pi}{n}))^2$, $f_2(\overrightarrow{x}) = 0.5[\max\{0,(1+\epsilon)(1-4(2x_1-1)^2)\}-2x_1+2]x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1+\frac{j\pi}{n}))^2$, $f_3(\overrightarrow{x}) = 1-x_2+\frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1+\frac{j\pi}{n}))^2$ where $J_1=\{j\lvert 3\le j\le n$, and $j-1$ is a multiplication of $3\}$, $J_2=\{j\lvert 3\le j\le n$, and $j-2$ is a multiplication of $3\}$, $J_3=\{j\lvert 3\le j\le n$, and j is a multiplication of $3\}$ and $\epsilon=0.1$. The search space is $[0,1]^2\times[-2,2]^{n-2}$.	
UF10 : $f_1(\overrightarrow{x}) = \cos(0.5x_1\pi)\cos(0.5x_2\pi)+\frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}[4y_j^2-\cos(8\pi y_j)+1]$, $f_2(\overrightarrow{x}) = 0\cos(0.5x_1\pi)\sin(0.5x_2\pi)+\frac{2}{\lvert J_2\rvert}\sum_{j\in J_1}[4y_j^2-\cos(8\pi y_j)+1]$, $f_3(\overrightarrow{x}) = \sin(0.5x_2\pi)+\frac{2}{\lvert J_3\rvert}\sum_{j\in J_1}[4y_j^2-\cos(8\pi y_j)+1]$ where $J_1=\{j\lvert 3\le j\le n$, and $j-1$ is a multiplication of $3\}$, $J_2=\{j\lvert 3\le j\le n$, and $j-2$ is a multiplication of $3\}$, $J_3=\{j\lvert 3\le j\le n$, and j is a multiplication of $3\}$ and $y_j = x_j - 2x_2\sin(2\pi x_1+\frac{j\pi}{n}), j=3,\ldots,n$. The search space is $[0,1]^2\times[-2,2]^{n-2}$.	

BIBLIOGRAPHY

[1] Al Moubayed, N., Petrovski, A., and McCall, J. (2014). D2mopso: Mopso based on decomposition and dominance with archiving using crowding distance in objective and solution spaces. *Evolutionary computation*, 22(1):47–77.

[2] Alexandropoulos, S.-A. N., Aridas, C. K., Kotsiantis, S. B., and Vrahatis, M. N. (2019). Multi-objective evolutionary optimization algorithms for machine learning: A recent survey. In *Approximation and optimization*, pages 35–55. Springer.

[3] Alshomrani, S., Bawakid, A., Shim, S.-O., Fernández, A., and Herrera, F. (2015). A proposal for evolutionary fuzzy systems using feature weighting: dealing with overlapping in imbalanced datasets. *Knowledge-Based Systems*, 73:1–17.

[4] Antonio, L. M. and Coello, C. A. C. (2015). A non-cooperative game for faster convergence in cooperative coevolution for multi-objective optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 109–116. IEEE.

[5] Bach, M. and Werner, A. (2017). Cost-sensitive feature selection for class imbalance problem. In *International Conference on Information Systems Architecture and Technology*, pages 182–194. Springer.

[6] Bader, J. and Zitzler, E. (2011). Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19(1):45–76.

[7] Barandela, R., Sánchez, J. S., Garcıa, V., and Rangel, E. (2003). Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851.

[8] Barbosa, H. J. and Barreto, A. M. (2001). An interactive genetic algorithm with co-evolution of weights for multiobjective problems. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 203–210.

[9] Batista, G. E., Bazzan, A. L., Monard, M. C., et al. (2003). Balancing training data for automated annotation of keywords: a case study. In *WOB*, pages 10–18.

[10] Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29.

[11] Beume, N., Naujoks, B., and Emmerich, M. (2007). Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

[12] Borrajo, M. L., Baruque, B., Corchado, E., Bajo, J., and Corchado, J. M. (2011). Hybrid neural intelligent system to predict business failure in small-to-medium-size enterprises. *International journal of neural systems*, 21(04):277–296.

[13] Breiman, L. (1996). Bagging predictors machine learning 24 (2), 123-140 (1996) 10.1023. *A: 1018054314350*.

[14] Bui, L. T., Dinh, T. T. H., et al. (2018). A novel evolutionary multi-objective ensemble learning approach for forecasting currency exchange rates. *Data & Knowledge Engineering*, 114:40–66.

