Enhancing Meta-heuristic Algorithms Using Center-based Sampling at Population Level

by

Rasa Khosrowshahli

A thesis submitted to the
School of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Applied Science in **Computer and Electrical Engineering**

Faculty of Engineering and Applied Sciences
Ontario Tech University
Oshawa, Ontario, Canada
November 2023

CERTIFICATE OF APPROVAL

Submitted by Rasa Khosrowshahli

In partial fulfillment of the requirements for the degree of

MASc, Electrical and Computer Engineering

Date of Defence: October 4th, 2023

Thesis title:
Enhancing Meta-heuristic Algorithms Using Center-based Sampling at Population Level

The undersigned certify that the student has presented their thesis, that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate through an oral examination. They recommend this thesis to the School of Graduate and Postdoctoral Studies for acceptance.

Examining Committee:

[illegible] Dr. Khalid Elgazzar
Chair of Examining Committee

[illegible] Dr. Masoud Makrehchi
Research Supervisor

[illegible] Dr. Shahryar Rahnamayan
Co-Research Supervisor

Akramul Azim Dr. Akramul Azim
Examining Committee Member

[illegible] Dr. Mohamed El-Darieby
Thesis Examiner, Ontario Tech University

☐ As research supervisor for the above student, the thesis was rendered acceptable without revisions.

____________________ Dr. Masoud Makrehchi

✓ As research supervisor for the above student, I read and approved the changes required by the final examiners and recommend the thesis for acceptance:

[illegible] Dr. Masoud Makrehchi

Abstract

In recent years, center-based sampling has demonstrated impressive results to enhance the efficiency and effectiveness of meta-heuristic algorithms. The strategy of center-based sampling can be utilized at either the operation and/or population levels. Despite the overall efficiency of the center-based sampling in population-based algorithms, utilization at the operation level requires customizing the strategy for a specific algorithm which degrades the scheme's generalization. This study proposes a center-based sampling at the population level, which is operation-independent and correspondingly can be embedded in any population-based optimization algorithm. In classic mutation and crossover operators, the number of parents involved is a few, causing ineffective exploration; however, the current proposed center-based sampling uses a multi-parent approach, which results in multiple center-based solutions. In this thesis, two proposed schemes, namely, 1) Clustering center-based sampling and 2) Average ranking center-based sampling, are applied to enhance population-based single- and multi-objective optimization algorithms, respectively, in order to enhance their exploration and exploitation capabilities. The conducted comprehensive center-based experiments are a novel strategy to enhance population-based mechanistic algorithms. In order to assess the performance of proposed schemes, the proposed strategy is applied to single- and multi-objective optimization problems and experimented with CEC-2017 benchmark functions. The experimental outcomes confirm that the proposed clustering center-based and ranking center-based samplings have a crucial positive impact on convergence rate of various families of optimization algorithms.

Keywords: Center-based Sampling; Population-based Algorithms; Single-objective Optimization; Multi-objective Optimization; Meta-heuristic Algorithms;

Statement of Contributions

The primary contributions of this thesis are as follows:

1. Utilizing population-level center-based sampling schemes during the optimization.
2. Proposed clustering center-based sampling for the single-objective optimization algorithms and average ranking center-based sampling for the multi-objective optimization algorithms.
3. Providing a comprehensive investigation to show the superiority of proposed center-based sampling in single-objective algorithms, such cases as DE, GA, PSO, etc., and multi- and many-objective algorithms, such cases as GDE3, NSGA-II, MOPSO, etc.

The following papers have contributed to the research outcomes during the Master's program:

- *Hiba, H., Rahnamayan, S., Bidgoli, A. A., Ibrahim, A., and Khosrowshahli, R. "A comprehensive investigation on novel center-based sampling for large-scale global optimization." Swarm and Evolutionary Computation 73 (2022): 101105 [1].*
- *Khosrowshahli, R., Rahnamayan, S., and Bidgoli, A. A., "Clustering Center-based Differential Evolution." 2022 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2022 [2].*
- *Khosrowshahli, R., Rahnamayan, S., Ibrahim, A., and Makrehchi, M., "Ranking Center-based NSGA-II." 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2023 [3].*
- *Khosrowshahli, R., and Rahnamayan, S., "Block Differential Evolution." 2023 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2023 [4].*
- *Khosrowshahli, R., Rahnamayan, S., Bidgoli, A. A., and Makrehchi, M., "Self-Supervised Learning Using Noisy-Latent Augmentation." 2023 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2023 [5].*

Author's Declaration

Acknowledgements

I would like to thank my research supervisors, Dr. Shahryar Rahnamayan and Dr. Masoud Makrehchi, for their continuing contributions to support, mentorship, and guidance that helped me throughout my education at Ontario Tech University and for their work within the graduate student community. Additionally, I would like to acknowledge Dr. Azam Asilian Bidgoli and Dr. Amin Ibrahim, whose research in meta-heuristic algorithms and center-based sampling was a helpful resource for this work. Finally, I must express my profound gratitude to my parents and family for providing me with unfailing support and continuous encouragement throughout my study journey. I hope this accomplishment will make them feel proud.

Contents

List of Tables

List of Figures

Abbreviations and Symbols

ABC Artificial Bee Colony.

CC Clustering Center-based.

CEC Congress of Evolutionary Computation.

CMA-ES Covariance Matrix Adaptation Evolution Strategy.

DE Differential Evolution.

GA Genetic Algorithm.

GDE Generalized Differential Evolution.

IGD Inverted Generational Distance.

MaF Many-objective Function.

MaOP Many-objective Objective Problems.

MOEA/D Multi-objective Evolutionary Algorithm based on Decomposition.

NC Number of Clusters.

NFE Number of Function Evaluation.

NP Number of Population.

NSGA Non-dominated Sorting Genetic Algorithm.

PSO Particle Swarm Optimization.

RC Ranking Center-based.

SPEA Strength Pareto Evolutionary Algorithm.

STD Standard Deviation.

Chapter 1

Introduction

Many real-world problems can be modeled and solved as optimization problems using robust algorithms. Therefore, the development of effective and efficient algorithms is essentially necessary for solving these problems. For the last decade, meta-heuristic algorithms such as Genetic Algorithms (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) have shown their superiority in solving a variety of real-world large-scale optimization problems in engineering and science fields. Despite the capabilities of these algorithms, they have some weaknesses; for instance, the computational time of Evolutionary Algorithms (EA) refers to the fact that EAs can be computationally expensive, especially when dealing with complex and high-dimensional optimization problems. On the other hand, the gradient-based approaches are single-solution-based algorithms that are prone to fall into local optima, resulting in sub-optimal solutions. In order to tackle these issues, researchers used population-based evolutionary algorithms to utilize some stochastic operators that can be employed to solve black-box real-world problems. These algorithms aim to search globally into the search space and find the desirable solution(s).

The center-based sampling strategy was first proposed by Rahnamayan and Wang in 2009 [6]. They discovered a strange superiority of a golden region for a center-based sampling over high-dimensional search spaces. For the first time, the uniform random sampling strategy was seriously questioned for an efficiently distributed sampling strategy for the optimization of black-box problems. The center-based sampling has been a leading idea for several studies to improve the performance of population-based and single-solution-based meta-heuristic algorithms in various aspects. So far, researchers have utilized this strategy in their studies on the DE algorithm and its variants.

1.1 Motivation

Recently, many researchers have focused on improving population-based operators such as mutation and crossover. However, the modification of operators makes their proposed scheme a tailored one, which is not applicable to other similar algorithms since they are operation-oriented. To avoid this universality limitation, a novel technique, center-based sampling, which can be applied to various population-based algorithms, is proposed in this thesis. Center-based sampling theory has been introduced and investigated in its general form using mathematical methods and Monte-Carlo simulation by Rahnamayan and Wang [6], and they investigated the likelihood of closeness to the center of a couple of solutions in a black-box problem by using Monte-Carlo simulation. In comparison to randomly produced points, they found that the chance of points being closer to an unidentified answer is dramatically higher in the center of the search area.

Therefore, in this work, a new case of utilizing the center-based sampling strategy is proposed at population level to accelerate over the generations. Previously, a novel center-based differential evolution algorithm was proposed in the same direction [7]. They generated a single center-based solution by using the top 10% of population members. Although there was only one center-based candidate solution in the population, they showed that it can enhance the performance of the parent algorithm. Frequently, the center-based solution was better than the best solution in the population. Since this proposed scheme showed effectiveness, it motivated us to inject more center-based solutions in the current population of some population-based algorithms, such as DE, GA, PSO, and their variants. Furthermore, the extended multi-objective version of the scheme is proposed in this thesis. In the literature, a center-based sampling scheme that uses a clustering algorithm to cluster population during optimization is the first attempt to boost the acceleration of population-based algorithms; however, researchers have utilized other center-based sampling schemes for different stages such as population initialization and mutation where they are specially designed to solve large-scale optimization problems.

The main goals of this thesis are as follows:

1. Detailed investigation on properties of problems and population-based algorithms.

2. Detailed study on the backbone of center-based strategy.

3. Proposing center-based sampling scheme with clustering algorithm for population-based algorithms, for single-, multi-, and many-objective optimization.

4. Conducting comprehensive case studies in single-, multi-, and many-objective optimization algorithms selected from various algorithm families for the proposed center-based sampling strategy.

5. Investigating parameter settings sensitivity for the proposed algorithm.

The study focuses on ”how” and ”when” to utilize population-level-based center-based sampling on single- and multi-objective optimization algorithms.

1.2 Objectives

The thesis aims to utilize the center-based sampling concept to enhance population-based single-, multi-, and many-objective optimization algorithms. In this direction, the proposed schemes would be at the population level, which means they should be search component/operation independent. Further, working on decision space is hard as it can be increased exponentially, and the objective of this thesis is to avoid clustering in large-scale dimensions and do it on objective space, which is limited. To this end, this method would be applicable to any large-scale dimensional problem with single or many objectives.

1.3 Thesis Outline

This thesis consists of five chapters and appendices, which are organized as follows:

Chapter 2 presents a background review that has relevance to the research, including the concept of center-based sampling and enhanced optimization algorithms by the center-based sampling strategies in various novel ways. In addition, this chapter provides an introduction and background review of single-objective population-based algorithms such as the Differential Evolution (DE), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithms. In addition, this chapter provides an introduction to the multi-objective population-based algorithms, namely, Non-dominated Sorting Genetic Algorithm II (NSGA-II) and III (NSGA-III), Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D), Strength Pareto Evolutionary Algorithm 2 (SPEA2), Multi-objective Particle Swarm Optimization (MOPSO), and Generalized Differential Evolution 3 (GDE3).

Chapter 4 proposes a novel center-based sampling strategy for single-objective optimization algorithms and investigates its effectiveness through a series of experimental analyses.

Chapter 5 proposes a novel center-based sampling strategy for multi- and many-objective optimization algorithms and investigates its effectiveness through a series of comprehensive experimental analyses.

Chapter 6 outlines the contribution of the thesis and suggests future research directions. Finally, a specification of the benchmark functions used in the experiments is provided in the appendices.

Chapter 2

Background Review

2.1 Meta-heuristic Algorithms

Meta-heuristic algorithms are a class of optimization algorithms that are designed to find good solutions to complex optimization problems, particularly when traditional optimization techniques may not be effective or efficient. These algorithms are not problem-specific but general-purpose, meaning they can be applied to a wide range of optimization problems.

Extensive search is impracticable when the problem size grows exponentially in the search space [8]. Traditional optimization algorithms, such as greedy algorithms, necessitate a number of assumptions that are difficult to verify in many circumstances. In order to address these limitations, a set of more versatile and elastic algorithms, called meta-heuristic algorithms, has been proposed recently. Meta-heuristic algorithms are often inspired by natural phenomena, where a population of candidate solutions evolves towards finding an optimal solution through iterative processes. In the literature, meta-heuristic search algorithms with population-based frameworks have demonstrated adequate ability to solve high-dimensional optimization problems, including Genetic Algorithm (GA) [9], Ant Colony Optimization (ACO) [10], Particle Swarm Optimization (PSO) [11], Artificial Bee Colony (ABC) [12], Differential Evolution (DE) [13], Imperialist Competitive Algorithm (ICA) [14], and Gravitational Search Algorithm (GSA) [15].

2.2 Center-based Sampling

Sampling candidate solutions over the problem landscape is a common strategy in optimization and search algorithms, particularly in the context of meta-heuristic algorithms. This process involves generating a set of potential solutions to an optimization problem from different regions or points in the solution space. In other words, sampling is a fundamental component of search and optimization algorithms that guarantees the search process is not limited to a specific area but rather covers a wider range of possibilities. Finding good or optimal solutions requires exploring many regions of the solution space, which is especially important in high-dimensional and difficult optimization issues.

Rahnamayan and Wang [6] proposed a center-based sampling strategy to investigate the probability of closeness to an unknown solution for a center point and a uniform random point. By using Monte-Carlo simulation, they measured the Euclidean distances of the points to the unknown solution for dimensions $D = 1, 2, ...,$ and 1000 [16]. On the hypercube diagonal, they generated uniform distributed points with 10^{-3} step-size to compute the probability of closeness to an unknown optimum solution in a black-box problem. For each dimension $D_i = 1, 2, ..., D$, they measured the Euclidean distance of the fixed-point x and the generated uniform random point called a solution. After updating the acceptable distance variable based on the smallest distance value for calculating the probability of closeness and the average distance, they discovered an increment in the probability of closer points to an unknown solution towards the center of the search space, but as an opposite way, when it is compared to a uniformly generated random point, this probability decreases.

To formulate the center of interval $[a, b]$ for $D = 1$ dimension , the following equation

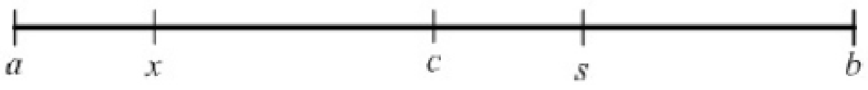


Figure 2.1: The visualization of uniform-random point x and the unknown solution s in the interval $[a, b]$ where c indicates corresponding the center of search space as $c = \frac{(a+b)}{2}$ on dimension $D = 1$.

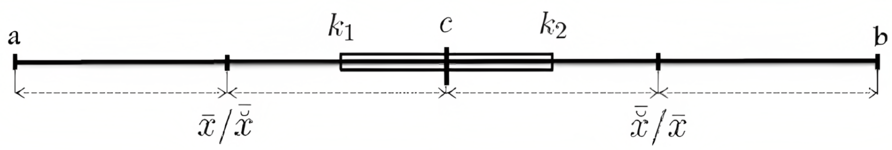


Figure 2.2: The visualization of a random point is closer to the unknown solution than x and $\hat{x}$. The k_1 and k_2 are the centers of intervals $[x/\hat{x}, r]$ and $[r, x/\hat{x}$.

is used:

For $D = 1$:

$$c = \frac{(a+b)}{2} \tag{2.1}$$

And, Figure 2.1 simulates the location of points in dimension $D = 1$ space.

For $D = n$:

$$c_i = \frac{(a_i + b_i)}{2} \tag{2.2}$$

Where $i = 1, ..., n$ and D is the dimension of the problem's search space.

Their simulation findings showed that the chance of the center point being near an unknown solution grows significantly as the dimension increases. The fascinating fact is that the likelihood of being close to the answer grows as the problem's size increases, approaching virtually one for the higher dimensions as demonstrated in Figure 2.3.

Rahnamayan et al. [17] provided an understandable explanation of why the opposite of a candidate solution based on the search space's center outperforms a random point in terms of proximity to an unidentified solution. They considered the interval for one-dimensional space, which is bounded by $[a, b]$ and has a center point c, as seen in Figure **??**. By assuming the random solution $x \in [a, c]$, and its opposite solution $\hat{x} \in [c, b]$ the average values of x and the opposite, $\hat{x}$ are located in the center of the sub-intervals $[a, c]$

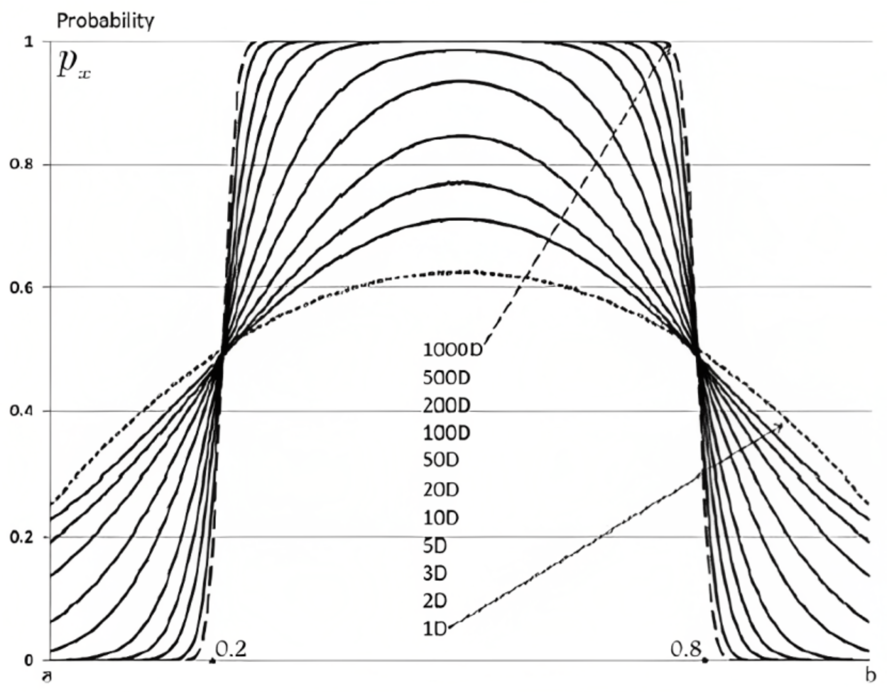
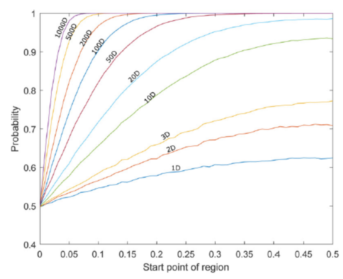


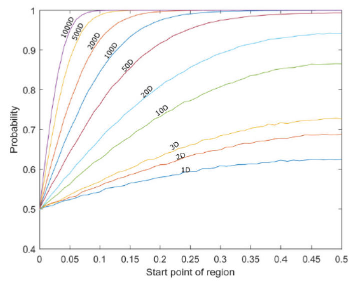
Figure 2.3: The graphs of Monte-Carlo simulations which present the probability of closeness of candidate-solution to an unknown solution in the interval [a,b], for different dimensions [6]

and $[c, b]$, respectively. The average of both random guesses will be located at $\hat{r} = c$. As a result of considering these mean values, they confirmed that a uniformly distributed solution $s \in [a, b]$ will (on average) be closer to the independent randomly generated points within the region $[k_1, k_2]$, where k_1 and k_2 are the centers of intervals $x/\hat{x}, r$ and $[r, \hat{x}/x]$. In fact, they explained the power of opposition-based searching resulted from the power of center-based sampling.

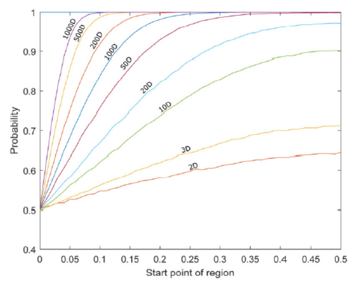
Regardless of the type of distance measurement, the probability of closeness of a randomly selected candidate solution in the shrunk region to an unknown solution increases compared to a candidate solution in the whole space. For instance, Fig. 2.4a a shows the probability of dimension 50 reaching one from the starting point 0.30 of the region for the Euclidean distance. On the other hand, the probability of dimension 50 reaches one from the starting point 0.52 and 0.36 of the region for Manhattan distance 2.4b and Cosine dissimilarity Figure 2.4c; respectively. Therefore, the probability of the Euclidean



(a) Euclidean distance



(b) Manhattan distance



(c) Cosine distance

Figure 2.4: The graphs of Monte-Carlo simulations present the probability of closeness of candidate-solution in a region around the center with different sizes to an unknown solution and three various distance measures.

distance rises faster in comparison with the Manhattan distance and Cosine dissimilarity [1].

The previously mentioned cases can be extended to worse cases when the solution is located: 1) on the border or 2) on the corner of the search space. They realized that the probability of center-point closeness to the solution on the corner of the search space can be increased exponentially as the dimensions increase linearly. This means that centroids are beneficial to be included when the size of the search space is immense. Further to this theory, they provided a Monte-Carlo simulation which shows the probability of candidate solution closeness to centroid region with different sizes to the solution that is located on the corner in Figure 2.5.

2.2.1 Center-based Sampling in Mutation Operation

Many papers have recently concentrated on improving the mutation and crossover operations of evolutionary algorithms. Center-based sampling is a novel technique that can be utilized in various evolutionary algorithms (e.g., GA, DE, PSO, etc). Tsutsui and Ghosh [18] proposed a center of mass crossover operator (CMX) in 1998. The study investigates the impact of multi-parent recombination in real-coded genetic algorithms. Real-coded genetic algorithms are optimization methods that use real-valued variables to represent candidate solutions. To produce new offspring, genetic algorithms often use

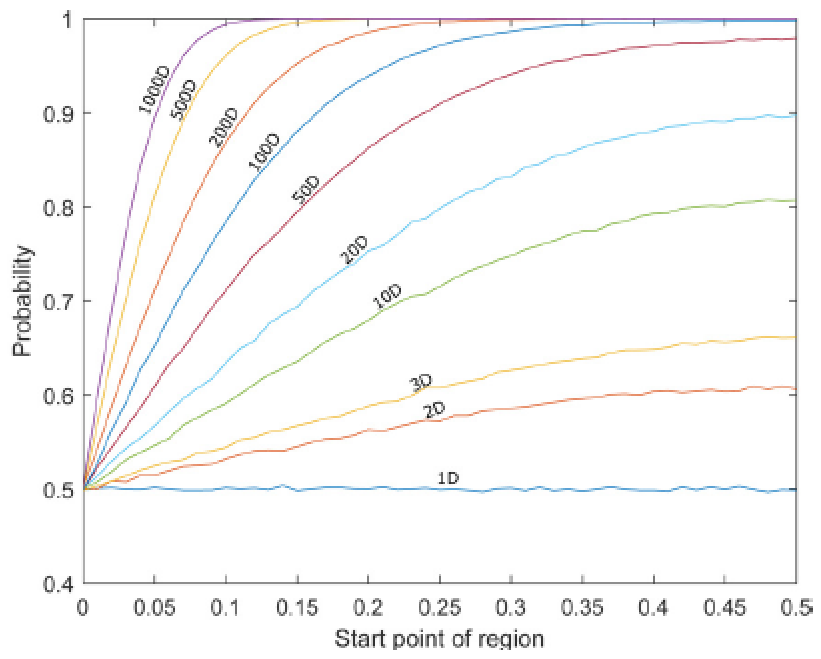


Figure 2.5: The graphs of Monte-Carlo simulations present the probability of closeness of candidate-solution in a region around the center with different sizes to the worst-case solution in the corner, in different dimensions.

two-parent crossover operators. However, this study introduces and explores multi-parent recombination operators, which involve more than two parents in the breeding process. In another study, Fan et al. created a new mutation operation which is Trigonometric Mutation Operation (TMO) algorithm. The base vector in the mutation operator is the center point of the geometric triangle. Thus, the mutation operation is defined as:

$$\begin{aligned} V_{i,G+1} = &\left(\frac{x_{r1,G} + x_{r2,G} + x_{r3,G}}{3}\right) + \\ &(p_2 - p_1) \cdot (x_{r1,G} - x_{r2,G}) + \\ &(p_3 - p_2) \cdot (x_{r2,G} - x_{r3,G}) + \\ &(p_1 - p_3) \cdot (x_{r3,G} - x_{r1,G}) \end{aligned} \tag{2.3}$$

where $r_1 \neq r_2 \neq r_3 \neq i$ and the parameters p_1, p_2, p_3 are defined as follows:

$$
\begin{aligned}
p_1 &= \frac{|f(x_{r_1}, G)|}{p^{'}} \\
p_2 &= \frac{|f(x_{r_2}, G)|}{p^{'}} \\
p_3 &= \frac{|f(x_{r_3}, G)|}{p^{'}} \\
p^{'} &= |f(x_{r_1}, G)| + |f(x_{r_1}, G)| + |f(x_{r_1}, G)|
\end{aligned} \tag{2.4}
$$

In another study, Sharma et al. proposed a geometric centroid-based mutation for the Shuffled Frog-Leaping Algorithm (CM-SFLA) [19]. With a probability parameter of CM, this technique uses a geometric centroid mutation (GCM). If the probability C_M becomes greater than the random number in the range of 0 and 1, the basic frog position is calculated; otherwise, the new frog position is calculated by the following equation:

$$D_{i,G} = \left(\frac{X_{\min,G} + X_{\mathrm{r1},G} + X_{r2,G}}{3}\right) + \mathrm{rand}(0,1) \cdot (X_{b,G} - X_{w,G}) \tag{2.5}$$

where $X_{r_1,G}$, $X_{r_2,G}$ are random frogs distinct and different from best and current frogs, and $X_{min,G}$ is the best frog with respect to fitness value.

The DE algorithm is a successful evolutionary algorithm used to solve large-scale optimization problems. In recent years, a number of substantial improvements to the DE have been put forth, one of which is a variation with modified main operators (i.e., mutation and crossover). Despite the success of center-based sampling, the mutation operator in DE was enhanced in terms of convergence rate and solution accuracy. In 2017, Hanan et al. [20] proposed a mutation scheme by replacing a randomly selected solution as the base vector with the center of three randomly selected solutions. First, the average of three randomly selected solutions for each dimension j are computed as follows:

$$\mu = \frac{x_{i_1} + x_{i_2} + x_{i_3}}{3} \tag{2.6}$$

Then, the standard deviation σ for each dimension j is computed as follows:

$$\sigma = \frac{max_j - min_j}{6} \tag{2.7}$$

where max_j and min_j are maximum and minimum values in the current population for dimension j.

The following equation shows the proposed center-based mutation function:

$$v_i = \mathcal{N}(\mu, \sigma) + F \cdot (x_{i_4} - x_{i_5}) \tag{2.8}$$

where $\mathcal{N}(\mu, \sigma)$ shows the sampled solution from normal distribution by μ and σ variables.

They showed impressive results against the classic DE on CEC-2013 Large-Scale Optimization Problems (LSGO) for dimension D=1000 by winning all 15 functions.

Later, in 2009, Hanan et al. [21] proposed five dynamic versions of the previously proposed center-based mutation scheme. The common proposed idea is dynamically changing the size of participants in the generation of center-based base vector over the optimization. Based on the provided results on CEC-2013 LSGO benchmark functions, the best scheme is Scheme 1, winning 14 out of 15 functions, which is described by applying 100% of the population at the beginning of optimization to be decreased gradually in the size of participants from the population based on the following equation:

$$DEcpop = NP - round(\frac{iter}{MAX_{NFC}} \times NP) \tag{2.9}$$

where *DEcpop* is the size of randomly selected population members for center-based base vector computation.

In addition, Hanan et al. [22] tried utilization of center-based base vector in the Success-History Based parameter Adaptation Differential Evolution (SHADE) algorithm, which is the winning scheme in the CEC-2013 LSGO competition. They evaluated their proposed Center-based SHADE (CSHADE), resulting in 15 out of 20 wins on CEC-2010 and 10 out of 15 wins on CEC-2013 benchmark functions against the SHADE algorithm for $D = 1000$ dimensions.

In the literature, the center-based sampling was primarily conducted in multi-objective optimization problems by Hanan et al. [23]. They proposed a new mutation operator based on center-based sampling for the Generalized Differential Evolution 3 (GDE3) algorithm to solve multi-objective problems. Their presented experimental results on IGD measurements for the comparisons between GDE3 and the proposed Center-based GDE3 (CGDE3) on the CEC-2017 MaF benchmark problems show an acceptable study in multi-objective problems. Specifically, when the dimensions in multi-objective optimization increase exponentially, the proposed center-based mutation scheme increases the convergence rate in a limited function call budget.

In summary, the center-based mutation utilized in population-based algorithms increases the opportunity to explore more promising regions in large-scale problems regardless of whether the objectives are single or multiple.

2.2.2 Center-based Sampling at Population Level

In this section, center-based sampling is investigated at the population level. The centroid-based population initialization for the DE algorithm's micro version was proposed by Salehinejad and Rahnamayan [24]. With this technique, they started the population inside a center percentile of the boundaries a and b. In Figure 2.6, a centroid region of selected boundaries in a 2D search space (light grey square) is highlighted by a dark grey square showing the reduced search space to enhance DE algorithm for large-scale problems. Two terms $\bar{\mathbf{x}}_d^{min}$ and $\bar{\mathbf{x}}_d^{max}$ represent the lower and upper boundaries of the centroid interval which are calculated as:

$$\bar{\mathbf{x}}_d^{min} = \bar{\mathbf{x}}_d^{min} + \frac{1-C}{2}(x_d^{max} - x_d^{min}) \tag{2.10}$$

and

$$\bar{\mathbf{x}}_d^{max} = \bar{\mathbf{x}}_d^{max} - \frac{1-C}{2}(x_d^{max} - x_d^{min}) \tag{2.11}$$

where C is the chosen centroid segment from the entire interval (0, 1). They declared a center-based population in a small size to enhance the performance of the classical DE algorithm for large-scale problems.

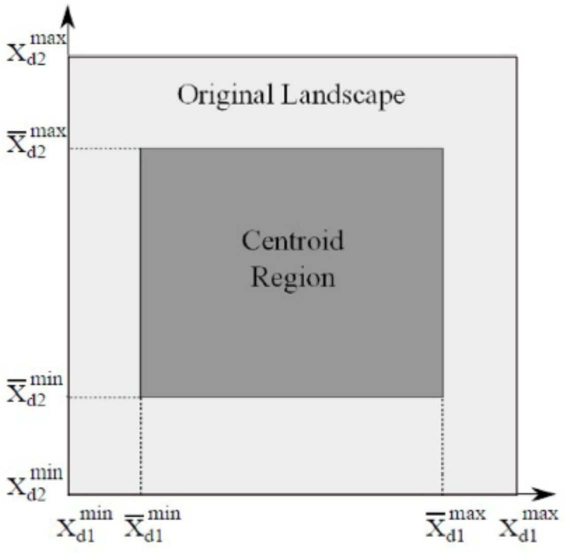


Figure 2.6: On a two-dimensional search space, centroid boundaries. D1 and D2 stand for dimensions. The initial boundaries of the dimensions are indicated by the light grey square that represents the original search space. The centroid region refers to the centroid bounds of the dimensions (dark grey square). [24]

Mousavirad et al. proposed a center-based Latin Hypercube Sampling (LHS) initialization of the population using Opposition-Based Learning (OBL) DE [7]; the aim was to tackle deceptive optimization problems. LHS is a statistical technique for producing random samples from a multivariate distribution. In order to combat landscape deception, this research combines a modified LHS and OBL with a differential evolution method. The outcomes attest to the proposed algorithm's higher performance than the classical DE.

In another similar work on PSO proposed by Mousavirad and Rahnamayan [25], a new component was added to the velocity operation, which is called the center of gravity factor. The new factor improves both exploration and exploitation at the same time since it is based on the center of some randomly chosen candidate solutions in the present population.

A hybridization technique on local search center-based sampling was applied to the DE algorithm by Xu et al. [26]. They calculated the centroid of top Q individuals from the sorted population of size NP as:

$$XC = \sum_{i=1}^{Q} \frac{X_i}{Q} \tag{2.12}$$

The difference between NP and Q was then calculated as $(NP - Q)$, which produced the new individuals. The initial population for the DE method was created by merging the $NP - Q$ and top Q individuals. This work motivated Khanum et al. [27] to propose a centroid population initialization in which $3 \times N_P$ (i.e., population size) number of individuals are generated randomly, and then a new population is created based on an average of three individuals as the initial population.

Furthermore, Mousavirad and Rahnamayan [28] proposed a method to employ center-based sampling at the population level of DE. They used the center of the entire population as a new individual to inject into the population at the end of each iteration. The experiments represent that the center-point could effectively direct the whole population towards better candidate solutions.

2.3 Single-objective Population-based Algorithms

2.3.1 Differential Evolution Algorithm

The Differential Evolution (DE) algorithm is regarded as an effective method for solving challenging optimization problems. In 1995, Price and Storn proposed DE as a robust stochastic population-based evolutionary algorithm in which its efficiency has been tested

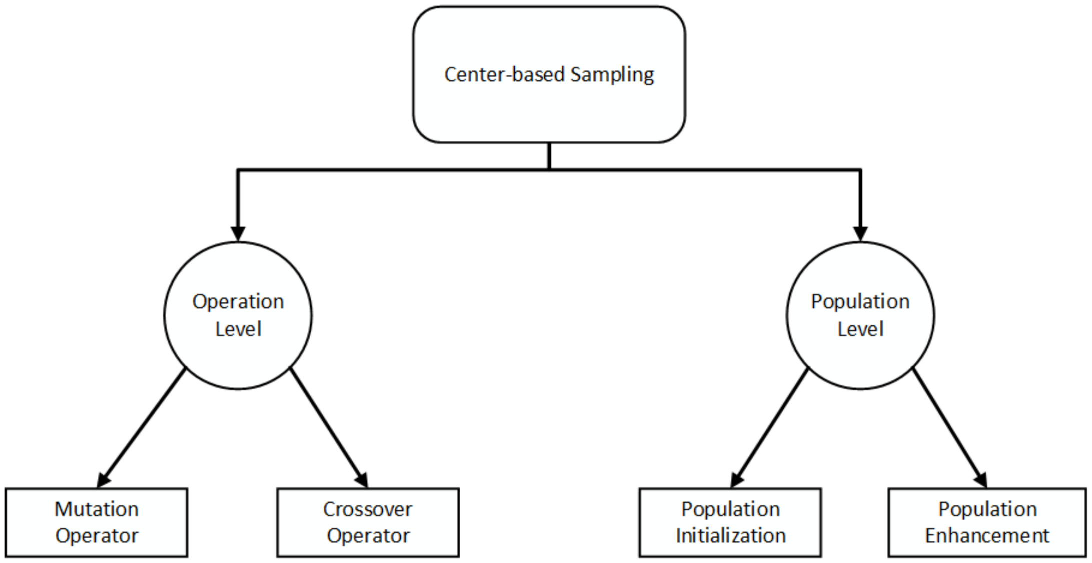


Figure 2.7: The flowchart shows how Center-based sampling has contributed to evolutionary algorithms in the literature.

on various large-scale and real-world problems [29]. Briefly, DE has three control parameters; population size NP, mutation scale factor F, and crossover rate CR, which directly impact the performance of the algorithm, and their optimal values are problem oriented. The initial stage of DE begins with uniform random population initialization. Following that, the main iterative optimization procedure starts with mutation and crossover operators to generate new candidate offspring solutions based on parents. Then, the greedy selection operator compares the parent and the candidate offspring solution to select the surviving individual for the next generation. After reaching the maximum number of function evaluations, DE returns the best solution found based on objective value.

2.3.2 Genetic Algorithm

Genetic algorithm (GA) is a meta-heuristic method inspired by natural evaluation theory commonly used to search for the optimal solution using an iterative process to discover the best solution. It falls under the broader category of evolutionary algorithms, which mimic the process of natural selection to search for optimal solutions. Originally introduced by John Holland [30] in the 1970s and further developed by other researchers, Genetic Algorithm has found applications in various fields, including engineering, computer science, finance, biology, and more [9].

The fundamental principles of Genetic Algorithm are rooted in the concepts of evolution, inheritance, and selection. The algorithm emulates the process of evolution through a population of potential solutions to a given problem. These solutions, often represented as individuals or chromosomes, are analogous to the genetic makeup of living organisms. In Genetic Algorithm, each potential solution to the problem is encoded as a chromosome composed of genes, which can take various forms depending on the nature of the problem. Common representations include binary strings, real-valued vectors, permutations, or any other suitable data structure. The choice of encoding has a significant impact on the algorithm's efficiency and the quality of the results.

The optimization process begins with the creation of an initial population. An initial population includes candidate solutions, which are represented as chromosomes. The individuals are initially generated using random permutations representing a potential order of features. A fitness function then evaluates the corresponding chromosomes to determine how well the solution meets specified conditions in the optimization problem. The better an individual's fitness value, the more favorable it is considered over the generation to be utilized as the parent. The core principle of natural selection, survival of the fittest, guides the selection process in Genetic Algorithm. Individuals with higher fitness values are more likely to be selected for reproduction, which increases their chances of passing their genetic genes (i.e., decision variables) to the next generation [31]. The parents are chosen by the "rank selection" or "tournament selection" method. Rank selection sorts the population according to fitness values and finds their ranks. Subsequently, according to the rank, each chromosome receives a selection probability, which is used to choose the parents. Pairs of selected individuals are chosen to create offspring by combining their genetic information. Crossover operators are applied to exchange genetic material between parents, producing one or more offspring. The goal is to create offspring with traits inherited from their parents. Some individuals in the new generation undergo random changes in their genetic information, mimicking genetic mutations in biological evolution. Mutation introduces diversity into the population and helps prevent convergence to suboptimal solutions.

Genetic Algorithms possess several advantages, including their ability to handle complex and multimodal problems, their adaptability to various problem domains, and their parallelizable nature, allowing for efficient implementations on modern computing architectures. The algorithm finds applications in numerous fields, such as parameter optimization, scheduling, engineering design, financial modeling, data mining, and machine learning, among others. It has been successfully employed in real-world scenarios where traditional optimization methods face challenges or are infeasible to be utilized.

2.3.3 Particle Swarm Algorithm

In contrast to evolutionary computation techniques, Kennedy and Eberhart [11] designed a novel algorithm by simulating social behavior called the Particle Swarm Optimization (PSO) algorithm that resembles a school of fish or flying birds in 1995. This method iteratively improves a population of candidate solutions (i.e., particles) by moving around the search space based on the particle's position and velocity. PSO is a gradient-free optimization algorithm, which gives the ability to search very large spaces and makes it robust not to be stuck in local optima. Particles are able to share information with each other by using an embedded mechanism called velocity. Each particle remembers its best-known position (i.e., Pbest) in its memory as well as the entire best-known particle position (i.e., Gbest) saved in the shared memory for all particles. The velocity of each particle is randomly initialized and updated based on two vectors, Pbest and Gbest, during optimization as follows:

$$v_{t+1}^{i} = \omega \times v_{t+1}^{i} + c_1 \times r_1(Pbest_t^{i} - x_t^{i}) + c_2 \times r_2(Gbest_t - x_t^{i}) \tag{2.13}$$

where t represents the current iteration, x_t^i is the position of i-th particle in the t-th iteration, $Pbest_t^i$ shows the Pbest for the i-th particle and $Gbest_t$ is the Gbest in the t-th iteration. The two terms, r_1 and r_2, are randomly generated vectors in the interval of [0, 1], and c_1 and c_2 are two learning coefficients which are called "self-cognitive" and "social-learning parameters", respectively. Another term to balance the global and local search is inertia weight ω proposed by Shi and Eberhart [32] to enhance the initial version of PSO in 1998.

A new position of particles is updated based on the updated velocity as follows:

$$x_{t+1}^i = x_t^i + v_{t+1}^i \tag{2.14}$$

Figure 2.8 illustrates updating particle positions based on mentioned equations.

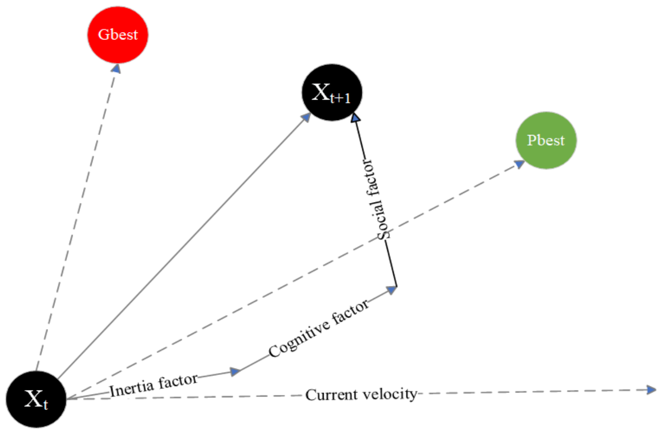


Figure 2.8: Vector-based view of particle position updating in PSO algorithm. The inertia factor is the first part to the right of Eq. 2.13, and the second and third part to the right of Eq. 2.14 is called cognitive factor and social factor, respectively.

2.3.4 Artificial Bee Colony Algorithm

A well-known meta-heuristic method called the Artificial Bee Colony (ABC) optimization algorithm was motivated by the foraging habits of honey bees. ABC belongs to swarm intelligence algorithms and was presented by Karaboga in 2005 [12].

ABC algorithm simulates the foraging behavior of bees, where bees search for food sources and transfer their findings to other bees within the same colony. The algorithm maintains a population of artificial bees, which represents potential solutions to the optimization problem. These artificial bees explore the search space and update their positions based on their fitness values.

The ABC consists of three main entities as follows: employed bees, onlooker bees, and scout bees. Firstly, a uniform randomly generated population of employed bees is created. Each employed bee explores a solution in the search space by exploiting the information from the current best solutions. Then, the employed bees communicate their findings to the onlooker bees. The onlooker bees are responsible for selecting a solution with respect to the probability proportional to its fitness value. After evaluating the selected solution, ABC decides to update its position if the fitness is improved. In another explanation, this process allows the algorithm to concentrate on promising regions in the search space. Lastly, if the neighborhood of food source has been explored enough, it is abandoned, meaning the bee exceeds a predefined number of iterations without improvement; it becomes a scout bee and explores new solutions random. Until the optimization reaches the maximum number of function calls, the ABC dynamically balances exploration and exploitation to efficiently search the search space.

2.3.5 Covariance Matrix Adaptation Evolution Strategy

A novel evolutionary algorithm optimization motivated by derandomized evolution strategy is Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [33, 34]. This algorithm optimizes an objective function by sampling λ candidate solutions from a multivariate normal distribution [35]. This strategy derives μ solutions from the λ solutions to adaptively predict the local covariance matrix of the objective function, resulting in a higher engagement chance of successful samples in further iterations. At iteration t, the CMA-ES algorithm generates new λ candidate solutions by adding a random Gaussian mutation defined by a covariance matrix C^t according to

$$x_k^{t+1} = \mathcal{N}(m^t, {\sigma^t}^2 C^t) = m^t + \sigma^t \cdot \mathcal{N}(0, C^t), \tag{2.15}$$

where m^t denotes the mean of sampled random normal distribution, σ^t is the mutation factor [36]. The λ samples are evaluated based on the objective function and ranked. In the next step, the weighted mean m is updated according to the following equation

$$m_i^{t+1} = m_i^t + \frac{1}{\lambda}\sum_{i=1}^{\lambda}(x_k^{t+1} - m_i^t), \tag{2.16}$$

and, the next generation of evolution route R_c is updated according to the following equation:

$$R_c^{t+1} = (1 - o_c)R_c^t + \frac{\sqrt{o_c(2 - o_c)\mu_{ef}}}{\sigma^t}(m^{t+1} - m^t), \tag{2.17}$$

where o_c is a weight coefficient in the range of [0,1], μ_{ef} is the variance efficient, and σ is the step size, and the covariance matrix C is updated as follows:

$$C^{t+1} = (1 - o_1 - o_\mu)C^t + o_1 R_c^{t+1}(R_c^{t+1})^T + o_\mu \sum_{i=1}^{\lambda} \omega_j y_j^{t+1}(y_j^{t+1})^T \tag{2.18}$$

where o_1 and o_μ are the covariane matrix learning rates in rank-1 and rank-μ, respectively. ω_i is the weight coefficient for individual x_i, and $y_j^{t+1} = \frac{(z_j^{t+1} - m^t)}{\sigma^t}$ is variation step size [37]. Furthermore, the evolution route R_σ and global step size σ are updated according to following equations:

$$R_\sigma^{t+1} = (1 - o_\sigma)R_\sigma^t + \frac{\sqrt{o_c(2 - o_c)\mu_{ef}}}{\sigma^t} \cdot (C^t)^{-\frac{1}{2}} \cdot (m^{t+1} - m^t) \tag{2.19}$$

$$\sigma^{t+1} = \sigma^t \cdot e^{\frac{o_\sigma \cdot R_{\sigma}{}^{t+1}}{d_\sigma \cdot E\mathcal{N}(0,I)} - \frac{o_\sigma}{d_\sigma}} \tag{2.20}$$

where o_σ is the learning rate of R_σ, d_σ is the damping parameter, and $E\mathcal{N}(0, I)$ is the Euclidean parametric expectation of a random vector that follows normal distribution $\mathcal{N}(0, I)$.