[15] Bui, L. T., Van Pham, B., Phan, V. A., et al. (2022). A multi-population coevolutionary approach for software defect prediction with

imbalanced data. In *2022 14th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–6. IEEE.

[16] Cai, X., Hu, M., Gong, D., Guo, Y.-n., Zhang, Y., Fan, Z., and Huang, Y. (2019). A decomposition-based coevolutionary multiobjective local search for combinatorial multiobjective optimization. *Swarm and Evolutionary Computation*, 49:178–193.

[17] Cateni, S., Colla, V., and Vannucci, M. (2014). A method for resampling imbalanced datasets in binary classification tasks for real-world problems. *Neurocomputing*, 135:32–41.

[18] Chandra, A. and Yao, X. (2004). Divace: Diverse and accurate ensemble learning algorithm. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 619–625. Springer.

[19] Chandra, A. and Yao, X. (2006). Multi-objective ensemble construction, learning and evolution. In *Proc. PPSN Workshop Multi-objective Problem Solving from Nature (Part 9th Int. Conf. Parallel Problem Solving from Nature: PPSN-IX)*, pages 9–13. Citeseer.

[20] Chandra, R., Frean, M., and Zhang, M. (2011). A memetic framework for cooperative coevolution of recurrent neural networks. In *The 2011 International Joint Conference on Neural Networks*, pages 673–680. IEEE.

[21] Chandra, R., Ong, Y.-S., and Goh, C.-K. (2018). Co-evolutionary multi-task learning for dynamic time series prediction. *Applied Soft Computing*, 70:576–589.

[22] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

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

[24] Coello, C. C. and Sierra, M. R. (2003). A coevolutionary multi-objective evolutionary algorithm. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 1, pages 482–489. IEEE.

[25] Coello Coello, C. A. (2017). Recent results and open problems in evolutionary multiobjective optimization. In *International Conference on Theory and Practice of Natural Computing*, pages 3–21. Springer.

[26] Coello Coello, C. A. and Reyes Sierra, M. (2004). A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *Mexican international conference on artificial intelligence*, pages 688–697. Springer.

[27] Darwin, C. (1871). R.(1859): On the origin of species by means of natural selection. *Murray. London.*

[28] Deb, K. (2011). Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer.

[29] Deb, K. and Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601.

[30] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[31] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*, pages 105–145. Springer.

[32] Denil, M. and Trappenberg, T. (2010). Overlap versus imbalance. In *Canadian conference on artificial intelligence*, pages 220–231. Springer.

[33] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

[34] Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164.

[35] Dubey, R., Zhou, J., Wang, Y., Thompson, P. M., Ye, J., Initiative, A. D. N., et al. (2014). Analysis of sampling techniques for imbalanced data: An n= 648 adni study. *NeuroImage*, 87:220–241.

[36] Durillo, J. J. and Nebro, A. J. (2011). jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771.

[37] E Silva, M. d. A. C., Coelho, L. d. S., and Lebensztajn, L. (2012). Multiobjective biogeography-based optimization based on predator-prey approach. *IEEE Transactions on Magnetics*, 48(2):951–954.

[38] Ehrlich, P. R. and Raven, P. H. (1964). Butterflies and plants: a study in coevolution. *Evolution*, pages 586–608.

[39] Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.

[40] Eriksson, R. and Olsson, B. (1998). Cooperativ6e coevolution in inventory control optimisation. In *Artificial Neural Nets and Genetic Algorithms*, pages 583–587. Springer.

[41] Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.

[42] Fernández, A., Carmona, C. J., Jose del Jesus, M., and Herrera, F. (2017). A pareto-based ensemble with feature and instance selection for learning from multi-class imbalanced datasets. *International Journal of neural systems*, 27(06):1750028.

[43] Ficici, S. G. and Pollack, J. B. (1998). Challenges in coevolutionary learning: Arms-race dynamics. In *Artificial Life VI: Proceedings of the sixth international conference on artificial life*, volume 6, page 238. MIT Press.

[44] Fogarty, J., Baker, R. S., and Hudson, S. E. (2005). Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction. In *Proceedings of Graphics Interface 2005*, pages 129–136.

[45] Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A. G., Parizeau, M., and Gagné, C. (2012). Deap: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, 13(1):2171–2175.

[46] Frank, A. (2010). Uci machine learning repository. *http://archive. ics. uci. edu/ml*.

[47] Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer.

[48] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2011). A review on ensembles for the class imbalance prob-

lem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

[49] García, S., Fernández, A., and Herrera, F. (2009). Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. *Applied soft computing*, 9(4):1304–1314.