2.4 Multi-objective Population-based Algorithms

2.4.1 Non-dominated Sorting Genetic Algorithm II

The NSGA-II algorithm modified the Non-dominated Sorting Genetic Algorithm (NSGA) that is inspired by the GA algorithm to solve multi-objective problems. NSGA-II was introduced by Deb et al. [38] to address its predecessor, NSGA, limitation by a crowding distance sorting mechanism to maintain diversity and improve convergence. This mechanism ensures that candidate solutions from crowded regions are less likely to be selected. Similar to GA, NSGA-II has its own crossover and mutation operators. The simulated binary crossover and polynomial mutation generate new offspring, and then the tournament is used as the selection operation to choose the next surviving individuals for the next generation. The simulated binary crossover (SBX) combines the decision variables of two parent solutions to produce a new offspring solution using a probability ratio. SBX takes a single randomly selected point to crossover between each decision variable and generates two offspring solutions based on the distance between the parent solutions at that variable. With the help of probability distribution, generating the offspring solutions depends on a parameter called the distribution index, which controls the spread of the solutions. The effect of high distribution indexes is to have more diverse solutions from the parent solutions, and low distribution indexes can close the gap between offspring and parent solutions. In general, SBX is designed to make a balance in exploration and exploitation in search space and maintain diversity in the population. The polynomial mutation operator is like the mutation operator in GA, which is designed to perturb the decision variables with a small change to maintain the feasibility of the solution. Polynomial mutation works by randomly selecting and perturbing a decision variable based on a polynomial distribution which is controlled by a perturbation magnitude index. The mutation index can help optimization algorithms converge faster if it sets to a low value, whereas a high value can increase the effectiveness of exploration in the search space. Finally, the tournament selection is used to pick N individuals from the population. The tournament is meant to make a competition between two individuals based on the front rank. If two individuals share the same rank, the individual with the higher crowding distance is selected.

2.4.2 Non-dominated Sorting Genetic Algorithm III

In this section, an extension of the previous study, NSGA-II, is discussed for solving multi-objective optimization problems. Whereas NSGA-II was a successful algorithm for addressing multi-objective problems, it shows drawbacks when the number of objectives increases to 5 or higher. To address this issue, Deb and Jain proposed the third version of NSGA, known as NSGA-III, in 2014 [39]. The fundamental components, such as crossover and mutation operators of the NSGA-III, are similar to the NSGA-II algorithm; however, the selection operator improved and became more practical. After generating the new population P_t from the last population Q_t by crossover and mutation operators, the selection mechanism is invoked to find the best members from the combined population $R_t = P_t \cup Q_t$ based on non-dominated sorting algorithm. Considering S_t is the selected population by a non-dominated sorting algorithm. If the size of S_t is greater than the number of population NP, NSGA-III uses reference points that are used to select the Pareto front F_l. In the original NSGA-III, they studied Das and Dennis' [40] algorithm to generate these reference points.

The proposed selection mechanism modifies the last step in the selection operator in the NSGA-II, which finds the solutions that have the largest crowding distance to remove extra members. Primarily, the calculated objective values for each member are adaptively normalized based on members in S_t set. Secondly, each reference point on the hyper-plane is connected to other reference points to create reference lines. Thirdly, all population members of S_t and F_l are associated with a reference point whose reference line is shorter to a member than other lines. Then, the number of associated members that are associated with each reference point is counted. If any of the reference points remain non-associated with members, then the solution with the smallest perpendicular distance (Euclidean) in the normalized objective space survives. On the other hand, if all the reference points are associated with at least one of the members in F_l, the selection of remaining members is done randomly. Until the target population number is attained, this process is repeated.

2.4.3 Multi-objective Evolutionary Algorithm based on Decomposition

In this section, the MOEA/D algorithm is reviewed, which was proposed by Zhang and Li in 2007 [41]. Decomposition is a basic strategy that breaks down M-objective optimization problem into μ single-objective optimization sub-problems using a set of uniformly distributed weight vectors $W = \{W^1, ..., W^\mu\}$ and optimizes them simultaneously. The fundamental operators to generate a new child is similar to other evolutionary algorithms such as GA where a child is generated by crossing two parents and mutating if necessary. The selection is performed to replace the population members and offspring to find a new optimal set. A predefined scalarizing function g is calculated to give each solution a value. If an individual x_j is compared with child u_i according to g based on the weight vector w_j and the child u_i has better g, x_j is replaced by u_i. Although there are several scalarizing functions that were studied in MOEA/D, in this study the PBI function as g_{PBI} is investigated, which is defined as follows:

$$g_{PBI}(x|w, z^*) = d_1 + \theta d_2, \tag{2.21}$$

where d_1 equals to:

$$d_1 = \frac{\|(f(x) - z^*)^T w\|}{\|w\|}, \tag{2.22}$$

and d_2 equals:

$$d_2 = \|f(x) - (z^* + d_1 \frac{w}{\|w\|})\|, \tag{2.23}$$

The scalarizing function g_{PBI} should be minimized to find better solutions in terms of $z^* = (z_1^*, ..., z_M^*)^T$, which is the ideal point. Due to it being difficult to find the exact ideal point z^* of a given MOP, its approximated point, which consists of the minimal function value for each objective $fi(i \in \{1, ..., M\})$, determined throughout the search process, is typically utilized for the calculation of the scalarizing function g_{PBI}. In Eq. (2.22) and (2.23), the terms d_1 and d_2 refer to the perpendicular distance between the objective function vector f(x) and w, respectively. The product of d_1 and d_2 is the value of the PBI function determined by Equation (2.21). The balance between the population's diversity (d_2) and convergence (d_1) is controlled by the penalty parameter $\theta > 0$. A significant number highlights the value of population variety, whereas a lower value promotes convergence toward the PF [42].

2.4.4 Strength Pareto Evolutionary Algorithm 2

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) is a multi-objective optimization algorithm widely used to solve complex optimization problems with multiple conflicting objectives. It is an improved version of the original SPEA algorithm [43]. Like other MOO algorithms, the main goal of SPEA2 algorithm is to find a set of solutions that results in the Pareto-optimal front. In other words, it seeks to find a set of solutions that achieve a good trade-off between the different objectives. The main components of SPEA2 are motivated by genetic algorithms to generate new candidate solutions such as selection, crossover, and mutation. The selection process is based on the concept of "strength." The term strength measures how well a solution performs compared to other solutions in the population. If a solution gets stronger, there is a higher probability of being selected as a candidate for the next generation. By generating new solutions, SPEA2 needs to evaluate the quality of each solution by considering both the objective values and the density of the solution in the objective space. In other words, solutions with better objective values and a lower density of neighboring solutions are assigned higher quality scores. In addition, SPEA2 employs an archive mechanism to maintain diversity in the population which results in effective exploration in space. The archive maintains the algorithm's greatest discoveries made during the optimization phase by storing the non-dominated solutions so far discovered. By iteratively repeating the process of generating new solutions, evaluating their fitness, and updating the archive, SPEA2 gradually improves the quality of the solutions and converges towards the Pareto-optimal front. In conclusion, SPEA2 is a multi-objective optimization algorithm that uses fitness assignment, genetic operators, and an archive mechanism to identify a group of solutions that reflect the Pareto-optimal front. It gives a wide range of solutions for making decisions in complex optimization problems while balancing the trade-off between competing objectives.

2.4.5 Multi-objective Particle Swarm Optimization

The concept of MOPSO is similar to the PSO algorithm used in solving single-objective optimization problems. It works by initializing a population of particles, where each particle represents a potential solution in the problem space. These particles move through the search space by adjusting their positions and velocities based on their own experiences and the experiences of their neighboring particles. The velocity update is influenced by the personal best position (Pbest) of each particle and the best position (Gbest) among all particles in the population. Since the problem has several objectives where all need to be optimized simultaneously, Coello Coello and Lechuga [44] proposed the idea of Pareto dominance to decide a particle's flight path, and it preserves previously discovered non-dominated vectors in a global repository that is subsequently used by other particles to influence their own flight. Particles search the search space as the iterations go on and move closer to the Pareto front. Through the course of optimization, the algorithm strives to keep a diverse set of non-dominated solutions. This is accomplished by employing strategies like crowding distance, which assesses the density of solutions in the objective space and directs particle motion toward uncharted territory.

2.4.6 Generalized Differential Evolution 3

Generalized Differential Evolution 3 (GDE3) is an evolutionary algorithm used for global optimization problems. This algorithm is an extension of the popular Differential Evolution (DE) algorithm with M objectives and K constraints, which was designed by Kukkonen and Lampinen to solve multi-objective optimization problems in 2005 [23]. As GDE3 is extended from DE, the fundamental evolutionary components of the algorithm are similar. The selection strategy is similar to the previous version $GDE2$ except in the following: 1) Applying constraints amid the selection process; 2) None of the solutions can dominate the other. Therefore, the selection in GDE3 obeys the following conditions: The new vector is chosen if it dominates the old vector in the space of constraint violations, and the old vector is chosen if both are infeasible solutions. The feasible vector is chosen if only one of them is a viable option. The dominant vector is chosen for the following generation if both are practical. The non-dominating situation chooses both vectors. Exceptionally, the number of generated population exceeds the previous generation's population, which needs domination. In this case, a similar selection strategy to the NSGA-II algorithm is conducted using the non-dominated sorting algorithm and crowding distance measurement.

Chapter 3

Proposed Clustering Center-based Sampling for Single-objective Population-based Algorithms

In this section, first, the proposed Clustering Center-based Sampling for single-objective is presented and explained in detail. Secondly, the CEC-2017 benchmark problems are reviewed and provided with their parameter setting. Thirdly, five case studies of five meta-heuristic population-based algorithms are tested to enhance the comparing algorithm. In the first case, the clustering center-based DE algorithm is proposed to indicate the effectiveness and efficiency of the proposed center-based sampling at the population level. In the second case, enhancing the PSO algorithm, another successful population-based algorithm with the proposed center-based strategy is proposed. In the third case, the oldest and foremost version of population-based algorithms, the GA, is investigated. In the fourth case, clustering center-based in the ABC algorithm is examined. Finally, the combination of center-based sampling strategy and CMA-ES algorithm is examined for the first time.

3.1 Introduction

Single-objective population-based algorithms are widely recognized as effective optimization methodologies that have demonstrated success in tackling a range of practical optimization challenges. These algorithms predominantly function as meta-heuristic techniques designed to navigate intricate search spaces. Their principal objective revolves around the efficient exploration of the search space, aimed at identifying optimal solutions within predefined temporal constraints. The primary challenges associated with single-objective population-based algorithms encompass the tendency to prematurely converge towards local optima due to a depletion of population diversity during initial optimization stages **??**, coupled with their characteristic gradual convergence rate.

Real-world single-objective problems refer to practical optimization challenges that involve the quest for a single optimal solution within a given context. These problems arise in various domains, such as engineering, finance, logistics, and beyond [45, 46].

In general, the objective of a problem is to maximize or minimize a single measurable parameter, often representing a desired outcome or performance metric. The goal is to find the best configuration or arrangement of variables, parameters, or resources to achieve the desired outcome while considering real-world constraints, limitations, and complexities. Solving real-world single-objective problems entails identifying the most favorable solution from a multitude of possibilities, taking into account the intricacies and intricacies specific to the domain and problem at hand [47].

3.2 Proposed Clustering Center-based Sampling in Single-objective Population-based Algorithms

In this section, a center-based sampling mechanism is proposed for the population-based algorithms to improve the exploration and exploitation capabilities. The idea of center-based sampling is used to generate new individuals and correspondingly improve the quality of the population in each generation. In the direction of objective values, the algorithm finds a fixed number of clusters NC from the current population or recently generated offspring based on fitness values F. In other words, the new center-based solutions are the center of candidate solution variables X in clusters on D dimensions. This technique has a benefit on clustering since it can be extended into the entire population based on a single metric rather than the entire search space with D dimensions. Accordingly, the proposed sampling strategy divides the population into a constant number of smaller clusters where the distance of candidate solutions in a cluster is the shortest based on their fitness value. There are many clustering techniques that could find the samples closest to each other such as K-means algorithm; however, a much more basic strategy could be done, which is discussed as follows.

Taking into account there is a given parameter, NC, as the number of clusters (or center-based solutions) in a population with NP number of individuals, whereas NC

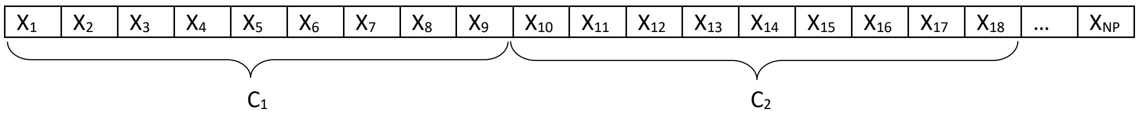


Figure 3.1: Illustrating an example of generating the clusters after sorting candidate solutions based on their fitness value in the population.

must be smaller than NP ($NC < NP$). The population is only generated by uniform randomly generated $NP - NC$ candidates in D search space dimensions. The next step is to augment the population to build NP by adding NC candidate solutions which are the gravity centers of the NC clusters. Before the clustering, the population is sorted based on their single fitness values in ascending order. The sorted order of individuals needs to be saved in the population, similar to the ranking selection operator in GA. In this direction, NC number of clusters with equal size of individuals are recognized. Specifically, a cluster would consist of $S_C = \lceil \frac{NP-NC}{NC} \rceil$ number of individuals so that the first cluster consists of S_C top-rank candidate solutions. Each cluster is in charge of injecting a new individual into the current population. Hence, the center of the ith cluster (i.e., x_{C_i}) is calculated as follows to be considered as a new individual.

$$\overrightarrow{x_{C_i}} = \frac{\sum_{n=1}^{S_C} \overrightarrow{x_n}}{S_C} \tag{3.1}$$

where $\overrightarrow{x_n}$ is the nth member of cluster ith. More precisely, the gravity center of all cluster members is computed to be considered as the variable of x_{C_i}. Fig. 3.1 illustrates the process of clustering and center-based sampling.

Then, the new cluster centroid-based samples are injected into the population as the last NC candidates. In each iteration, all candidates in the current population, except the last NC members, are updated using the crossover, mutation, and selection operators, the same as the classic DE. However, by sorting the population, the best generated center-based samples participate in the evolution process as the parents in the next generation. In fact, the sorting leads to a shuffling of the population and, consequently for the center-based samples to participate in the generative operators, as well, as the members of population and also potentially as the parents (target vectors) in the next generation.

Algorithm 1 represents the conceptual pseudo-code of the proposed clustering center-based sampling algorithm on a single-objective population-based algorithm.

Algorithm 1: Conceptual algorithmic description of clustering center-based sampling on a population-based algorithm

```
Input  : $D$: Dimension of problem, $Max_{NFC}$: Maximum number of function
         calls, $NP$: Population size, $NC$: Number of clusters, $F_{obj}$: Objective
         function , $F$: Scaling factor , $C_R$: Crossover rate
Output: $x^*$ : Best candidate solution
Initialize $NP$ population $Pop$ using uniform distribution;
Calculate objective values, $Fits$, for population $Pop$;
$NFC = NP$;
$S_C = \lceil \frac{NP-NC}{NC} \rceil$;
while $NFC < Max_{NFC}$ do
    Generate $NP - NC$ offspring by the parent algorithm from the current
     population $Pop$ in size of $NP$;
    Calculate objective values, $Fits$, for population in size of $NP - NC$;
    $NFC = NFC + (NP - NC)$;
    Sort the $NP - NC$ population based on $Fits$ in ascending order;
    for $i \leftarrow 1$ to $NC$ do
        $start \leftarrow ((i-1) * S_C) + 1$;
        $end \leftarrow i * S_C$;
        $x_{c_i} = \frac{\sum_{j=start}^{end} x_j}{S_C}$;
        $Pop(i + (NP - NC)) = x_{c_i}$;
        $Fits(i + (NP - NC)) = F_{obj}(x_{c_i})$;
    end
    $NFC = NFC + NC$;
end
```

Embedding the center-based sampling into any population-based algorithm enables the population to re-center their clusters by using the advantages of center-based candidates to get closer to the global solution. This study builds clusters of the population based on the fitness values of its individuals; therefore, the generated clusters are balanced in terms of population size. Accordingly, several scenarios are imaginable regarding

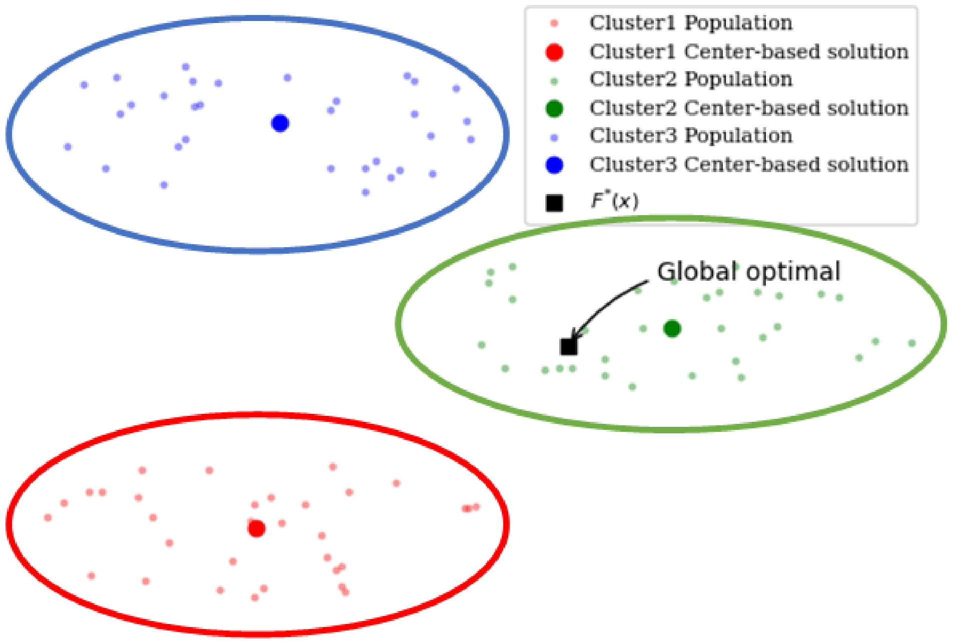

Cluster1 Population
Cluster1 Center-based solution
Cluster2 Population
Cluster2 Center-based solution
Cluster3 Population
Cluster3 Center-based solution
$F^*(x)$
Global optimal

Figure 3.2: A case of solutions placement when the global optimal is inside of the cluster clouds, and the center-based solutions are placed close to the center of gravity.

the relationship between the fitness-based clusters of the population and actual clusters of candidate solutions in the search space. Indeed, it is not possible to have any kind of assumption about the correlation between the individuals' fitness distance and their Euclidean distance in the search space as it is a problem and optimization stage-oriented factor. However, their extreme cases can be categorized as follows:

- Let us assume the clusters based on fitness values are the same as the clusters of candidate solutions in the search space. For this case, the population is divided into some sub-populations (K sub-clouds), in which their centroids are located inside their own cluster's convex hull thus,

  a) if the solution is inside this sub-cloud, then its centroid point will be in a better position to this solution compared to other edge-side members as visualized in Figure 3.2,

  b) if the solution is outside of this sub-cloud's convex hull, then again, the centroid

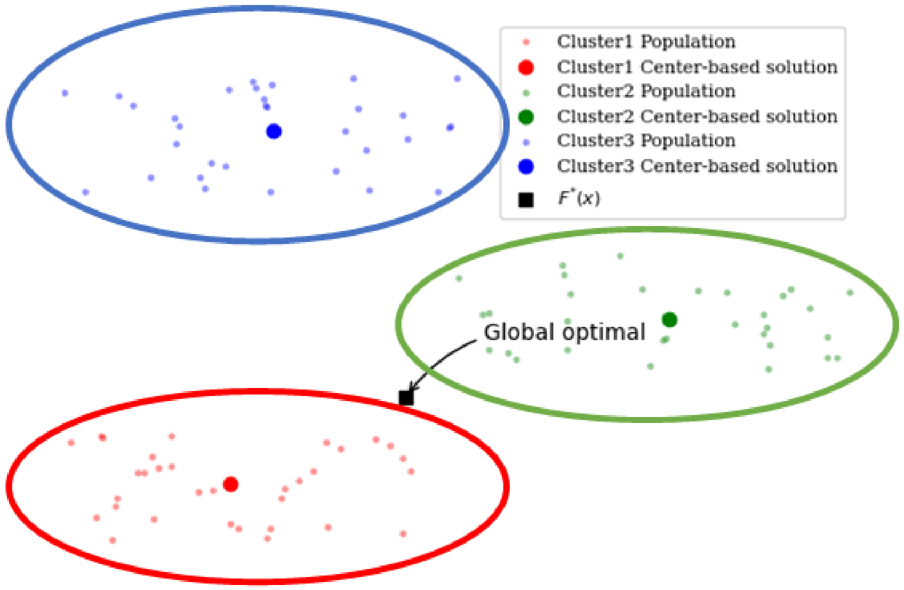

Cluster1 Population
Cluster1 Center-based solution
Cluster2 Population
Cluster2 Center-based solution
Cluster3 Population
Cluster3 Center-based solution
$F^*(x)$
Global optimal

Figure 3.3: A case of solutions placement when the global optimal is outside of the cluster clouds, and the center-based solutions are placed close to the center of gravity.

point would be in a better position compared to members of the sub-cloud which are in the opposite side (tail members) of the moving direction of the cloud toward the solution as shown in Figure 3.3.