[50] García-Pedrajas, N. and Cerruela-García, G. (2021). Cooperative coevolutionary instance selection for multilabel problems. *Knowledge-Based Systems*, 234:107569.

[51] García-Pedrajas, N., Hervás-Martínez, C., and Ortiz-Boyer, D. (2005). Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE transactions on evolutionary computation*, 9(3):271–302.

[52] Goh, C. K., Tan, K. C., Liu, D., and Chiam, S. C. (2010). A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research*, 202(1):42–54.

[53] Grimme, C. and Schmitt, K. (2006). Inside a predator-prey model for multi-objective optimization: A second study. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 707–714.

[54] Heywood, M. I. (2015). Evolutionary model building under streaming data for classification tasks: opportunities and challenges. *Genetic Programming and Evolvable Machines*, 16(3):283–326.

[55] Hillis, W. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. g. langton, c. taylor, jd farmer, s.

rasmussen, eds. *Artificial Life II (Santa Fe Institute Studies in the Sciences of Complexity, vol. 10. Addison-Wesley, Reading, MA.*

[56] Huband, S., Barone, L., While, L., and Hingston, P. (2005). A scalable multi-objective test problem toolkit. In *International conference on evolutionary multi-criterion optimization*, pages 280–295. Springer.

[57] Huband, S., Hingston, P., Barone, L., and While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506.

[58] Isaac, A., Nehemiah, H. K., Dunston, S. D., Christo, V. E., and Kannan, A. (2022). Feature selection using competitive coevolution of bio-inspired algorithms for the diagnosis of pulmonary emphysema. *Biomedical Signal Processing and Control*, 72:103340.

[59] Khoshgoftaar, T., Seiffert, C., and Van Hulse, J. (2008). Hybrid sampling for imbalanced data. In *Proceedings of IRI*, volume 8, pages 202–207.

[60] Klijn, D. and Eiben, A. (2021). A coevolutionary approach to deep multi-agent reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 283–284.

[61] Kong, X., Yang, Y., Lv, Z., Zhao, J., and Fu, R. (2023). A dynamic dual-population co-evolution multi-objective evolutionary algorithm for constrained multi-objective optimization problems. *Applied Soft Computing*, page 110311.

[62] Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207.

[63] Li, H., He, F., Chen, Y., and Pan, Y. (2021). Mlfs-ccde: multi-objective large-scale feature selection by cooperative coevolutionary differential evolution. *Memetic Computing*, 13(1):1–18.

[64] Li, H. and Zhang, Q. (2008). Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE transactions on evolutionary computation*, 13(2):284–302.

[65] Li, K., Chen, R., Fu, G., and Yao, X. (2018). Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315.

[66] Li, K., Chen, R., and Yao, X. (2023). A data-driven evolutionary transfer optimization for expensive problems in dynamic environments. *IEEE Transactions on Evolutionary Computation*.

[67] Li, K., Fialho, A., Kwong, S., and Zhang, Q. (2013). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130.

[68] Li, K., Kwong, S., Cao, J., Li, M., Zheng, J., and Shen, R. (2012). Achieving balance between proximity and diversity in multi-objective evolutionary algorithm. *Information Sciences*, 182(1):220–242.

[69] Li, K., Kwong, S., and Deb, K. (2015a). A dual-population paradigm for evolutionary multiobjective optimization. *Information Sciences*, 309:50–72.

[70] Li, K., Xiang, Z., Chen, T., and Tan, K. C. (2020). Bilo-cpdp: Bi-level programming for automated model discovery in cross-project defect prediction. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 573–584.

[71] Li, M., Yang, S., and Liu, X. (2015b). Pareto or non-pareto: Bi-criterion evolution in multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):645–665.

[72] Liang, J., Chen, G., Qu, B., Yue, C., Yu, K., and Qiao, K. (2021). Niche-based cooperative co-evolutionary ensemble neural network for classification. *Applied Soft Computing*, 113:107951.

[73] Liu, H.-L., Gu, F., and Zhang, Q. (2013). Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE transactions on evolutionary computation*, 18(3):450–455.

[74] Liu, M., Li, K., and Chen, T. (2020). Deepsqli: Deep semantic learning for testing sql injection. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 286–297.

[75] Liu, Y., Li, X., and Hao, Q. (2019). A new constrained multi-objective optimization problems algorithm based on group-sorting. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 221–222.

[76] Lohn, J. D., Kraus, W. F., and Haith, G. L. (2002). Comparing a coevolutionary genetic algorithm for multiobjective optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1157–1162. IEEE.