- Let us assume the clusters based on fitness values are very weak correlated with the clusters of candidate solutions in the search space. In this case, the centroid of the clusters would be in arbitrary positions (not necessarily inside of the determined clusters' convex-hull); this will increase the algorithm's exploration capability to discover other new promising regions in the search space, which are hard to be visited if the algorithms' ordinary operations are applied 3.4.

It is worth mentioning that the proposed algorithm does not require extra fitness calls compared to the original algorithm. As a result, the proposed method possesses the power of DE generative operators while it is enriched by the utilization of center-based sampling. Moreover, generating center-based samples results in multi-parent crossover [28], [48], which means the new individual is the output of the collective collaboration of

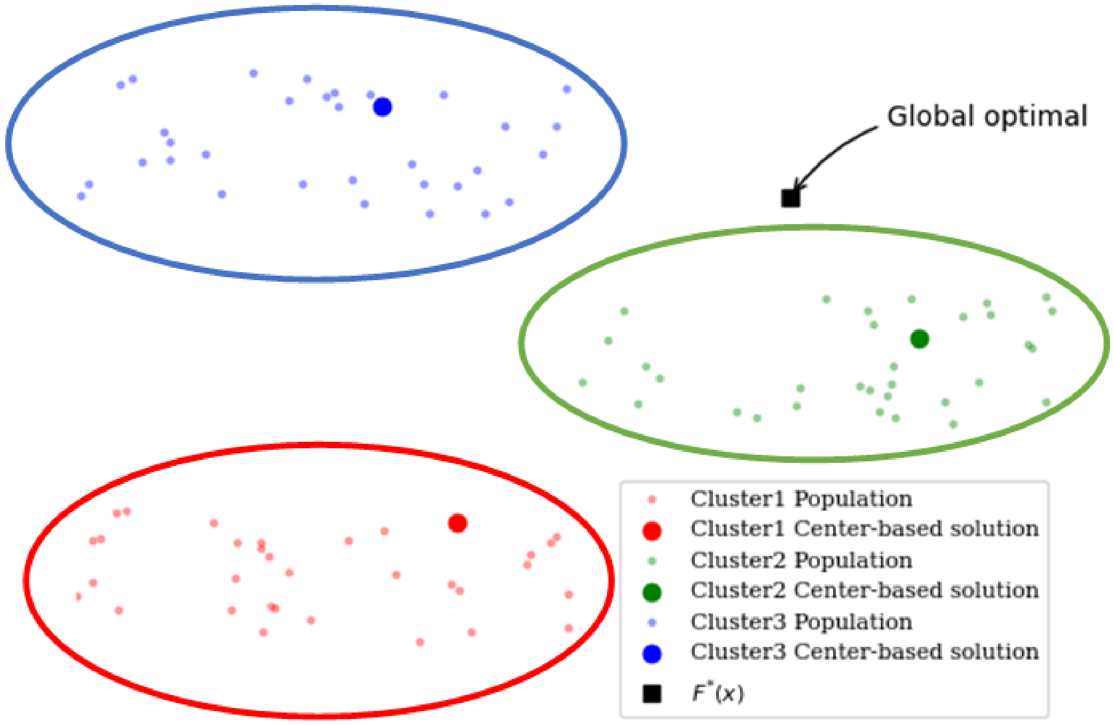

Global optimal
Cluster1 Population
Cluster1 Center-based solution
Cluster2 Population
Cluster2 Center-based solution
Cluster3 Population
Cluster3 Center-based solution
$F^*(x)$

Figure 3.4: A case of solutions placement when the global optimal is outside of the cluster clouds, and the center-based solutions are placed inside of cluster clouds but much closer to the global optimal than other solutions.

a number of parents (i.e., members of a cluster).

3.3 CEC-2017 Single-objective Benchmark Functions

The CEC-2017 [49] offers 30 benchmark functions; however, the second function (F2) is deprecated according to [49]. The properties of the functions are as follows: 1) Unimodal functions: $F1$ and $F3$, 2) Simple multi-modal functions: $F4 - F10$, 3) Hybrid functions: $F11$ - $F20$, and 4) Composite functions: $F21 - F30$.

Each optimization algorithm has a limited budget to call the fitness function, which is set as $Max_{NFC} = 3 \times 10^5$. In the provided results, the mean and standard deviation of the function error value $f(\overrightarrow{x}) - f(\overrightarrow{x}^*)$ are calculated over 31 independent runs for each test function, where $\overrightarrow{x}$ is the best solution in the population when the algorithm terminates and $\overrightarrow{x}^*$ is the global optimal solution.

The non-parametric Wilcoxon rank-sum statistical test with a 95% confidence interval is performed to prove the statistical significance of the experimental results among each method. In the last two rows of each result table, $w/t/l$ indicates that the proposed scheme wins in w functions, ties in t functions, and loses in l functions based on Wilcoxon rank-sum test results regarding the best and average fitness values of the population when the algorithm terminates.

3.4 Case Study One: Clustering Center-based Differential Evolution Algorithm

A Clustering Center-based DE (CCDE) technique is suggested in this thesis to enhance the DE's capabilities for exploitation and exploration [2].

Experimental Results:

In order to assess the performance of the proposed scheme, a series of experiments are conducted on CEC-2017 benchmark functions [49] on dimensions 30, 50, and 100. The control parameters are set as follows:

- Population size: $NP = 100$
- Mutation scale factor: $F = 0.8$
- Crossover rate: $CR = 0.9$
- Strategy [13]: *DE/rand/1/bin*

It should be noted that " ‡ " signifies that the proposed method under consideration is better than the classic DE, " † " means the opposite, and " ≈ " means the same statistical results as equivalent to each other.

To demonstrate the improvement of the proposed algorithm compared to the classical DE, the Improved Accuracy Rate (IAR) is provided. This measure calculates the difference ratio of results from both algorithms as follows:

$$IAR = \frac{Error\ of\ DE}{Error\ of\ CCDE} \tag{3.2}$$

where the error of algorithm is calculated as $f(\overrightarrow{x}) - f(\overrightarrow{x}^*)$.

Tables 3.1, 3.2, and 3.3 represent the results of the CCDE and DE on dimensions 30, 50, and 100, respectively. Each column of CCDE shows the results for a specific number of clusters (i.e., NC). The IAR metric shows the ratio of the classic DE error and the CCDE with $NC = 10$ (as the winner scheme) error to evaluate the performance of the algorithms in reaching the global optimal solution. If $IAR > 1$ indicates that the proposed method with the related NC value outperforms the classic DE as it is able to reach less error. From Tables 3.1, 3.2, and 3.3, it is seen that the calculated IAR value for each function is greater than one, which proves that the proposed algorithm achieves significantly better results compared to the classic DE.

The comparison is provided based on the following: 1) Wilcoxon rank-sum test on best fitness values, and 2) Wilcoxon rank-sum test on average fitness values of the whole population when the algorithm terminates. The improvement in average fitness values is an indicator of the overall quality of the whole population rather than the only best candidate solution, which is achieved by center-based sampling. From Table 3.1, the best candidate solution resulting from CCDE has a better fitness value than the classic DE on all benchmark functions for dimension $D = 30$. In terms of the average value of population, CCDE outperforms the classic DE on all of the functions. In most of the functions, IAR is a large number; for instance, on $D = 30$, CCDE on F_3 could find a candidate solution that is $6.97E07$ times better amount than the amount obtained by the classic DE. In addition to F_3, IAR value for F_1 is a result of division by zero error value since the CCDE could find the best global candidate solution before $\frac{1}{3}$ of MAX_{NFC}.

In addition to the comparison between CCDE and DE, the sum of errors (i.e., SE) values from Table 3.4 represents the best-recommended value for number of clusters which is an important parameter for the proposed algorithm. According to Table 3.1, CCDE with $NC = 10$ offers more winners compared to $NC = 5$ and $NC = 20$. Obviously, increasing the number of injected center-based samples improves the performance of the algorithm compared to the original algorithm, but a high value for the number of clusters can negatively affect the diversity of the population. This could be the reason behind the superiority of 10 clusters.

Similar results are obtained on $D = 50$ and $D = 100$ dimensions. As seen in Table 3.2, the CCDE achieved better results than the DE on all functions for $D = 50$ as IAR represents a value higher than one. Similarly, the best cluster number for the CCDE is $NC = 10$, where it could get the best performance. Finally, Table 3.3 represents the results on dimension $D = 100$. The best candidate solution resulting from the proposed method has a significantly smaller fitness value than the classic DE, and $NC = 10$ is the best value for the number of injected center-based samples into the population. The difference between these two algorithms is mainly significant as the high values of IAR reveal. For instance, looking at Table 3.3, for F_{30} function on $D = 100$, the CCDE was able to reach a candidate solution that is $3.38E06$ times better than the candidate solution found by the classic DE.

In order to investigate the best value for NC, the sum of errors is calculated to compare all schemes. This metric is suggested in [49] and computed as follows:

$$SE = 0.2 \times \sum_{i=1}^{29} ef_{30D} + 0.3 \times \sum_{i=1}^{29} ef_{50D} + 0.5 \times \sum_{i=1}^{29} ef_{100D} \tag{3.3}$$

where ef_{30D}, ef_{50D}, and ef_{100D} are the error of functions on dimensions 30, 50, and 100, respectively. Table 3.4 shows the sum of error values for each value of NC where $NC = 10$ achieved the least error value overall.

In addition to numerical results, the performance plots for three sample benchmark functions are illustrated in Figure 3.5 to reveal the performance of CCDE versus the classic DE. The performance plots indicate that the CCDE decreases the fitness values of the best candidate very sharply compared to the classic DE. As it is presented in the figure, even before the optimizer consumes half of Max_{NFC}, it would be able to find a significantly better solution. These plots clearly indicate the exploration power of the proposed method. Furthermore, Figure 3.6 represents the performance plots with $\frac{1}{6}$ of the function call budget to show the capability of the proposed method to make a significant difference with the classic DE even with having a low number of function calls budget. As a result, the algorithm can easily tackle the expensive optimization problems which do not afford a high value of fitness calls.

Table 3.1: Results of the algorithms on CEC-2017 benchmark functions on $D = 30$ with Max_{NFC} budget. IAR is for comparison between the winner NC and the classic DE. The symbols "‡", "*dagger*", and "≈" indicate that the CCDE algorithm performs either better than, worse than, or similarly to the comparable algorithm (DE), respectively.

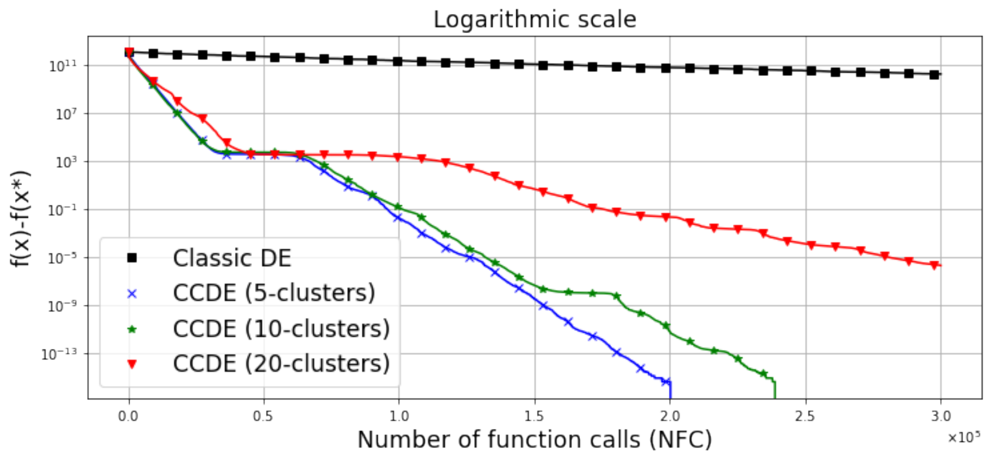
$30D$		**CCDE**			**DE**	IAR
F_n		$NC = 5$	$NC = 10$	$NC = 20$	***Classic***	$NC = 10$
F_1	Mean	0.00E+00‡	0.00E+00‡	2.00E-06‡	1.72E+10	$\frac{1.72E+10}{0}$ = inf
	Std	0.00E+00	0.00E+00	9.00E-06	7.46E+09	
F_3	Mean	3.10E-07‡	6.93E-03‡	2.34E+02‡	4.83E+05	6.97E+07
	Std	1.10E-06	3.04E-02	3.94E+02	3.01E+05	
F_4	Mean	6.65E+00‡	1.60E+01‡	5.92E+01‡	1.90E+02	1.19E+01
	Std	1.24E+01	2.19E+01	2.71E+01	6.19E+01	
F_5	Mean	7.32E+01‡	6.55E+01‡	9.37E+01‡	2.69E+02	4.11E+00
	Std	5.83E+01	3.09E+01	1.97E+01	1.67E+01	
F_6	Mean	4.72E+00‡	1.06E+01‡	1.86E+01‡	5.64E+01	5.34E+00
	Std	3.23E+00	4.77E+00	6.83E+00	1.10E+01	
F_7	Mean	1.14E+02‡	8.15E+01‡	1.31E+02‡	3.11E+02	3.81E+00
	Std	6.05E+01	2.76E+01	3.21E+01	2.17E+01	
F_8	Mean	7.83E+01‡	5.86E+01‡	7.41E+01‡	2.69E+02	4.58E+00
	Std	5.71E+01	2.67E+01	1.64E+01	1.49E+01	
F_9	Mean	2.12E+01‡	7.33E+01‡	2.68E+02‡	3.37E+03	4.59E+01
	Std	4.19E+01	8.91E+01	2.31E+02	1.02E+03	
F_{10}	Mean	6.41E+03‡	4.06E+03‡	4.40E+03‡	8.50E+03	2.09E+00
	Std	1.50E+03	2.10E+03	1.27E+03	3.27E+02	
F_{11}	Mean	5.27E+01‡	8.53E+01‡	8.77E+01‡	9.08E+02	1.06E+01
	Std	2.73E+01	3.60E+01	2.81E+01	4.06E+02	
F_{12}	Mean	1.61E+04‡	9.12E+03‡	3.32E+04‡	5.13E+09	5.62E+05
	Std	1.86E+04	6.01E+03	2.21E+04	1.48E+09	
F_{13}	Mean	5.94E+02‡	5.29E+02‡	1.32E+04‡	7.75E+08	1.46E+06
	Std	1.99E+03	1.14E+03	1.13E+04	2.47E+08	
F_{14}	Mean	1.05E+02‡	2.15E+02‡	2.99E+02‡	5.42E+04	2.53E+02
	Std	6.98E+01	9.75E+01	1.61E+02	2.00E+04	
F_{15}	Mean	5.01E+01‡	1.90E+02‡	1.13E+03‡	4.71E+07	2.48E+05
	Std	5.03E+01	4.54E+02	1.21E+03	1.17E+07	
F_{16}	Mean	3.01E+02‡	3.87E+02‡	8.78E+02‡	2.60E+03	6.71E+00
	Std	2.15E+02	2.57E+02	3.82E+02	2.41E+02	
F_{17}	Mean	5.59E+01‡	8.69E+01‡	1.87E+02‡	1.31E+03	1.51E+01
	Std	3.45E+01	5.82E+01	1.64E+02	2.92E+02	
F_{18}	Mean	2.02E+03‡	9.00E+03‡	4.36E+04‡	1.60E+06	1.78E+02
	Std	3.73E+03	8.90E+03	4.68E+04	5.77E+05	
F_{19}	Mean	2.47E+01‡	2.93E+01‡	1.40E+03‡	4.69E+04	1.60E+03
	Std	2.64E+01	1.77E+01	2.95E+03	2.63E+04	
F_{20}	Mean	6.66E+01‡	8.77E+01‡	2.88E+02‡	9.41E+02	1.07E+01
	Std	5.23E+01	7.71E+01	2.10E+02	3.33E+02	
F_{21}	Mean	2.57E+02‡	2.47E+02‡	2.68E+02‡	4.68E+02	1.90E+00
	Std	5.15E+01	2.39E+01	1.44E+01	1.50E+01	
F_{22}	Mean	1.58E+03‡	5.91E+02‡	8.51E+02‡	8.49E+03	1.44E+01
	Std	2.75E+03	1.65E+03	1.70E+03	3.66E+02	
F_{23}	Mean	4.18E+02‡	4.22E+02‡	4.46E+02‡	6.25E+02	1.48E+00
	Std	5.13E+01	2.57E+01	2.42E+01	1.84E+01	
F_{24}	Mean	4.79E+02‡	4.96E+02‡	5.22E+02‡	7.09E+02	1.43E+00
	Std	3.32E+01	2.01E+01	2.43E+01	1.94E+01	
F_{25}	Mean	3.79E+02‡	3.80E+02‡	3.85E+02‡	4.62E+02	1.22E+00
	Std	3.68E+00	1.06E+01	1.62E+01	2.24E+01	
F_{26}	Mean	1.68E+03‡	1.84E+03‡	2.26E+03‡	3.43E+03	1.86E+00
	Std	3.71E+02	3.88E+02	5.44E+02	1.82E+02	
F_{27}	Mean	5.00E+02‡	5.00E+02‡	5.00E+02‡	5.00E+02	1.00E+00
	Std	2.16E-04	3.86E-04	2.67E-04	6.98E-05	
F_{28}	Mean	5.00E+02‡	5.00E+02‡	4.94E+02‡	5.00E+02	1.00E+00
	Std	1.75E-04	3.74E-04	3.53E+01	7.39E-05	
F_{29}	Mean	4.47E+02‡	3.98E+02‡	6.08E+02‡	2.21E+03	5.55E+00
	Std	1.14E+02	1.03E+02	1.47E+02	2.07E+02	
F_{30}	Mean	2.40E+02‡	2.74E+02‡	1.32E+03‡	5.66E+06	2.07E+04
	Std	3.43E+01	5.43E+01	2.06E+03	4.54E+06	
WRS	Sum	1.958E+02	**1.960E+02**	1.951E+02	Mean IAR = **2.57E+06** (*without F_1)	
Best	$w/t/l$	**29/0/0**	**29/0/0**	**29/0/0**		
Ave	$w/t/l$	**29/0/0**	**29/0/0**	**29/0/0**		

Table 3.2: Results of the algorithms on CEC-2017 benchmark functions on $D = 50$ with Max_{NFC} budget. The IAR indicates an improved accuracy rate which shows the relative improvement and compares the winner NC and the classic DE. The symbols "‡", "*dagger*", and "≈" indicate that the CCDE algorithm performs either better than, worse than, or similarly to the comparable algorithm (DE), respectively.

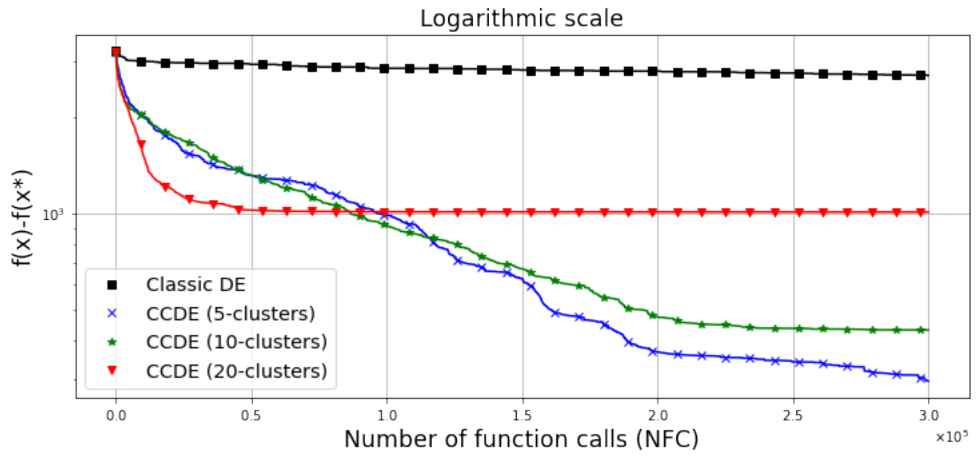
$50D$		**CCDE**			**DE**	IAR
F_n		$NC = 5$	$NC = 10$	$NC = 20$	***Classic***	$NC = 10$
F_1	Mean	3.90E+01‡	2.06E+03‡	2.13E+03‡	1.14E+11	5.56E+07
	Std	1.36E+02	4.02E+03	2.08E+03	3.81E+10	
F_3	Mean	1.62E+04‡	2.55E+04‡	4.49E+04‡	1.36E+06	5.32E+01
	Std	6.25E+03	7.97E+03	1.03E+04	8.28E+05	
F_4	Mean	5.09E+01‡	6.45E+01‡	9.52E+01‡	1.62E+03	2.51E+01
	Std	3.28E+01	4.61E+01	4.43E+01	5.92E+02	
F_5	Mean	1.32E+02‡	1.47E+02‡	2.02E+02‡	5.17E+02	3.53E+00
	Std	7.77E+01	3.55E+01	3.08E+01	3.58E+01	
F_6	Mean	1.48E+01‡	3.99E+00‡	4.32E+00‡	8.29E+01	2.08E+01
	Std	4.52E+00	2.04E+00	1.91E+00	1.33E+01	
F_7	Mean	2.33E+02‡	3.26E+02‡	3.16E+02‡	7.53E+02	2.31E+00
	Std	1.06E+02	1.29E+02	1.50E+02	8.54E+01	
F_8	Mean	1.19E+02‡	1.73E+02‡	1.96E+02‡	5.24E+02	3.04E+00
	Std	4.36E+01	4.55E+01	3.01E+01	3.38E+01	
F_9	Mean	3.68E+02‡	9.51E+02‡	3.79E+03‡	1.78E+04	1.87E+01
	Std	3.15E+02	7.55E+02	2.95E+03	4.83E+03	
F_{10}	Mean	1.27E+04‡	8.97E+03‡	7.38E+03‡	1.53E+04	1.70E+00
	Std	1.90E+03	3.85E+03	1.98E+03	3.75E+02	
F_{11}	Mean	1.09E+02‡	1.36E+02‡	1.92E+02‡	1.26E+04	9.28E+01
	Std	4.08E+01	3.90E+01	5.74E+01	5.56E+03	
F_{12}	Mean	1.32E+05‡	1.47E+05‡	4.99E+05‡	1.20E+11	8.17E+05
	Std	1.01E+05	8.11E+04	4.87E+05	3.34E+10	
F_{13}	Mean	3.92E+03‡	2.31E+03‡	4.66E+03‡	1.44E+10	6.24E+06
	Std	5.71E+03	3.04E+03	4.34E+03	4.60E+09	
F_{14}	Mean	2.60E+03‡	5.22E+03‡	3.13E+04‡	1.46E+06	2.80E+02
	Std	2.16E+03	7.63E+03	3.33E+04	4.71E+05	
F_{15}	Mean	1.03E+04‡	9.93E+03‡	7.82E+03‡	1.39E+09	1.40E+05
	Std	8.53E+03	9.62E+03	5.87E+03	5.43E+08	
F_{16}	Mean	9.75E+02‡	1.02E+03‡	1.41E+03‡	5.39E+03	5.26E+00
	Std	6.21E+02	4.72E+02	3.89E+02	3.65E+02	
F_{17}	Mean	6.63E+02‡	8.00E+02‡	1.16E+03‡	4.41E+03	5.52E+00
	Std	3.19E+02	2.77E+02	3.31E+02	3.67E+02	
F_{18}	Mean	4.36E+04‡	5.36E+04‡	3.06E+05‡	2.88E+07	5.37E+02
	Std	3.75E+04	4.93E+04	2.48E+05	8.93E+06	
F_{19}	Mean	2.79E+02‡	5.14E+03‡	1.48E+04‡	1.82E+08	3.54E+04
	Std	9.43E+02	7.02E+03	8.29E+03	7.96E+07	
F_{20}	Mean	2.97E+02‡	4.30E+02‡	1.01E+03‡	2.72E+03	6.33E+00
	Std	2.70E+02	2.36E+02	3.99E+02	2.43E+02	
F_{21}	Mean	3.35E+02‡	3.28E+02‡	3.83E+02‡	7.16E+02	2.18E+00
	Std	9.46E+01	2.94E+01	3.07E+01	2.00E+01	
F_{22}	Mean	1.31E+04‡	1.05E+04‡	8.80E+03‡	1.56E+04	1.48E+00
	Std	2.10E+03	4.32E+03	3.02E+03	3.78E+02	
F_{23}	Mean	5.68E+02‡	6.14E+02‡	6.68E+02‡	9.71E+02	1.58E+00
	Std	5.87E+01	5.92E+01	5.40E+01	4.38E+01	
F_{24}	Mean	6.61E+02‡	7.11E+02‡	7.79E+02‡	1.07E+03	1.51E+00
	Std	6.35E+01	4.39E+01	6.84E+01	4.06E+01	
F_{25}	Mean	4.56E+02‡	4.57E+02‡	4.70E+02‡	1.55E+03	3.39E+00
	Std	2.35E+01	2.82E+01	3.88E+01	3.76E+02	
F_{26}	Mean	2.86E+03‡	3.58E+03‡	4.99E+03‡	6.18E+03	1.73E+00
	Std	5.71E+02	6.65E+02	8.55E+02	4.63E+02	
F_{27}	Mean	5.00E+02‡	5.00E+02‡	5.00E+02‡	5.00E+02	1.00E+00
	Std	1.58E-04	3.10E-04	2.14E-04	3.13E-05	
F_{28}	Mean	5.00E+02‡	5.00E+02‡	5.00E+02≈	5.00E+02	1.00E+00
	Std	1.50E-04	3.28E-04	1.70E-05	2.72E-05	
F_{29}	Mean	8.74E+02‡	6.77E+02‡	1.14E+03‡	1.30E+04	1.92E+01
	Std	4.51E+02	1.90E+02	2.61E+02	4.09E+03	
F_{30}	Mean	1.42E+03‡	1.06E+03‡	2.82E+03‡	3.98E+09	3.77E+06
	Std	1.53E+03	1.52E+03	2.38E+03	1.40E+09	
WRS	Sum	1.950E+02	**1.961E+02**	1.866E+02	Mean IAR = **2.30E+06**	
Best	$w/t/l$	**29/0/0**	**29/0/0**	**28/1/0**		
Ave	$w/t/l$	**28/1/0**	**29/0/0**	**28/0/1**		

Table 3.3: Results of the algorithms on CEC-2017 benchmark functions on $D = 100$ with Max_{NFC} budget. The IAR indicates an improved accuracy rate, which shows the relative improvement and compares the winner NC and the classic DE. The symbols "‡", "*dagger*", and "≈" indicate that the CCDE algorithm performs either better than, worse than, or similarly to the comparable algorithm (DE), respectively.

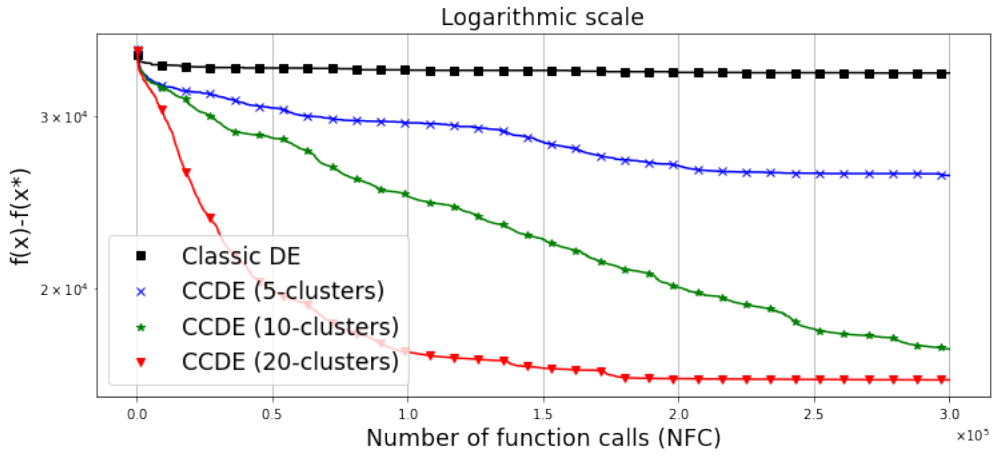
$100D$		**CCDE**			**DE**	IAR
F_n		$NC = 5$	$NC = 10$	$NC = 20$	***Classic***	$NC = 10$
F_1	Mean	5.75E+03‡	5.41E+03‡	1.25E+06‡	9.82E+11	1.81E+08
	Std	7.41E+03	7.26E+03	6.27E+06	1.79E+11	
F_3	Mean	2.30E+05‡	2.25E+05‡	2.46E+05‡	7.09E+06	3.15E+01
	Std	2.12E+04	2.58E+04	2.11E+04	5.25E+06	
F_4	Mean	2.10E+02‡	2.57E+02‡	3.05E+02‡	1.70E+04	6.62E+01
	Std	3.89E+01	5.45E+01	5.13E+01	5.87E+03	
F_5	Mean	3.96E+02‡	4.69E+02‡	5.66E+02‡	1.30E+03	2.76E+00
	Std	7.60E+01	7.75E+01	5.42E+01	7.38E+01	
F_6	Mean	3.72E+01‡	4.48E+01‡	5.50E+01‡	1.08E+02	2.40E+00
	Std	5.59E+00	4.78E+00	4.72E+00	9.48E+00	
F_7	Mean	6.49E+02‡	9.13E+02‡	1.25E+03‡	4.57E+03	5.00E+00
	Std	1.26E+02	1.50E+02	1.76E+02	1.42E+03	
F_8	Mean	4.27E+02‡	5.10E+02‡	6.49E+02‡	1.29E+03	2.53E+00
	Std	1.33E+02	7.90E+01	7.32E+01	6.59E+01	
F_9	Mean	7.26E+03‡	1.17E+04‡	2.17E+04‡	7.38E+04	6.33E+00
	Std	5.07E+03	6.72E+03	1.13E+04	1.44E+04	
F_{10}	Mean	2.61E+04‡	1.74E+04‡	1.62E+04‡	3.31E+04	1.91E+00
	Std	7.34E+03	8.23E+03	5.06E+03	5.47E+02	
F_{11}	Mean	5.27E+02‡	7.67E+02‡	1.18E+03‡	1.31E+06	1.71E+03
	Std	1.44E+02	1.86E+02	4.10E+02	8.66E+05	
F_{12}	Mean	1.85E+06‡	3.20E+06‡	9.36E+06‡	2.30E+11	7.19E+04
	Std	1.11E+06	1.61E+06	4.06E+06	6.30E+10	
F_{13}	Mean	6.35E+03‡	7.32E+03‡	8.71E+03‡	1.93E+10	2.63E+06
	Std	5.28E+03	6.25E+03	6.42E+03	8.41E+09	
F_{14}	Mean	5.04E+04‡	7.08E+04‡	2.95E+05‡	4.52E+07	6.39E+02
	Std	3.05E+04	4.65E+04	1.02E+05	1.12E+07	
F_{15}	Mean	1.87E+03‡	2.20E+03‡	1.32E+03‡	1.08E+10	4.92E+06
	Std	1.97E+03	2.93E+03	1.22E+03	5.43E+09	
F_{16}	Mean	3.66E+03‡	3.34E+03‡	3.67E+03‡	1.24E+04	3.71E+00
	Std	1.55E+03	7.52E+02	8.71E+02	6.21E+02	
F_{17}	Mean	2.96E+03‡	2.64E+03‡	3.12E+03‡	8.63E+04	3.27E+01
	Std	1.09E+03	7.36E+02	5.68E+02	4.60E+04	
F_{18}	Mean	1.85E+05‡	2.99E+05‡	1.13E+06‡	2.85E+08	9.55E+02
	Std	7.41E+04	1.27E+05	4.44E+05	7.26E+07	
F_{19}	Mean	1.85E+03‡	2.38E+03‡	1.47E+03‡	6.54E+09	2.74E+06
	Std	2.16E+03	2.30E+03	1.34E+03	5.23E+09	
F_{20}	Mean	2.53E+03‡	1.96E+03‡	3.36E+03‡	6.59E+03	3.36E+00
	Std	1.15E+03	7.66E+02	1.19E+03	2.49E+02	
F_{21}	Mean	6.11E+02‡	6.74E+02‡	7.98E+02‡	1.57E+03	2.33E+00
	Std	1.19E+02	8.82E+01	8.55E+01	9.19E+01	
F_{22}	Mean	3.09E+04‡	2.69E+04‡	2.16E+04‡	3.44E+04	1.28E+00
	Std	3.68E+03	7.22E+03	5.30E+03	4.69E+02	
F_{23}	Mean	1.02E+03‡	1.12E+03‡	1.30E+03‡	2.04E+03	1.83E+00
	Std	1.19E+02	8.16E+01	9.48E+01	1.14E+02	
F_{24}	Mean	1.57E+03‡	1.77E+03‡	2.04E+03‡	2.86E+03	1.61E+00
	Std	1.68E+02	1.65E+02	1.59E+02	1.68E+02	
F_{25}	Mean	7.70E+02‡	7.94E+02‡	8.42E+02‡	1.24E+04	1.56E+01
	Std	5.41E+01	5.67E+01	4.43E+01	2.38E+03	
F_{26}	Mean	1.05E+04‡	1.31E+04‡	1.64E+04‡	2.34E+04	1.79E+00
	Std	1.18E+03	1.74E+03	2.27E+03	1.81E+03	
F_{27}	Mean	5.00E+02‡	5.00E+02‡	5.00E+02‡	5.00E+02	1.00E+00
	Std	1.32E-04	3.93E-04	3.16E-04	3.15E-05	
F_{28}	Mean	5.00E+02‡	5.04E+02‡	5.58E+02≈	5.00E+02	9.92E-01
	Std	1.62E-04	2.32E+01	6.12E+01	4.91E-05	
F_{29}	Mean	3.15E+03‡	2.80E+03‡	3.40E+03‡	1.54E+05	5.50E+01
	Std	1.35E+03	8.24E+02	5.12E+02	5.57E+04	
F_{30}	Mean	4.28E+03‡	3.52E+03‡	1.12E+04‡	1.19E+10	3.38E+06
	Std	7.14E+03	4.52E+03	1.87E+04	5.21E+09	
WRS	Sum	1.914E+02	**1.951E+02**	1.885E+02		
Best	$w/t/l$	**29/0/0**	**29/0/0**	**28/1/0**	Mean IAR = **6.73E+06**	
Ave	$w/t/l$	**28/0/1**	**29/0/0**	**28/1/0**		



(a) F1, D=30

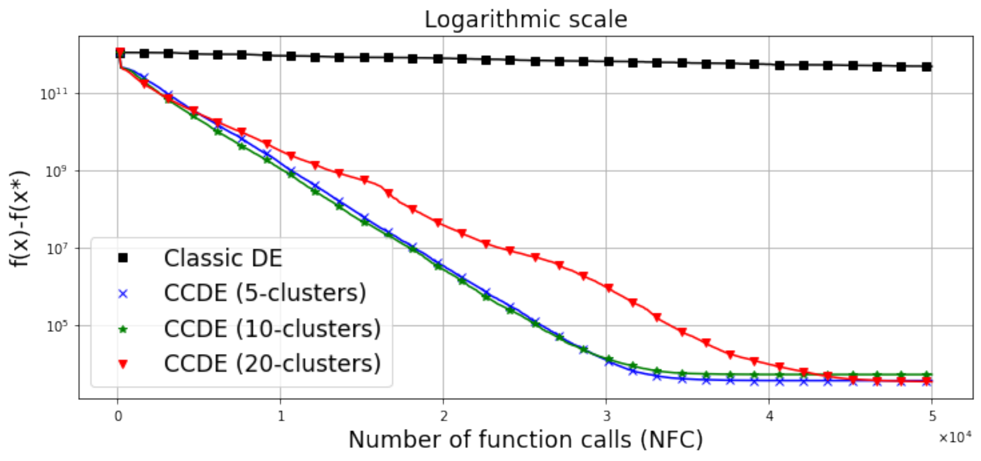


(b) F20, D=50

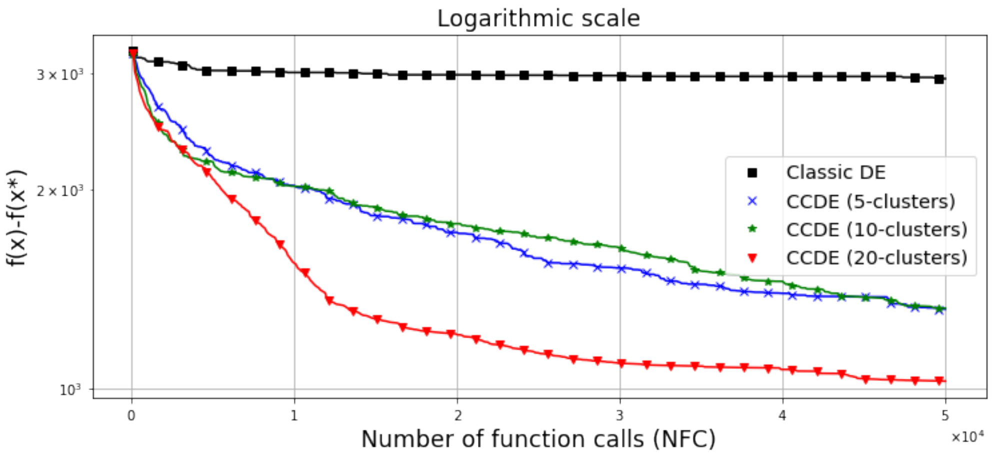


(c) F10, D=100

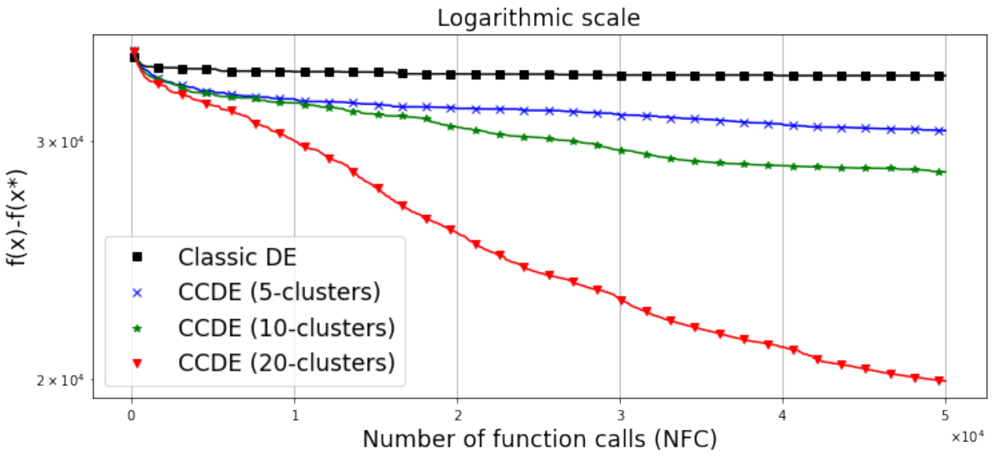
Figure 3.5: The performance plots for the best candidate solution resulted from the proposed method for three sample benchmark functions on full function call budget on various numbers of clusters (NC).



(a) F1, D=30



(b) F20, D=50



(c) F10, D=100

Figure 3.6: The performance plots for the best candidate solution resulted from the proposed method for three sample benchmark functions with $\frac{1}{6}$ of the function call budget on various number of clusters (NC).

3.5 Case Study Two: Clustering Center-based Genetic Algorithm

In this section, Clustering Center-based Genetic Algorithm (CCGA) is tested to solve single-objective optimization problems efficiently. Usually, GA is used to address problems, such as timetabling and scheduling issues, that seem particularly suitable for a solution by GA. Thus, the effectiveness of GA in large-scale problems would be questionable since premature convergence happens. Premature convergence is when GA gets stuck in a local optimum and fails to explore other, potentially better, regions of the search space. Furthermore, GA has difficulty in maintaining diversity as it ensures exploration of different regions in the solution space. One of the ways to address GA's issues is to utilize center-based sampling by computing the center of gravity in population members. As it is proposed in this thesis, the population in GA is initialized with NP randomly generated solutions. In each generation, a new set of NP offspring solutions are generated from the parents selected. In this proposed CCGA, the size of offspring solutions is decreased to $NP - NC$ solutions where NC is the number of center-based solutions. The number of center-based solutions must equal the size of clusters in the population. Therefore, the individuals are clustered into NC clusters with equally shaped. In the study for GA, offspring solutions are considered as the candidates to be clustered and sampled. In order to select the best population out of the combination of the last population and new offspring, the selection operator in GA is applied after

Table 3.4: Comparison between number of clusters (NC) setting effect in clustering analyzed on CEC-2017 benchmark functions over all three dimensions 30, 50, and 100

Algorithm	**Sum of Errors (SE)**
Classic DE	3.55E+12
CCDE - $NC = 5$	1.61E+11
CCDE - $NC = 10$	**1.45E+11**
CCDE - $NC = 20$	1.69E+11

appending center-based solutions. This procedure is repeated until the optimizer reaches the Max_{NFE} budget.

Experimental Results:

To verify the effectiveness and efficiency of the proposed center-based sampling strategy in GA, a series of experiments are presented in this section. The tests are applied to CEC 2017 single-objective benchmark problems with D = 30, 50, and 100 dimensions. The CEC 2017 benchmark consists of 29 functions ($F_1 - F_{30}$ except F_2). To begin analyzing the performance, Table 3.5 shows the results of proposed CCGA and GA for dimension $D = 30$. In summary, the number of losses is greater than wins against the classical GA by 19, sharing 72% of problems. Clearly, the obtained results show the proposed CCGA is not robust on $D = 30$. The functions that are outperformed by the proposed algorithm are F_3 and F_9. Only in six functions, including $F_6, F_{10}, F_{21}, F_{24}, F_{26}$, and F_{27}, the proposed CCGA against its classic GA is tied based on the Wilcoxon test.

However, increasing the problems' dimension will show the effectiveness of center-based solutions in the optimizer's exploration. Looking at Table 3.6, the proposed CCGA, in close competition, surpassed the GA in 11 functions and lost 9 functions. In detail, 80% of the simple multi-modal functions and 40% of composition functions were outperformed by the proposed CCGA. Furthermore, Table 3.7 shows comparative results on $D = 100$ dimensions. In this table, the number of wins for the proposed CCGA is increased to 18, but the number of losses is stayed at 9. More profoundly, all of the unimodal and simple multi-modal functions were outperformed by the proposed CCGA. In addition, CCGA was successful in 80% of composition functions with $D = 100$ dimensions.

In summary, the proposed center-based sampling could enhance the GA optimizer where the search space dimension is exponentially large. In other words, center-based solutions cover GA's randomly generated offspring solutions and remove immature individuals.

Table 3.5: Results of the proposed CC-GA and GA algorithms on CEC-2017 benchmark functions on $D = 30$. Each cell in the table represents the mean and standard deviation (inside parentheses) of 31 runs. The symbols "+", "−", and "=" indicate that the CC-GA algorithm performs either better than, worse than, or similarly to the comparable algorithm (GA), respectively.