[77] López, V., Fernández, A., García, S., Palade, V., and Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141.

[78] Luke, S. (2013). *Essentials of metaheuristics*, volume 2. Lulu Raleigh.

[79] Ma, X., Li, X., Zhang, Q., Tang, K., Liang, Z., Xie, W., and Zhu, Z. (2018). A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441.

[80] Melo-Acosta, G. E., Duitama-Muñoz, F., and Arias-Londoño, J. D. (2022). An instance selection algorithm for big data in high imbalanced datasets based on lsh. *arXiv preprint arXiv:2210.04310*.

[81] Meyer, H. (1998). My enemy, my friend. *Journal of Business Strategy*, 19(5):42–47.

[82] Mienye, I. D. and Sun, Y. (2022). A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, 10:99129–99149.

[83] Millán-Giraldo, M., García, V., and Sánchez, J. (2013). Instance selection methods and resampling techniques for dissimilarity representation with imbalanced data sets. In *Pattern Recognition-Applications and Methods*, pages 149–160. Springer.

[84] Ming, F., Gong, W., Wang, L., and Gao, L. (2022). Balancing convergence and diversity in objective and decision spaces for multimodal multi-objective optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

[85] Mishra, S. (2017). Handling imbalanced data: Smote vs. random undersampling. *Int. Res. J. Eng. Technol*, 4(8):317–320.

[86] Moriarty, D. E. and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary computation*, 5(4):373–399.

[87] Moyano, J. M., Gibaja, E. L., Cios, K. J., and Ventura, S. (2020). Generating ensembles of multi-label classifiers using cooperative co-evolutionary algorithms. In *ECAI 2020*, pages 1379–1386. IOS Press.

[88] Mu, C., Jiao, L., Liu, Y., and Li, Y. (2015). Multiobjective non-dominated neighbor coevolutionary algorithm with elite population. *Soft Computing*, 19(5):1329–1349.

[89] Nguyen, M. H., Abbass, H. A., and McKay, R. I. (2007). Analysis of ccme: Coevolutionary dynamics, automatic problem decomposition, and regularization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(1):100–109.

[90] Ojha, V. K., Abraham, A., and Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60:97–116.

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

[92] Paredis, J. (1994). Steps towards co-evolutionary classification neural networks. In *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 102–108.

[93] Pollack, J. B. and Blair, A. D. (1998). Co-evolution in the successful learning of backgammon strategy. *Machine learning*, 32(3):225–240.

[94] Popovici, E., Bucci, A., Wiegand, R. P., and De Jong, E. D. (2012). Coevolutionary principles.

[95] Potter, M. A. and Jong, K. A. D. (1994). A cooperative coevolutionary approach to function optimization. In *International conference on parallel problem solving from nature*, pages 249–257. Springer.

[96] Potter, M. A. and Jong, K. A. D. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1):1–29.

[97] Press, W. H. and Dyson, F. J. (2012). Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26):10409–10413.

[98] Quinlan, J. R. (1993). Program for machine learning. *C4. 5*.

[99] Rashid, A., Ahmed, M., Sikos, L. F., and Haskell-Dowland, P. (2020). Cooperative co-evolution for feature selection in big data with random feature grouping. *Journal of Big Data*, 7(1):1–42.

[100] Ren, Y., Zhang, L., and Suganthan, P. N. (2016). Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational intelligence magazine*, 11(1):41–53.

[101] Rokach, L. (2016). Decision forest: Twenty years of research. *Information Fusion*, 27:111–125.

[102] Rosin, C. D. and Belew, R. K. (1996). A competitive approach to game learning. In *Proceedings of the ninth annual conference on Computational learning theory*, pages 292–302.

[103] Salzberg, S. L. (1994). C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993.

[104] Srinivas, N. and Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248.

[105] Stanley, K. O. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of artificial intelligence research*, 21:63–100.

[106] Tang, E. K., Suganthan, P. N., and Yao, X. (2006). An analysis of diversity measures. *Machine learning*, 65(1):247–271.

[107] Thu Bui, L. and Alam, S. (2008). *Multi-Objective Optimization in Computational Intelligence: Theory and Practice: Theory and Practice*. IGI global.

[108] Tomek, I. (1976). Two modifications of cnn.

[109] Van Truong, V., BUI, L. T., and NGUYEN, T. T. (2019a). An ensemble co-evolutionary based algorithm for classification problems. *Research and Development on Information and Communication Technology*, (1).