Fn	**CC-GA**	**GA**
F1	4.0117e+11 (1.4825e+10) -	**3.7928e+11 (1.7200e+10)**
F3	**1.7477e+17 (4.3930e+15)** +	1.7985e+17 (8.2887e+15)
F4	3.9611e+07 (3.0845e+05) -	**3.9443e+07 (3.7378e+05)**
F5	2.6116e+04 (7.5252e+01) -	**2.6038e+04 (1.1217e+02)**
F6	**1.5587e+03 (3.8085e+01)** =	1.5670e+03 (3.9768e+01)
F7	3.6677e+05 (6.9984e+02) -	**3.6621e+05 (9.9424e+02)**
F8	4.8647e+04 (8.2751e+01) -	**4.8555e+04 (1.2433e+02)**
F9	**2.7619e+06 (1.6519e+05)** +	2.9071e+06 (2.0974e+05)
F10	**2.3566e+04 (5.7982e+02)** =	2.3632e+04 (5.6101e+02)
F11	1.8665e+15 (9.2380e+11) -	**1.8657e+15 (8.0006e+11)**
F12	1.4738e+14 (8.4533e+10) -	**1.4722e+14 (8.4047e+10)**
F13	1.7486e+14 (2.7012e+10) -	**1.7482e+14 (2.6179e+10)**
F14	1.7846e+12 (3.6322e+08) -	**1.7842e+12 (2.6799e+08)**
F15	3.2138e+14 (4.1251e+10) -	**3.2132e+14 (2.3314e+10)**
F16	7.0312e+09 (2.9382e+06) -	**7.0279e+09 (2.7741e+06)**
F17	1.6727e+18 (7.5216e+14) -	**1.6719e+18 (5.1593e+14)**
F18	9.9940e+12 (3.3489e+09) -	**9.9893e+12 (5.5352e+09)**
F19	8.3993e+18 (3.4188e+15) -	**8.3974e+18 (3.4112e+15)**
F20	1.0368e+05 (4.2231e+02) -	**1.0331e+05 (2.6018e+02)**
F21	**1.4601e+10 (1.2806e+07)** =	1.4602e+10 (1.8260e+07)
F22	5.7075e+06 (2.9738e+03) -	**5.7050e+06 (4.6606e+03)**
F23	1.2589e+10 (1.5434e+07) -	**1.2578e+10 (1.5309e+07)**
F24	7.2603e+06 (4.9216e+03) =	**7.2591e+06 (4.9743e+03)**
F25	1.1286e+10 (1.1274e+07) -	**1.1278e+10 (1.3103e+07)**
F26	3.4493e+10 (3.3683e+07) =	**3.4476e+10 (5.1661e+07)**
F27	5.6204e+06 (2.7831e+03) =	**5.6194e+06 (3.6598e+03)**
F28	3.4454e+10 (2.8397e+07) -	**3.4433e+10 (3.8137e+07)**
F29	5.3881e+18 (5.9294e+15) -	**5.3823e+18 (4.5540e+15)**
F30	1.4576e+19 (3.9018e+15) -	**1.4573e+19 (3.3802e+15)**
w/t/l	**2/6/21**	

Table 3.6: Results of the proposed CC-GA and GA algorithms on CEC-2017 benchmark functions on $D = 50$. Each cell in the table represents the mean and standard deviation (inside parentheses) of 31 runs. The symbols "+", "−", and "=" indicate that the CC-GA algorithm performs either better than, worse than, or similarly to the comparable algorithm (GA), respectively.

Fn	**CC-GA**	**GA**
F1	9.4261e+11 (5.2701e+10) =	**9.2017e+11 (6.2834e+10)**
F3	**7.3818e+06 (1.3931e+05)** =	7.3866e+06 (2.6004e+05)
F4	**5.9872e+07 (8.6697e+05)** +	6.0275e+07 (8.4570e+05)
F5	**4.3603e+04 (1.5447e+02)** +	4.3765e+04 (3.0681e+02)
F6	**1.7653e+03 (4.0456e+01)** +	1.8727e+03 (5.7812e+01)
F7	**6.6249e+05 (2.1769e+03)** +	6.6877e+05 (4.0618e+03)
F8	**1.1723e+05 (3.5122e+02)** +	1.1776e+05 (3.7253e+02)
F9	**7.8278e+06 (4.8696e+05)** +	8.3511e+06 (4.7986e+05)
F10	**2.6574e+04 (8.4738e+02)** +	2.7750e+04 (7.0650e+02)
F11	**6.1812e+08 (3.9027e+06)** +	6.2585e+08 (5.6023e+06)
F12	5.6823e+14 (4.5597e+11) -	**5.6746e+14 (3.8949e+11)**
F13	3.9827e+14 (1.6421e+11) -	**3.9789e+14 (1.4136e+11)**
F14	8.5574e+11 (6.4610e+08) -	**8.5471e+11 (5.9750e+08)**
F15	4.4455e+14 (6.2558e+10) -	**4.4432e+14 (6.1130e+10)**
F16	2.3410e+09 (1.9605e+06) -	**2.3378e+09 (1.6011e+06)**
F17	3.9593e+19 (2.6687e+16) -	**3.9528e+19 (2.1162e+16)**
F18	1.2157e+13 (3.9231e+09) -	**1.2149e+13 (3.3278e+09)**
F19	7.3907e+17 (5.6517e+14) -	**7.3841e+17 (3.5777e+14)**
F20	3.7688e+05 (6.9708e+02) =	**3.7680e+05 (6.9699e+02)**
F21	**2.1846e+10 (3.7011e+07)** =	2.1868e+10 (5.4789e+07)
F22	**1.0629e+07 (9.1170e+03)** +	1.0639e+07 (1.3734e+04)
F23	5.0011e+10 (6.2909e+07) =	**4.9993e+10 (7.4680e+07)**
F24	**1.0870e+07 (7.8108e+03)** +	1.0882e+07 (1.2673e+04)
F25	**3.4592e+10 (4.1033e+07)** +	3.4623e+10 (5.7896e+07)
F26	**5.4412e+10 (6.0200e+07)** =	5.4424e+10 (6.8305e+07)
F27	**1.9060e+07 (1.1712e+04)** =	1.9065e+07 (2.1142e+04)
F28	**3.9070e+10 (4.9828e+07)** =	3.9091e+10 (5.3359e+07)
F29	1.8920e+20 (1.0077e+17) =	**1.8916e+20 (1.0209e+17)**
F30	6.6225e+20 (9.0034e+17) -	**6.6093e+20 (7.1943e+17)**
w/t/l	**11/9/9**	

Table 3.7: Results of the proposed CC-GA and GA algorithms on CEC-2017 benchmark functions on $D = 100$. Each cell in the table represents the mean and standard deviation (inside parentheses) of 31 runs. The symbols "+", "−", and "=" indicate that the CC-GA algorithm performs either better than, worse than, or similarly to the comparable algorithm (GA), respectively.

Fn	**CC-GA**	**GA**
F1	**4.5735e+12 (2.1921e+11)** +	4.8975e+12 (2.3956e+11)
F3	**2.0941e+17 (1.4000e+17)** +	4.7000e+17 (2.6552e+17)
F4	**2.1144e+08 (2.6436e+06)** +	2.1577e+08 (3.6503e+06)
F5	**8.4939e+04 (5.3143e+02)** +	8.6621e+04 (7.1521e+02)
F6	**1.7903e+03 (3.6824e+01)** +	2.0139e+03 (3.7325e+01)
F7	**1.4663e+06 (5.2951e+03)** +	1.4960e+06 (8.6346e+03)
F8	**2.2456e+05 (8.0952e+02)** +	2.2713e+05 (1.2174e+03)
F9	**2.4441e+07 (1.4433e+06)** +	2.5442e+07 (1.3717e+06)
F10	**3.9855e+04 (2.3450e+03)** +	4.5135e+04 (9.0650e+02)
F11	2.2741e+18 (1.2869e+16) -	**2.2573e+18 (1.2404e+16)**
F12	**1.1209e+15 (1.6165e+12)** =	1.1217e+15 (1.9058e+12)
F13	4.3878e+14 (5.0916e+11) -	**4.3841e+14 (5.4409e+11)**
F14	1.0032e+12 (2.4432e+09) -	**1.0002e+12 (2.5016e+09)**
F15	3.8511e+14 (2.1229e+11) -	**3.8456e+14 (1.7339e+11)**
F16	6.9273e+09 (9.0933e+06) -	**6.9157e+09 (1.3929e+07)**
F17	2.7041e+19 (6.6537e+16) -	**2.6949e+19 (6.7169e+16)**
F18	5.0833e+11 (1.7513e+09) =	**5.0773e+11 (3.1876e+09)**
F19	8.2979e+18 (1.3549e+16) -	**8.2723e+18 (1.3494e+16)**
F20	**4.0691e+05 (7.1001e+02)** +	4.0986e+05 (1.3169e+03)
F21	**6.0418e+10 (1.3970e+08)** +	6.0780e+10 (1.6582e+08)
F22	**2.0656e+07 (2.1795e+04)** +	2.0750e+07 (3.3080e+04)
F23	**1.9491e+10 (3.4714e+07)** +	1.9633e+10 (5.4478e+07)
F24	**4.2448e+07 (4.9529e+04)** +	4.2556e+07 (6.5761e+04)
F25	**1.1082e+11 (1.7179e+08)** +	1.1121e+11 (2.3637e+08)
F26	**1.1337e+11 (2.2413e+08)** +	1.1393e+11 (2.5612e+08)
F27	**2.1653e+07 (1.8426e+04)** +	2.1711e+07 (2.7499e+04)
F28	**1.0994e+11 (1.9696e+08)** +	1.1023e+11 (2.5334e+08)
F29	3.7198e+20 (4.8191e+17) -	**3.7158e+20 (6.2718e+17)**
F30	9.7594e+20 (1.3329e+18) -	**9.7504e+20 (1.1479e+18)**
w/t/l	**18/2/9**	

3.6 Case Study Three: Clustering Center-based Particle Swarm Optimization

In this section, a clustering center-based sampling on the particles in the PSO algorithm is tested, called CCPSO. In the proposed algorithm, the aim is to calculate the centroid solutions on a number of clusters of solutions. Although the proposed algorithm is similar to the previous case studies, the way of finding suitable candidate solutions may vary. The proposed strategy starts with NP candidate particles and updates the velocity and position of $NP - NC$ particles. The reason behind the $NP - NC$ is the proposed strategy is unwilling to use the extra function call budget in every iteration. After selecting the best solutions and updating $Pbest$ set, the center-based sampling could improve the best solutions in the search space. The improvement could help to fill the gap between the particles and explore other regions in search space. As before, a constant number of clusters NC divides the $Pbest$ particles based on their fitness values. The strategy sorts the particles based on fitness values and finds closer solutions. Each cluster of particles is averaged on their parameters to find the center of gravity. Thus, NC clusters of particles result in NC center-based solutions. To take advantage of the NC center-based solutions, they are added to the $NP - NC$ population of Pbest particles. Furthermore, having NC center-based solution will lead the optimization to promising regions that help to closer positions to global optima.

Experimental Results

The mean and the standard deviation of obtained error values by the proposed CCPSO and classic PSO algorithms with $D = 30$, 50, and 100 dimensions on CEC-2017 benchmark functions are provided in Tables 3.8, 3.9, and 3.10, respectively. Looking at all tables, it can be understood from the w/t/l counts in the last rows that the pro-

Table 3.8: Results of the proposed CCPSO and PSO algorithms on CEC-2017 benchmark functions on $D = 30$. Each cell in the table represents the mean and standard deviation (inside parentheses) of 31 runs. The symbols "+", "−", and "=" indicate that the CCPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (PSO), respectively.

Fn	**CCPSO**	**PSO**
F1	**4.1774e+08 (2.6698e+08)** +	1.3989e+11 (3.9032e+10)
F3	**5.4266e+04 (1.4905e+04)** +	7.7990e+04 (2.3308e+04)
F4	**1.0285e+02 (3.0612e+01)** +	1.8832e+03 (9.9411e+02)
F5	**1.0913e+02 (2.0035e+01)** +	2.0097e+02 (3.5519e+01)
F6	**3.2163e+01 (1.0947e+01)** +	6.1137e+01 (1.0082e+01)
F7	**1.4607e+02 (4.0146e+01)** +	4.6902e+02 (9.0951e+01)
F8	**9.3077e+01 (2.2046e+01)** +	1.6852e+02 (2.7464e+01)
F9	**1.2518e+03 (6.8188e+02)** +	3.7992e+03 (1.4042e+03)
F10	**3.7582e+03 (5.2609e+02)** +	4.2237e+03 (7.4168e+02)
F11	**1.6970e+02 (4.4907e+01)** +	8.2771e+02 (4.0388e+02)
F12	**2.9738e+07 (2.9809e+07)** +	2.8323e+09 (1.5947e+09)
F13	**1.0026e+05 (1.0622e+05)** +	1.0769e+07 (3.0097e+07)
F14	**1.1877e+05 (1.0265e+05)** +	4.8278e+05 (5.2583e+05)
F15	**3.5531e+04 (1.9454e+04)** +	8.2080e+04 (6.6853e+04)
F16	**9.8331e+02 (2.7311e+02)** +	1.4007e+03 (2.9620e+02)
F17	**4.8777e+02 (2.3807e+02)** +	6.9160e+02 (2.6627e+02)
F18	**6.3674e+05 (4.5322e+05)** +	2.9595e+06 (5.2099e+06)
F19	**3.0843e+05 (5.8391e+05)** +	2.1816e+07 (3.0474e+07)
F20	**4.5438e+02 (1.9217e+02)** +	5.5244e+02 (1.6955e+02)
F21	**2.9857e+02 (2.4353e+01)** +	3.9263e+02 (3.9132e+01)
F22	**1.7462e+03 (1.9608e+03)** +	3.8785e+03 (1.3569e+03)
F23	**5.2655e+02 (3.9411e+01)** +	6.9719e+02 (7.4414e+01)
F24	**5.9981e+02 (5.8117e+01)** +	7.6408e+02 (9.0726e+01)
F25	**4.2522e+02 (3.1213e+01)** +	1.0005e+03 (1.9525e+02)
F26	**2.6404e+03 (9.6947e+02)** +	4.4721e+03 (9.3991e+02)
F27	**4.8713e+02 (1.0612e+01)** +	6.2811e+02 (6.8716e+01)
F28	**5.0123e+02 (3.7276e+01)** +	1.6971e+03 (5.0597e+02)
F29	**1.0139e+03 (1.8505e+02)** +	1.7494e+03 (3.9787e+02)
F30	**2.0281e+06 (2.0229e+06)** +	7.3062e+07 (1.0543e+08)
w/t/l	29/0/0	

Table 3.9: Results of the proposed CCPSO and PSO algorithms on CEC-2017 benchmark functions on $D = 50$. Each cell in the table represents the mean and standard deviation (inside parentheses) of 31 runs. The symbols "+", "−", and "=" indicate that the CCPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (PSO), respectively.

Fn	**CCPSO**	**PSO**
F1	**4.3784e+09 (1.5480e+09)** +	5.7880e+11 (9.9671e+10)
F3	**1.6360e+05 (3.3586e+04)** +	1.8851e+05 (4.7309e+04)
F4	**3.1006e+02 (6.2846e+01)** +	1.0467e+04 (3.4600e+03)
F5	**2.3333e+02 (3.4473e+01)** +	4.5294e+02 (5.3294e+01)
F6	**5.4234e+01 (1.2593e+01)** +	8.4178e+01 (1.2614e+01)
F7	**3.7633e+02 (1.0334e+02)** +	1.3288e+03 (1.7957e+02)
F8	**2.2370e+02 (3.3111e+01)** +	4.4785e+02 (4.9772e+01)
F9	**8.0355e+03 (2.2349e+03)** +	1.2434e+04 (3.3323e+03)
F10	**6.9300e+03 (8.8820e+02)** +	7.7032e+03 (8.7835e+02)
F11	**7.9963e+02 (2.9398e+02)** +	7.3396e+03 (4.1698e+03)
F12	**5.0427e+08 (2.7337e+08)** +	7.4543e+10 (3.7339e+10)
F13	**1.1090e+05 (6.8470e+04)** +	1.3709e+10 (1.4983e+10)
F14	**5.9931e+05 (4.6861e+05)** +	5.6934e+06 (9.0970e+06)
F15	**4.8012e+04 (3.4598e+04)** +	1.8622e+08 (7.7388e+08)
F16	**1.9389e+03 (4.2373e+02)** +	2.8058e+03 (4.7062e+02)
F17	**1.4589e+03 (3.8524e+02)** +	2.0815e+03 (4.0878e+02)
F18	**2.4341e+06 (1.5662e+06)** +	1.8533e+07 (1.3895e+07)
F19	**1.9221e+06 (2.3810e+06)** +	3.8944e+07 (3.3788e+07)
F20	**1.0594e+03 (2.9750e+02)** +	1.3384e+03 (3.7497e+02)
F21	**4.2260e+02 (4.2990e+01)** +	6.9252e+02 (5.9549e+01)
F22	**6.9549e+03 (1.5080e+03)** +	8.6789e+03 (9.2593e+02)
F23	**8.3205e+02 (9.1481e+01)** +	1.3089e+03 (1.3024e+02)
F24	**9.6610e+02 (8.5361e+01)** +	1.3878e+03 (1.7198e+02)
F25	**6.9594e+02 (5.9784e+01)** +	6.7658e+03 (1.5695e+03)
F26	**5.6147e+03 (9.3668e+02)** +	1.0249e+04 (1.3171e+03)
F27	**5.8531e+02 (1.3273e+02)** +	1.5090e+03 (2.8062e+02)
F28	**7.8876e+02 (2.5770e+02)** +	5.4222e+03 (1.1041e+03)
F29	**2.1062e+03 (5.3692e+02)** +	4.9238e+03 (1.5348e+03)
F30	**4.7206e+07 (4.1088e+07)** +	6.1098e+08 (4.1649e+08)
w/t/l	29/0/0	

Table 3.10: Results of the proposed CCPSO and PSO algorithms on CEC-2017 benchmark functions on $D = 100$. Each cell in the table represents the mean and standard deviation (inside parentheses) of 31 runs. The symbols "+", "−", and "=" indicate that the CCPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (PSO), respectively.

Fn	**CCPSO**	**PSO**
F1	**1.1668e+11 (3.7582e+10)** +	2.2191e+12 (2.8621e+11)
F3	**4.4064e+05 (6.2046e+04)** +	5.3222e+05 (9.3243e+04)
F4	**1.8427e+03 (4.1760e+02)** +	4.9485e+04 (9.6772e+03)
F5	**6.6735e+02 (6.8322e+01)** +	1.2746e+03 (7.8412e+01)
F6	**6.9694e+01 (9.5429e+00)** +	9.4808e+01 (4.5368e+00)
F7	**1.4822e+03 (1.5672e+02)** +	4.1867e+03 (3.4492e+02)
F8	**7.1507e+02 (9.9736e+01)** +	1.3831e+03 (1.0544e+02)
F9	**2.7170e+04 (3.5443e+03)** +	3.8789e+04 (5.6042e+03)
F10	**1.5810e+04 (1.5102e+03)** +	1.9382e+04 (1.5368e+03)
F11	**7.1211e+04 (2.0821e+04)** +	1.3262e+05 (3.0453e+04)
F12	**9.4281e+09 (3.9423e+09)** +	6.5513e+11 (1.6495e+11)
F13	**1.5145e+08 (1.8641e+08)** +	1.1731e+11 (3.4725e+10)
F14	**4.0986e+06 (1.5344e+06)** +	2.3569e+07 (1.1604e+07)
F15	**5.1387e+04 (2.8109e+04)** +	2.3721e+10 (1.1403e+10)
F16	**4.7035e+03 (8.0663e+02)** +	9.1791e+03 (9.8834e+02)
F17	**3.8709e+03 (5.2445e+02)** +	1.8712e+04 (1.0400e+04)
F18	**4.6518e+06 (2.4655e+06)** +	3.3712e+07 (2.2607e+07)
F19	**1.6727e+07 (1.2954e+07)** +	2.0759e+10 (1.3821e+10)
F20	**3.1168e+03 (4.4298e+02)** +	3.4369e+03 (4.4932e+02)
F21	**1.0526e+03 (8.3924e+01)** +	1.8919e+03 (1.3097e+02)
F22	**1.8281e+04 (1.4940e+03)** +	2.1059e+04 (1.1641e+03)
F23	**1.8793e+03 (2.3063e+02)** +	2.7565e+03 (2.4418e+02)
F24	**2.6351e+03 (2.3397e+02)** +	4.4597e+03 (3.3313e+02)
F25	**2.0191e+03 (3.0835e+02)** +	2.2702e+04 (3.1645e+03)
F26	**2.0153e+04 (2.4175e+03)** +	3.7325e+04 (3.4084e+03)
F27	**1.3800e+03 (3.3108e+02)** +	3.9004e+03 (5.7825e+02)
F28	**3.3602e+03 (6.3061e+02)** +	2.4080e+04 (3.5312e+03)
F29	**6.1527e+03 (1.0175e+03)** +	2.2577e+04 (1.1519e+04)
F30	**4.3799e+08 (2.4832e+08)** +	5.3261e+10 (2.6337e+10)
w/t/l	29/0/0	

posed CCPSO won 29 out of 29 problems for the experimented dimensions set. In other words, CCPSO method has outperformed the classic PSO algorithm in all extensive single-objective optimization benchmark problems proposed in CEC 2017 regardless of the search space's complexity. This result verifies that the proposed center-based sampling can allegedly make PSO algorithm more robust in any condition of optimization problems by exploring the center of batches of solutions.

3.7 Case Study Four: Clustering Center-based Artificial Bee Colony Algorithm

In this algorithm, the tested center-based strategy is added to the ABC algorithm in a similar way. Although the proposed scheme does not cost extra budget for function evaluation, the candidate solutions are selected from the population in size of $NP - NC$ to retain the dedicated budget in every generation. First, the population is initialized randomly. Secondly, the employed bees and onlooker bees are generated to be replaced if their fitness value is improved from the previous generation. After the replacement, the proposed center-based sampling is added to the algorithm to augment the $NP - NC$ population with NC center-based solutions. In the next generation, a population in size of NP is again used to generate new samples; however, the dedicated budget must not exceed. In order to keep the $NP - NC$ function calls for generating new samples by ABC operators, it is necessary to find the potential individuals that could lead the optimization faster. In this direction, the NP population is sorted based on their fitness value, and the worst solutions are removed from the population. This technique will benefit the algorithm not to waste energy and memory by keeping a fixed number of bad solutions from the population which could harm the optimization in the further steps.