[110] Van Truong, V., Bui, L. T., and Nguyen, T. T. (2019b). A multi-objective cooperative coevolutionary approach for remote sensing image classification. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–5. IEEE.

[111] Van Truong, V., Thu, B. L., and Thanh, N. T. (2018). A coevolutionary approach for classification problems: Preliminary results. In *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 81–86. IEEE.

[112] Van Veldhuizen, D. A. (1999). Multiobjective evolutionary algorithms: classifications, analyses, and new innovations [ph. d. thesis]. *Department of Electrical and Computer Engineering. Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio*.

[113] Villar, J. R., González, S., Sedano, J., Chira, C., and Trejo-Gabriel-Galan, J. M. (2015). Improving human activity recognition and its application in early stroke diagnosis. *International journal of neural systems*, 25(04):1450036.

[114] Wang, C., Qin, F., Xiang, X., Jiang, H., and Zhang, X. (2023). A dual-population based co-evolutionary algorithm for capacitated electric vehicle routing problems. *IEEE Transactions on Transportation Electrification*.

[115] Wang, H., Jiao, L., and Yao, X. (2014). Two_arch2: An improved two-archive algorithm for many-objective optimization. *IEEE transactions on evolutionary computation*, 19(4):524–541.

[116] Wang, J., Zhang, W., and Zhang, J. (2015). Cooperative differential evolution with multiple populations for multiobjective optimization. *IEEE Transactions on Cybernetics*, 46(12):2848–2861.

[117] Wang, R., Ma, W., Tan, M., Wu, G., Wang, L., Gong, D., and Xiong, J. (2021). Preference-inspired coevolutionary algorithm with active diversity strategy for multi-objective multi-modal optimization. *Information Sciences*, 546:1148–1165.

[118] Wiegand, R. P. (2004). *An analysis of cooperative coevolutionary algorithms*. George Mason University.

[119] Williams, P. J. (2021). *Ensemble learning through cooperative evolutionary computation*. PhD thesis, University of Otago.

[120] Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421.

[121] Xie, D., Ding, L., Hu, Y., Wang, S., Xie, C., and Jiang, L. (2013). A multi-algorithm balancing convergence and diversity for multi-objective optimization. *J. Inf. Sci. Eng.*, 29(5):811–834.

[122] Xu, B., Gong, D., Zhang, Y., Yang, S., Wang, L., Fan, Z., and Zhang, Y. (2022). Cooperative co-evolutionary algorithm for multi-objective optimization problems with changing decision variables. *Information Sciences*, 607:278–296.

[123] Yang, X., Wang, Y., Byrne, R., Schneider, G., and Yang, S. (2019). Concepts of artificial intelligence for computer-assisted drug discovery. *Chemical reviews*, 119(18):10520–10594.

[124] Zăvoianu, A.-C., Lughofer, E., Amrhein, W., and Klement, E. P. (2013). Efficient multi-objective optimization using 2-population cooperative coevolution. In *International Conference on Computer Aided Systems Theory*, pages 251–258. Springer.

[125] Zhan, Z.-H., Li, J., Cao, J., Zhang, J., Chung, H. S.-H., and Shi, Y.-H. (2013). Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems. *IEEE transactions on cybernetics*, 43(2):445–463.

[126] Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W., Tiwari, S., et al. (2008). Multiobjective optimization test instances for the cec 2009 special session and competition. *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report*, 264:1–30.

[127] Zhao, Q. and Higuchi, T. (1996). Evolutionary learning of nearest-neighbor mlp. *IEEE Transactions on Neural Networks*, 7(3):762–767.

[128] Zhao, W., Alam, S., and Abbass, H. A. (2014). Mocca-ii: A multi-objective co-operative co-evolutionary algorithm. *Applied Soft Computing*, 23:407–416.

[129] Zhipeng, J. and Chao, L. (2019). Financial time series forecasting based on characterized candlestick and the support vector classification with cooperative coevolution. *Journal of Computers*, 14:195–209.

[130] Zhou, A., Zhang, Q., and Zhang, G. (2012). A multiobjective evolutionary algorithm based on decomposition and probability model. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

[131] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195.

[132] Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. In *International conference on parallel problem solving from nature*, pages 832–842. Springer.

[133] Zitzler, E., Laumanns, M., and Thiele, L. (2002). Spea2: Improving the strength pareto evolutionary algorithm.–evolutionary methods for design. *Optimization and Control with Applications to Industrial Problems, Athens, Greece*, pages 95–100.

[134] Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271.