Experimental Results:

To be able to assess the performance of the proposed center-based strategy at population level of ABC algorithm, CEC-2017 series of benchmark functions are tested on three types of dimensions, 30, 50, and 100. In this investigation, it is seen that the majority of problems were outperformed by the proposed algorithm indicating definite acceptance.

Looking at Table 3.11, it can be seen that 20 functions were won by the CC-ABC algorithm against its classic parent ABC algorithm on dimensions 30. Although the rest is split into 5 losses and 4 ties, the strength of the proposed CC-ABC algorithm is admitted by 68% of wins over 29 benchmark problems.

In the latter, from Tables 3.12 and 3.13, it can be understood that the proposed CC-ABC algorithm gained more strength on the convergence of optimization into the promising region when the dimensions increased to 50 and 100, respectively. This demonstrates that, as the dimension of search space increases, the harder ABC algorithm finds to converge; however, utilization of the proposed center-based strategy can mitigate the large dimensions.

In sum, the influence of the proposed center-based sampling on the ABC algorithm for solving single-objective optimization problems was studied. All extensive experiments were performed on CEC 2017 single objective optimization benchmark functions, which were outperformed by the proposed CC-ABC algorithm against its classic version for dimensions 30, 50, and 100. The experiments confirmed that the proposed center-based sampling at the population level of ABC algorithm has more than a 68% success ratio.

Table 3.11: Results of the proposed CC-ABC and ABC algorithms on CEC-2017 benchmark functions on $D = 30$. The symbols "+", "−", and "=" indicate that the CC-ABC algorithm performs either better than, worse than, or similarly to the comparable algorithm (ABC), respectively.

Fn	**CC-ABC**	**ABC**
F1	**3.3784e+3 (2.66e+3)** =	5.9799e+3 (6.16e+3)
F3	**1.2828e+4 (3.76e+3)** +	2.4697e+5 (4.88e+4)
F4	**7.5462e+1 (2.79e+1)** +	8.7654e+1 (1.41e-1)
F5	**7.1509e+1 (1.40e+1)** +	2.6947e+2 (2.42e+1)
F6	4.8276e-1 (5.69e-1) -	**1.2116e-1 (5.50e-2)**
F7	**1.1527e+2 (2.18e+1)** +	3.4209e+2 (2.21e+1)
F8	**5.3310e+1 (1.08e+1)** +	2.8485e+2 (1.45e+1)
F9	1.5734e+2 (9.51e+1) -	**3.2662e+1 (2.73e+1)**
F10	**2.4561e+3 (4.81e+2)** +	8.6085e+3 (3.89e+2)
F11	**6.8654e+1 (2.82e+1)** +	1.0398e+3 (4.43e+3)
F12	**1.5166e+5 (8.83e+4)** +	6.8869e+5 (6.06e+5)
F13	**1.1072e+4 (8.45e+3)** =	1.4954e+4 (1.62e+4)
F14	**6.9874e+3 (7.07e+3)** +	7.2003e+4 (6.26e+4)
F15	**1.1618e+3 (1.21e+3)** +	6.5473e+3 (8.37e+3)
F16	**7.4386e+2 (2.74e+2)** +	2.0200e+3 (2.49e+2)
F17	**2.9758e+2 (1.58e+2)** +	1.0552e+3 (2.08e+2)
F18	**1.0103e+5 (4.26e+4)** +	1.4709e+6 (1.87e+6)
F19	**4.5552e+3 (2.33e+3)** +	1.4764e+4 (1.83e+4)
F20	**2.4247e+2 (1.20e+2)** +	1.1618e+3 (2.54e+2)
F21	**2.5191e+2 (1.13e+1)** +	4.6395e+2 (1.46e+1)
F22	1.0042e+2 (1.12e+0) -	**1.0000e+2 (4.37e-3)**
F23	**4.0981e+2 (1.88e+1)** +	6.0338e+2 (1.78e+1)
F24	**4.7510e+2 (1.62e+1)** +	6.6473e+2 (2.04e+1)
F25	4.0803e+2 (1.89e+1) -	**3.8719e+2 (7.11e-2)**
F26	**2.1800e+3 (6.93e+2)** +	3.6658e+3 (2.20e+2)
F27	5.3616e+2 (9.95e+0) -	**5.0999e+2 (6.67e+0)**
F28	**3.9768e+2 (7.70e+0)** +	4.3586e+2 (2.31e+1)
F29	7.5742e+2 (2.04e+2) =	**7.5523e+2 (1.18e+2)**
F30	**4.0572e+3 (9.69e+2)** =	6.1463e+3 (3.92e+3)
w/t/l	20/4/5	

Table 3.12: Results of the proposed CC-ABC and ABC algorithms on CEC-2017 benchmark functions on $D = 50$. The symbols "+", "−", and "=" indicate that the CC-ABC algorithm performs either better than, worse than, or similarly to the comparable algorithm (ABC), respectively.

Fn	**CC-ABC**	**ABC**
F1	**6.5024e+2 (8.31e+2)** +	1.0260e+10 (3.44e+9)
F3	**7.2237e+4 (9.12e+3)** +	4.9094e+5 (1.12e+5)
F4	**9.3889e+1 (5.09e+1)** +	1.0374e+3 (2.60e+2)
F5	**1.6802e+2 (2.41e+1)** +	5.6787e+2 (2.28e+1)
F6	**4.8488e+0 (2.65e+0)** +	3.6685e+1 (5.31e+0)
F7	**3.0212e+2 (5.50e+1)** +	1.2410e+3 (1.31e+2)
F8	**1.7043e+2 (2.50e+1)** +	5.6212e+2 (2.63e+1)
F9	**1.7446e+3 (5.62e+2)** +	9.1563e+3 (1.76e+3)
F10	**5.0558e+3 (5.38e+2)** +	1.5520e+4 (4.87e+2)
F11	**1.1893e+2 (2.44e+1)** +	3.3645e+4 (8.78e+3)
F12	**9.5859e+5 (3.86e+5)** +	1.1864e+8 (8.83e+7)
F13	**1.5757e+3 (1.51e+3)** +	8.5403e+3 (8.96e+3)
F14	**6.9042e+4 (4.05e+4)** +	2.5127e+5 (1.81e+5)
F15	8.4177e+3 (4.77e+3) =	**8.0090e+3 (5.88e+3)**
F16	**1.2533e+3 (3.43e+2)** +	4.2801e+3 (2.91e+2)
F17	**1.0792e+3 (2.54e+2)** +	2.7754e+3 (2.26e+2)
F18	**3.5597e+5 (1.40e+5)** +	4.1360e+6 (2.74e+6)
F19	1.5716e+4 (4.76e+3) =	**1.4581e+4 (1.17e+4)**
F20	**6.3307e+2 (2.60e+2)** +	2.7142e+3 (2.04e+2)
F21	**3.2730e+2 (2.24e+1)** +	7.6242e+2 (2.63e+1)
F22	**4.1196e+3 (2.87e+3)** +	1.5712e+4 (5.74e+2)
F23	**5.9604e+2 (4.31e+1)** +	9.6727e+2 (2.49e+1)
F24	**6.5153e+2 (3.62e+1)** +	9.8849e+2 (2.61e+1)
F25	**5.8829e+2 (1.63e+1)** +	1.3579e+3 (1.93e+2)
F26	**5.2622e+3 (8.82e+2)** +	6.6006e+3 (2.30e+2)
F27	7.8295e+2 (7.13e+1) -	**5.6462e+2 (5.11e+1)**
F28	**5.3313e+2 (2.48e+1)** +	7.5822e+2 (8.50e+1)
F29	**1.2855e+3 (3.22e+2)** +	2.8527e+3 (2.85e+2)
F30	**8.7076e+5 (8.95e+4)** +	9.9632e+5 (2.43e+5)
w/t/l	26/2/1	

Table 3.13: Results of the proposed CC-ABC and ABC algorithms on CEC-2017 benchmark functions on $D = 100$. The symbols "+", "−", and "=" indicate that the CC-ABC algorithm performs either better than, worse than, or similarly to the comparable algorithm (ABC), respectively.

Fn	**CC-ABC**	**ABC**
F1	**4.7643e+3 (2.85e+3)** +	1.3418e+11 (1.39e+10)
F3	**2.6769e+5 (1.84e+4)** +	2.2708e+6 (3.72e+6)
F4	**3.0108e+2 (5.30e+1)** +	1.4520e+4 (2.80e+3)
F5	**5.2376e+2 (4.62e+1)** +	1.4214e+3 (5.76e+1)
F6	**2.6622e+1 (4.64e+0)** +	8.1157e+1 (4.42e+0)
F7	**1.1882e+3 (1.69e+2)** +	6.5789e+3 (6.88e+2)
F8	**5.5172e+2 (5.82e+1)** +	1.4145e+3 (5.07e+1)
F9	**1.0398e+4 (1.28e+3)** +	5.9740e+4 (5.82e+3)
F10	**1.2022e+4 (1.06e+3)** +	3.3358e+4 (7.97e+2)
F11	**3.3279e+3 (1.62e+3)** +	5.1846e+5 (8.72e+4)
F12	**5.9321e+6 (1.77e+6)** +	1.5039e+10 (4.08e+9)
F13	**3.6977e+3 (1.81e+3)** +	2.4789e+4 (8.98e+3)
F14	**8.0183e+5 (1.80e+5)** +	5.8820e+6 (2.72e+6)
F15	**1.0261e+3 (9.26e+2)** +	5.8419e+3 (3.28e+3)
F16	**3.9472e+3 (5.26e+2)** +	1.0524e+4 (5.03e+2)
F17	**2.9766e+3 (4.34e+2)** +	7.2291e+3 (4.26e+2)
F18	**7.3148e+5 (2.07e+5)** +	6.0967e+7 (2.45e+7)
F19	**1.4621e+3 (1.23e+3)** +	1.6553e+4 (1.00e+4)
F20	**2.2495e+3 (3.99e+2)** +	6.5769e+3 (3.83e+2)
F21	**6.1410e+2 (6.16e+1)** +	1.6686e+3 (4.75e+1)
F22	**1.4589e+4 (1.17e+3)** +	3.4163e+4 (8.66e+2)
F23	**9.5034e+2 (6.04e+1)** +	1.7764e+3 (3.29e+1)
F24	**1.5224e+3 (1.14e+2)** +	2.1813e+3 (3.98e+1)
F25	**8.6667e+2 (4.62e+1)** +	2.0023e+4 (2.28e+3)
F26	1.7162e+4 (1.69e+3) =	**1.7068e+4 (4.69e+2)**
F27	1.1311e+3 (9.98e+1) -	**8.5976e+2 (5.38e+1)**
F28	**6.9973e+2 (2.54e+1)** +	1.0198e+4 (2.75e+3)
F29	**3.6560e+3 (5.15e+2)** +	8.8885e+3 (4.95e+2)
F30	**9.1108e+3 (4.20e+3)** +	4.8259e+6 (2.17e+6)
w/t/l	27/1/1	

3.8 Case Study Five: Clustering Center-based CMA-ES

In the final, the tested center-based strategy in population is experimented on CMA-ES algorithm, which has a slight difference mechanism than the previous population-based algorithms. CMA-ES tries to extend and shrink the occupied search space by its individuals over several generations. When the distribution of the population changes, the space between solutions would be a golden region to reach the global optima. A fast and promising way to search inside the gap is by utilizing center-based solutions based on current individuals. As seen from the Monte Carlo simulation, it is proven that the center-based solution could increase the exploitation and exploration, resulting in accelerating the convergence of optimization. After sampling solutions in size of $NP - NC$ by the multivariate normal distribution, the proposed clustering approach is used to find clusters. In the next step, the centroid of each cluster is calculated, and the total NC number of center-based solutions is added to the population to locate NP individuals.

Experimental Results:

In order to investigate the performance of the proposed Clustering Center-based CMA-ES algorithm versus CMA-ES, a comprehensive experiment was conducted. The proposed CC-CMA-ES algorithm was applied and tested on CEC 2017 single objective benchmark functions with three types of dimensions, namely, 30, 50, and 100. The results of CMA-ES and CC-CMA-ES on CEC 2017 benchmark with 29 single objective functions are summarized in Tables 3.14, 3.15, and 3.16. As it can be seen from Table 3.14, all of the functions were outperformed by the proposed CC-CMA-ES. In some cases, the improved accuracy rate (IAR) shows the significance of superiority. For instance, CC-CMA-ES resulted in 1.19e+4 times better fitness value than the parent algorithm on F15. Besides, as it is revealed from the mentioned tables above, the standard deviation (STD) values

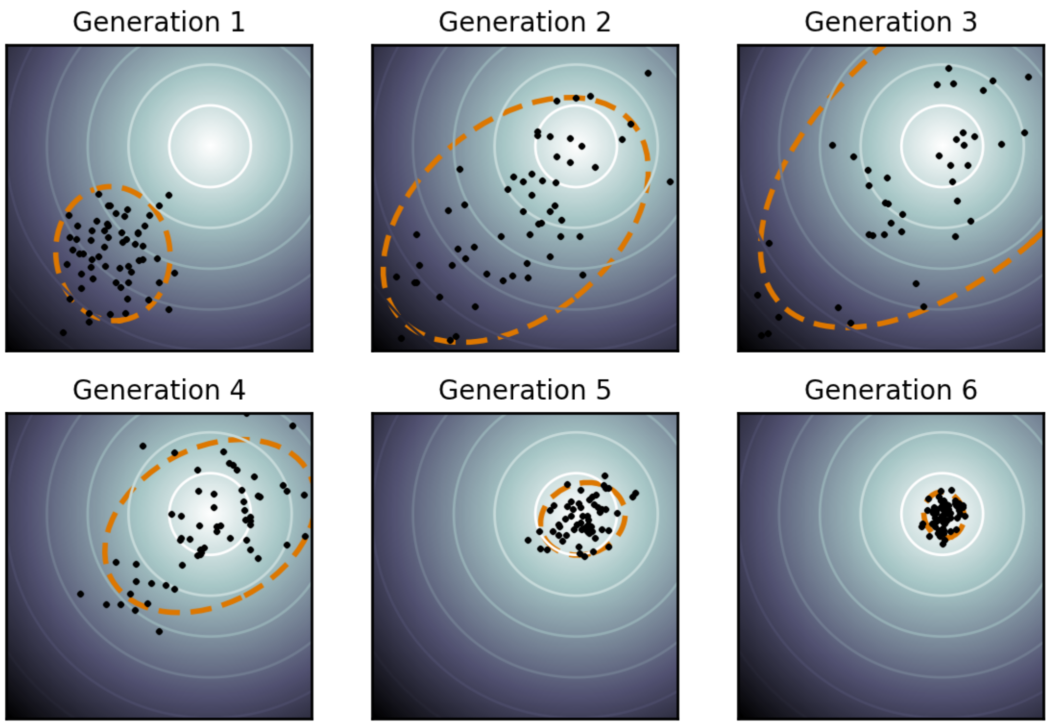


Figure 3.7: Illustration of an actual optimization run with covariance matrix adaptation on a simple two-dimensional problem. The population (dots) is much larger than necessary but clearly shows how the population distribution (dotted line) changes during the optimization. [50]

of the proposed algorithm compared to classical DE for the winner schemes are much less than the parent algorithm. The latter shows higher robustness and effectiveness that occurs from contribution of NC center-based solutions in population.

Table 3.15 shows the results obtained from the same proposed center-based strategy on CMA-ES algorithm and its classic scheme on CEC 2017 with dimension 50. Similar to the results on dimensions 30, it can be seen that the algorithm demonstrates 28 wins out of 29 functions meaning only one of the functions, F9, was challenging since the tie was declared. This table clearly shows evidence for the superiority of proposed center-based strategy on the population-based algorithms.

The last Table 3.16, provided the last set of experiments on CMA-ES. As the number of dimensions increases, the search space extends exponentially and causes the optimization harder to find promising regions. Based on the w/t/l numbers in the last row, the proposed algorithm loses 10 functions and wins 12 functions concluding the experiments. Furthermore, it is noticeable that the difference between CC-CMA-ES results is very close to the CMA-ES for those losing functions such as F6, F7, F16, F17, F20, F21, F23, and F24.

In overall, the proposed CC-CMA-ES performs better on almost all functions on dimensions 30 and 50, where it won 12 out of 29 functions on dimensions 100 for CEC 2017 single objective benchmark functions. As can be observed, the proposed center-based strategy accelerates the convergence speed into promising regions on search space for CMA-ES algorithm.

Table 3.14: Results of the proposed CC-CMA-ES and CMA-ES algorithms on CEC-2017 benchmark functions on $D = 30$. The symbols "+", "−", and "=" indicate that the CC-CMA-ES algorithm performs either better than, worse than, or similarly to the comparable algorithm (CMA-ES), respectively.

Fn	**CC-CMA-ES**	**CMA-ES**
F1	**1.9299e+8 (4.93e+8)** +	1.6649e+11 (2.28e+9)
F3	**6.7819e+4 (1.18e+4)** +	3.3459e+7 (9.01e+7)
F4	**1.3530e+2 (3.68e+1)** +	4.3533e+4 (1.16e+4)
F5	**1.9146e+2 (1.14e+1)** +	8.2760e+2 (1.14e+1)
F6	**5.7427e+0 (6.00e+0)** +	9.6149e+1 (6.75e+1)
F7	**1.9583e+2 (1.29e+1)** +	2.6698e+2 (2.00e+1)
F8	**1.9102e+2 (1.24e+1)** +	7.5060e+2 (7.20e+0)
F9	**7.9416e+0 (2.51e+1)** +	4.3922e+3 (1.36e+4)
F10	**6.8034e+3 (3.68e+2)** +	7.5160e+3 (3.80e+2)
F11	**3.0958e+3 (1.94e+3)** +	5.4331e+4 (2.35e+3)
F12	**8.9281e+7 (8.61e+7)** +	4.2999e+10 (7.80e+8)
F13	**2.6710e+7 (2.98e+7)** +	4.9794e+10 (3.09e+9)
F14	**4.7263e+5 (9.42e+5)** +	1.8178e+8 (6.68e+7)
F15	**2.1872e+6 (1.82e+6)** +	2.6050e+10 (1.56e+4)
F16	**1.4870e+3 (2.41e+2)** +	5.2545e+3 (3.13e+2)
F17	**7.0761e+2 (2.02e+2)** +	1.5323e+6 (2.95e+1)
F18	**3.3264e+6 (3.01e+6)** +	3.0377e+8 (1.28e+8)
F19	**9.1080e+6 (1.05e+7)** +	3.7958e+10 (1.20e+8)
F20	**6.5958e+2 (1.75e+2)** +	1.6558e+3 (1.01e+2)
F21	**3.8556e+2 (1.29e+1)** +	9.0713e+2 (1.37e+1)
F22	**6.0996e+3 (2.36e+3)** +	8.1988e+3 (1.56e+3)
F23	**5.7494e+2 (1.41e+1)** +	1.0247e+3 (3.81e+0)
F24	**6.4386e+2 (1.55e+1)** +	8.8755e+2 (1.59e+1)
F25	**4.1250e+2 (1.37e+1)** +	2.0608e+4 (1.11e+3)
F26	**3.0520e+3 (6.71e+2)** +	8.6111e+3 (6.84e+1)
F27	**5.6736e+2 (2.60e+1)** +	8.0302e+2 (4.52e+1)
F28	**6.7144e+2 (8.71e+1)** +	5.1085e+3 (7.55e+2)
F29	**1.1962e+3 (2.61e+2)** +	8.0273e+3 (1.29e+4)
F30	**8.5519e+6 (5.70e+6)** +	8.3670e+9 (1.05e+9)
w/t/l	29/0/0	

Table 3.15: Results of the proposed CC-CMA-ES and CMA-ES algorithms on CEC-2017 benchmark functions on $D = 50$. The symbols "+", "−", and "=" indicate that the CC-CMA-ES algorithm performs either better than, worse than, or similarly to the comparable algorithm (CMA-ES), respectively.

Fn	**CC-CMA-ES**	**CMA-ES**
F1	**4.4566e+8 (4.92e+8)** +	2.6405e+11 (5.44e+10)
F3	**1.7588e+5 (2.50e+4)** +	1.1814e+9 (5.38e+9)
F4	**2.8782e+2 (9.83e+1)** +	1.0374e+5 (1.76e+4)
F5	**3.5183e+2 (1.55e+1)** +	1.3772e+3 (2.67e+2)
F6	**2.2149e+0 (1.96e+0)** +	1.1067e+2 (6.09e+1)
F7	**3.9375e+2 (1.34e+1)** +	4.3765e+2 (1.25e+2)
F8	**3.5624e+2 (1.52e+1)** +	1.3744e+3 (1.69e+2)
F9	**6.8415e+2 (1.46e+3)** =	4.2942e+4 (4.47e+4)
F10	**1.1529e+4 (1.69e+3)** +	1.3519e+4 (7.96e+2)
F11	**1.2629e+4 (4.26e+3)** +	1.4023e+5 (9.76e+4)
F12	**6.0674e+8 (5.17e+8)** +	1.1889e+11 (7.05e+9)
F13	**1.6795e+8 (1.81e+8)** +	9.0378e+10 (1.20e+10)
F14	**2.8330e+6 (3.65e+6)** +	2.9077e+8 (1.28e+8)
F15	**1.7335e+7 (2.71e+7)** +	6.8913e+10 (1.63e+4)
F16	**2.4039e+3 (3.99e+2)** +	1.2461e+4 (3.75e+2)
F17	**1.8831e+3 (3.75e+2)** +	5.9680e+7 (3.21e+2)
F18	**8.7793e+6 (1.00e+7)** +	7.9405e+8 (3.71e+8)
F19	**9.1379e+6 (1.41e+7)** +	2.2754e+10 (7.84e+1)
F20	**1.5143e+3 (2.07e+2)** +	3.2973e+3 (2.16e+2)
F21	**5.6211e+2 (2.24e+1)** +	1.3573e+3 (2.14e+2)
F22	**1.1666e+4 (2.24e+3)** +	1.4018e+4 (7.29e+2)
F23	**8.2473e+2 (2.45e+1)** +	1.5983e+3 (4.99e+1)
F24	**9.0262e+2 (2.31e+1)** +	1.2878e+3 (3.23e+1)
F25	**5.6312e+2 (3.50e+1)** +	9.4968e+4 (7.95e+3)
F26	**5.1050e+3 (9.25e+2)** +	1.3010e+4 (8.45e+2)
F27	**7.9773e+2 (9.37e+1)** +	1.2598e+3 (1.25e+2)
F28	**1.6466e+3 (1.96e+3)** +	7.5020e+3 (8.65e+2)
F29	**2.1114e+3 (4.14e+2)** +	1.0903e+6 (1.34e+6)
F30	**5.8032e+7 (5.37e+7)** +	3.2270e+10 (1.45e+8)
w/t/l	28/1/0	

Table 3.16: Results of the proposed CC-CMA-ES and CMA-ES algorithms on CEC-2017 benchmark functions on $D = 100$. The symbols “+”, “−”, and “=” indicate that the CC-CMA-ES algorithm performs either better than, worse than, or similarly to the comparable algorithm (CMA-ES), respectively.

Fn	**CC-CMA-ES**	**CMA-ES**
F1	5.3794e+9 (5.05e+9) =	**5.0328e+9 (2.70e+9)**
F3	**8.5460e+5 (1.41e+5)** +	9.2876e+7 (3.10e+8)
F4	5.6262e+2 (2.27e+2) =	**5.3686e+2 (1.08e+2)**
F5	5.4971e+2 (3.16e+2) -	**8.8992e+1 (2.11e+1)**
F6	1.9644e+0 (1.56e+0) -	**7.5912e-1 (8.05e-1)**
F7	8.6676e+2 (1.31e+2) -	**1.4656e+2 (6.43e+0)**
F8	5.4162e+2 (3.00e+2) -	**9.4790e+1 (1.48e+1)**
F9	**1.6037e+3 (1.27e+3)** +	1.2188e+4 (4.32e+4)
F10	**1.2376e+4 (4.17e+3)** =	1.4191e+4 (1.11e+4)
F11	**3.2639e+5 (9.83e+4)** +	1.1709e+6 (2.19e+6)
F12	**1.6999e+9 (1.33e+9)** +	3.2312e+10 (6.76e+10)
F13	**3.3946e+8 (2.47e+8)** +	3.8680e+9 (8.22e+9)
F14	**1.0878e+7 (9.03e+6)** +	4.9450e+8 (2.12e+8)
F15	**3.6138e+7 (6.34e+7)** +	8.8839e+9 (1.41e+10)
F16	3.0336e+3 (9.48e+2) -	**1.9018e+3 (8.33e+2)**
F17	2.8869e+3 (6.93e+2) -	**2.3247e+3 (6.77e+2)**
F18	**1.1320e+7 (8.19e+6)** +	6.6448e+8 (3.55e+8)
F19	**9.0596e+7 (1.22e+8)** +	2.7047e+9 (5.47e+9)
F20	3.4498e+3 (7.66e+2) -	**1.4926e+3 (4.99e+2)**
F21	6.7092e+2 (3.12e+2) -	**3.5171e+2 (3.00e+1)**
F22	**1.3557e+4 (5.35e+3)** =	1.6473e+4 (1.11e+4)
F23	9.4345e+2 (8.51e+1) -	**8.9213e+2 (4.88e+1)**
F24	1.4242e+3 (1.69e+2) -	**1.3226e+3 (6.43e+1)**
F25	9.6638e+2 (1.44e+2) =	**9.3809e+2 (8.55e+1)**
F26	7.6362e+3 (1.84e+3) =	**7.7060e+3 (7.42e+2)**
F27	**8.1814e+2 (7.31e+1)** +	9.8794e+2 (5.02e+2)
F28	**3.2809e+3 (2.78e+3)** =	4.8302e+3 (5.51e+3)
F29	**3.3283e+3 (7.06e+2)** +	3.8916e+3 (5.70e+3)
F30	**1.9190e+8 (2.82e+8)** +	3.8938e+10 (2.51e+10)
w/t/l	12/7/10	

3.9 Summary

Table 3.17: Summary of win, tie, and lose ratio collected from all tables for CEC-2017 benchmark problems on $D = 30$, $D = 50$, and $D = 100$ dimensions. The symbols w show wins, t shows ties, and l shows losses of the ratio of the proposed algorithm against its parent algorithm.

Dimensions	D=30			D=50			D=100			Sum		
Algorithms	w	t	l	w	t	l	w	t	l	w	t	l
CCDE vs. DE	**29**	0	0	**29**	0	0	**29**	0	0	**87**	0	0
CCGA vs. GA	2	6	21	11	9	9	18	2	9	31	17	39
CCPSO vs. PSO	**29**	0	0	**29**	0	0	**29**	0	0	**87**	0	0
CC-ABC vs. ABC	20	4	5	26	2	1	27	1	1	73	7	7
CC-CMA-ES vs. CMA-ES	29	0	0	28	1	0	12	7	10	69	8	10

In this chapter, five population-based algorithms are studied, and the proposed scheme, Clustering Center-based sampling at the population level, is applied to all the algorithms to solve CEC-2017 single-objective benchmark problems. To summarize the results, Tables 3.17 provide a comprehensive outlook on the $w/t/l$ ratio. By looking at Table 3.17, one can understand that center-based sampling involved in any kind of single-objective population-based optimization algorithm could enhance the exploration and exploitation. In general, DE and PSO are global optimization algorithms meaning they aim to find the global optimum of a function, rather than getting stuck in local optima. However, some landscapes are harder to reach the global optimum when the population is converged into a far region. In this thesis, the proposed technique is added to population-based algorithms to solve black-box problems which use no extra budget to find the closer region to global optima. On the performance, it is deniable to see the effect on every problem in the CEC-2017 benchmark. For instance, the proposed Clustering Center-based sampling on DE and PSO algorithms could outperform the parent algorithms on all of the problems. Furthermore, center-based sampling of GA's population could result in better fitness values when the dimension is $D = 100$. Regardless of the population-based algorithm's operators, the proposed Clustering Center-based sampling could improve the

ABC algorithm by resulting in more accuracy in a range of 70% and 93%. The last algorithm approved the impact of center-based solution by winning all problems with $D = 30$ and $D = 50$ dimensions search space.

Overall, the conducted experiments show that the proposed Clustering Center-based sampling on single-objective optimization problems was successful, where more than 90% of CEC-2017 benchmark problems were outperformed on average.

Chapter 4

Proposed Ranking Center-based Sampling for Multi-objective Population-based Algorithms

In this section, the proposed Ranking Center-based Sampling for multi-objectives is presented and explained in detail. Then, the CEC-2017 many-objective benchmark problems are then examined, and their parameter settings are given. Finally, five case studies of five meta-heuristic population-based algorithms are tested to enhance the comparing algorithm. In the first case, the proposed center-based sampling is tested on the NSGA-II algorithm with two clustering techniques, namely, K-Means and Average Ranking. To indicate their effectiveness and efficiency, the two schemes are tested in a fair condition. The winner scheme is applied to other case studies such as NSGA-III, MOEA/D, SPEA2, MOPSO, and GDE3 algorithms. A summary of the overall experiment results is given in the final part so that you can quickly assess how well the suggested method is working.

4.1 Introduction

Multi-objective population-based algorithms have gained prominence as renowned optimization methodologies that have demonstrated efficacy in addressing a multitude of real-world optimization challenges. These algorithms predominantly function as meta-heuristic strategies tailored for navigating intricate search spaces. Their primary objective centers around efficiently exploring the search space to identify optimal solutions within prescribed temporal constraints. The principal challenges faced by multi-objective population-based algorithms encompass an early convergence towards local optima due to the degradation of population diversity during initial optimization stages, coupled with a typically gradual convergence rate.

Real-world multi-objective problems are complex optimization scenarios encountered in various domains, spanning from engineering and manufacturing to finance and urban planning. These problems involve the simultaneous optimization of multiple, often conflicting objectives, where the number of objectives can be substantial, exceeding ten or even reaching the hundreds. Each objective represents a distinct and relevant performance metric, ranging from cost and efficiency to environmental impact and safety. The challenge lies in identifying a set of solutions that collectively represent a trade-off among these objectives, known as the Pareto front, where improving one objective may lead to the degradation of another. Handling multi-objective problems with numerous objectives requires advanced techniques that consider the intricate interplay between diverse objectives while accommodating real-world constraints, fostering a deeper understanding of the complex trade-offs and yielding well-informed decisions for practical applications.

4.2 Proposed Ranking Center-based Sampling in Multi-objective Population-based Algorithms

In this section, a center-based sampling is proposed utilizing the Objective-wise Average Rank Clustering algorithm to enhance multi-objective evolutionary algorithms' weakness in solving multi- and many-objective optimization problems. Several population-based algorithms, including PSO, DE, SHADE, LSHADE [51, 7, 1], have utilized the center-based sampling strategy to generate centroid points, either during population initialization or throughout the iterative optimization process. This study is inspired by the proposed center-based sampling strategy that successfully passed experiments on single-objective optimization problems described in the previous chapter. The proposed scheme would be at the population level, so, it could be utilized in a wide range of population-based optimization algorithms.

In this direction, this thesis proposes an Objective-wise Average Rank Clustering Center-based (OW-ARCC) approach by having a number of clusters NC from the candidate solutions for center-based sampling. This parameter can be translated to the aimed number of center-based solutions that are generated based on two or more individuals from the population. The proposed scheme considers not only the quality of candidate solutions but also their distribution in generating center-based solutions. This approach is pivotal in filling gaps in sparse regions of the objective space, which is a key advantage of the scheme. Although the quantity of participated solutions in calculating center-based solutions is not the main reason to increase exploration and exploitation, the distance between solutions is more important. Accordingly, the divided NC clusters are considered a potential neighborhood to find accelerator solutions by the center-based strategy.

In classic population-based algorithms, the generation of offspring by the crossover and mutation operators is set to be the same size as the initial population. In the proposed strategy, the offspring size is limited by $NP - NC$ to allocate enough space and budget for NC center-based solutions for each generation.

Contrary to the proposed clustering center-based sampling, which is for single-objective optimization, the aim is to cluster the solutions based on multi-objective values. Therefore, a simple average rank clustering approach divides individuals into clusters. The "average rank" term refers to a sorting-based ranking of candidate solutions based on multi-objective values. In other words, the candidate solutions are sorted based on the average of their ranks per objective in ascending order and separated into equal sizes of clusters. The detail of the proposed average ranking center-based sampling approach is explained in Algorithm 2.

Choosing a set of candidate solutions to generate high-quality center-based samples is challenging. In the following sections, the search for rightful candidate solutions that could result in more valuable solutions by center-based sampling will be discussed.

It is assumed that the centroid of individuals' decision variables could fill the gaps between the Pareto front set. Another advantage of using this strategy is to improve the quality of future population generations since the center-based solutions are injected into the offspring sets. Increasing the problem dimension, the probability of the center-based solution's closeness to the golden region could be increased [1]. In other words, the golden region guarantees the located solutions in this area have a better advantage to be closer to the optimal Pareto-front.

4.3 CEC-2017 Many-objective Benchmark Functions

Extensive experiments have been carried out to examine how the suggested algorithm performs when compared to its parent algorithm, NSGA-II. A series of MaF benchmark functions were proposed for solving Many-Objective Problems (MaOPs) in the CEC-2017 competition [52]. The benchmark suite offers 15 mainly scalable functions $MaF1$-$MaF15$ on M=5, M=10, and M=15 number of objectives tailored for experimental and comparative studies of evolutionary many-objective optimization (EMaO) algorithms [52].

In the experiments, M=2 and M=3 objectives are additionally added to the experimental plan to mitigate the multi-objective optimization along with many-objective cases. Because of benchmark design, $MaF8$, $MaF9$, and $MaF13$ are excluded in $M = 2$ objective problems; because they do not accommodate bi-objective cases. Several scalable continuous benchmark suites, such as DTLZ [53] and WFG [54], have been commonly used in many-objective optimization, and they utilized a baseline for the MaF functions development. The non-parametric Wilcoxon rank-sum statistical test with a 95% confidence interval is performed to prove the statistical significance of the experimental results. In the last row of each provided numerical table, $+/-/=$ indicates that the proposed schemes win in $+$ number of functions, lose in $-$ functions, and tie in $=$ function with respect to the Wilcoxon rank-sum test. Separate comparisons against the parent method, NSGA-II, are presented for each proposed scheme. The best result for each problem is highlighted in boldface. The control parameters in this study are set as follows:

- Population size: $NP = 100$,
- Maximum number of function evaluations: $Max_{NFE} = \max\{100000, 10000 \times M\}$,

Algorithm 2: Pseudo-code for the proposed center-based NSGA-II using objective-wise average rank clustering scheme. This algorithm works on two candidate solution sets, namely, offspring or Pareto-front. ArgSort is a function that returns sorted indices explained in Algorithm 3

```
Input  : $Max_{NFE}$: Maximum number of function calls, $NP$: Population size,
         $NC$: Number of Clusters, $F_{obj}$: Objective function , $M$: Number of
         Objectives,
Output: $Pareto$ : Pareto Front set
Initialize population $P$ in size $NP$ using uniform random distribution;
Calculate objective values, $P_{obj}$, for population $P$;
$NFE = NP$;
$S_C = \lceil \frac{NP-NC}{NC} \rceil$;
while $NFE < Max_{NFE}$ do
    // Offspring generation
    Create an empty $Q$ offspring population;
    while $Q.size == NP - NC$ do
        Select parents from $P$ using tournament selection;
        Apply crossover and mutation operators to create a child solution $Child$;
        Evaluate objectives for the $Child$ solution;
        Add $Child$ solution to $Q$ list;
    end
    // Center-based sampling
    $AveRank = \frac{\sum_{m=1}^{M} ArgSort(ArgSort(Q_{obj}(m))}{M}$;
    for $i \leftarrow 1$ to $NC$ do
        $start \leftarrow ((i-1) * S_C) + 1$;
        $end \leftarrow i * S_C$;
        $X_{c_i} = \frac{\sum_{j=start}^{end} x_j}{S_C}$;
        Add $X_{c_i}$ solution to $Q$ list;
        Evaluate objectives for the $X_{c_i}$ solution;
    end
    $NFE = NFE + NP$;
    Combine populations $P$ and $Q$ into $R$;
    // Selection operation
    Perform non-dominated sorting on $R$ to classify solutions into Pareto fronts;
    Calculate crowding distance for solutions in each Pareto front;
    Create a new population $P'$ by selecting solutions based on Pareto fronts and
    crowding distance;
    $P \leftarrow$P$'$;
end
```

Algorithm 3: Pseudo-code for ArgSort function used in Algorithm 2.

```
$indices \leftarrow [0, 1, 2, \ldots, \text{len}(arr) - 1]$    ▷ Create a list of indices
for $i \leftarrow 0$ to $len(arr) - 1$ do
    for $j \leftarrow i + 1$ to $len(arr) - 1$ do
        if $arr[indices[j]] < arr[indices[i]]$ then
            SWAP($indices[i]$, $indices[j]$) ▷ Swap indices if elements are out of order
        end
    end
end
return $indices$;
```

- Number of center-based solutions: $NC = 10$

Each proposed method was run 31 times independently. Table C.1 lists and explains the fundamental characteristics of functions.

In order to evaluate the performance of the proposed algorithm, the inverse generational distance (IGD) is a metric used in multi-objective optimization problems. Multi-objective optimization involves optimizing multiple conflicting objectives simultaneously, typically with no single solution that can optimize all objectives perfectly. Instead, the goal is to find a set of solutions that represents a trade-off between the objectives [55, 56, 57]. IGD metric is computed as the following equation:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i}}{n} \tag{4.1}$$

where n is the number of solutions in the Pareto-optimal set. The Pareto-optimal set represents the set of all solutions that cannot be improved in one objective without degrading at least one of the other objectives. Technically, IGD measures the convergence and distribution of PF solutions simultaneously. The smaller the IGD value, the better the approximation set is.

4.4 Case Study One: Ranking Center-based Non-dominated Sorting Genetic Algorithm II

In this section, one of the oldest population-based multi-objective algorithms is studied to investigate the proposed scheme. NSGA-II algorithm provides a population in size of NP, which generates offspring as the same size NP solutions to be replaced by the combination of whole $2 \times NP$ solutions. There is great potential to utilize center-based sampling in NSGA-II algorithm. The previously proposed center-based strategies were worked on the mutation operators (i.e., operation-level), such as GDE3. Unlike NSGA-II and other evolutionary algorithms, the mutation operators do not have a mathematical structure. To address this issue, the entire population that is generated by the NSGA-II could be used in a way that maximizes performance. In this study, the proposed two schemes for center-based sampling by using two various sets of candidate solutions to generate the center-based samples, namely, offspring and Pareto front. The "Offspring" set is the freshly generated solutions by NSGA-II operators, and the "Pareto front" set is the selected solutions from the collection of the previous population and fresh offspring based on the employed non-dominated sorting algorithm employed in NSGA-II. The experimenting proposed ranking center-based sampling on NSGA-II algorithm begins with population initialization followed by objective function computation. The iterative optimization loop is the main component of NSGA-II method, where the tournament selection, crossover, and mutation operators sequentially run to generate new offspring solutions. The proposed strategy is placed in this algorithm's flow after having new $NP - NC$ offspring solutions by the mentioned operators.

Similar to the single-objective optimization case study, the proposed idea is to find groups of candidate solutions. In the literature, there are many unsupervised ways to cluster data that have no labels. Let's consider there is no prior knowledge of which groups of candidate solutions would result in more effective center-based solutions. In this study, it is assumed solutions that have the closest distance with respect to their objective values, with a high probability, will result in solutions closer to optimal Pareto-front.

In order to begin from simple steps, two clustering algorithms are studied as follows: 1) K-Means clustering: K-Means is one of the well-known techniques that aims to partition a given M dimensional data into K clusters, where each data point belongs to the cluster with the nearest mean, commonly known as the centroid. and 2) Average Ranking clustering: This method calculates the average of solutions ranks in each objective criterion. It is an extended version of the proposed clustering center-based strategy for single-objective optimization. More information is explained in Section 4.2.

After predicting NC numbers of groups of $NP - NC$ candidate solutions, the center-based sampling strategy is used to calculate the centroid of each cluster. After adding new center-based candidate solutions into the remaining NC places in offspring to fill the NP members, the non-dominated sorting operator in NSGA-II is invoked to find Pareto-front set.

Experimental Results:

There are four schemes to be evaluated in this case study which are explained as follows:

1. Average Ranking clustering on Offspring solutions.
2. Average Ranking clustering on Pareto-front solutions.
3. K-Means clustering on Offspring solutions.
4. K-Means clustering on Pareto-front solutions.

By this end, one can understand which proposed scheme is the most successful in terms of statistical tests carried out by the Wilcoxon rank-sum algorithm.

The first stage of experiments is applied to CEC-2017 MaF benchmark functions with their original dimensions, which could change based on the number of objectives. Tables A.1, A.2, A.3, A.4, and A.5 in Appendix show the provided results on the five various objectives $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$, respectively.

Table 4.1: Accumulative comparison of each proposed scheme on original dimensions over all $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**			**M=15**			**Sum**		
	w	*l*	*t*	*w*	*l*	*t*	*w*	*l*	*t*	*w*	*l*	*t*	*w*	*l*	*t*	*w*	*l*	*t*
AveRank on Offspring	4	1	7	6	3	6	6	2	7	9	1	5	8	1	6	**33**	**8**	**31**
AveRank on ParetoFront	5	0	7	4	4	7	6	2	7	8	1	6	6	2	7	29	9	34
K-means on Offspring	5	1	6	2	3	10	4	2	9	7	1	7	7	2	6	25	9	38
K-means on ParetoFront	4	2	6	1	5	9	6	3	6	6	3	6	6	1	8	23	14	35

To summarize, Table 4.1 represents the accumulative comparison of the mentioned scheme in a single table based on the win, tie, and lose ratio. From this table, one can understand that the most promising scheme is the Average Ranking clustering technique on Offspring solutions resulting in 33 wins out of 72 functions and leading the competition by 4 functions than the second-ranked scheme, which has its clustering technique but used Pareto-front candidate solutions. This means the center-based solutions that are calculated from a set of selected solutions would not necessarily give the optimal performance. As can be seen from Table 4.1, the winning scheme is "Average Ranking on Offspring", it is decided to continue with this strategy for the rest of the experiments and label the proposed strategies as Ranking Center-based NSGA-II (RC-NSGA-II).

The second stage of experiments is applied to CEC-2017 MaF benchmark functions with constantly set $D = 1000$ dimensions, known as large-scale global optimization problems. Tables A.6, A.7, A.8, A.9, and A.10 in Appendix show the provided results on the five various objectives $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$, respectively.

To summarize, Table 4.2 represents the accumulative comparison of the mentioned scheme in a single table based on the win, tie, and lose ratio. From this table, one can understand that the most promising scheme is the Average Ranking clustering technique

Table 4.2: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-NSGA-II (Average Ranking on Offspring) scheme on $D = 1000$ dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**			**M=15**		
	w	l	t	w	l	t	w	l	t	w	l	t	w	l	t
RC-NSGA-II	**9**	2	1	**11**	1	1	**10**	0	3	**9**	3	1	**8**	3	2

on Offspring solutions resulting in 33 wins out of 72 functions and leading the competition by 4 functions than the second-ranked scheme, which has its clustering technique but used Pareto-front candidate solutions. This means the center-based solutions that are calculated from a set of selected solutions would not necessarily give the optimal performance.

In conclusion, the NSGA-II has been studied for four different applications of center-based sampling. Overall, the well-performed center-based scheme at the population level is Average Ranking clustering on Offspring solutions. This experiment gives valuable insight into further studies for other population-based algorithms.

4.5 Case Study Two: Ranking Center-based Non-dominated Sorting Genetic Algorithm III

In this section, the experiment center-based sampling strategy for multi-objective optimization is added to the NSGA-III algorithm. The center-based sampling is known to be a good solution for large-scale optimization problems to find an unknown solution searching on the centroid of search space. NSGA-III is designed to solve optimization problems with more than two conflicting objectives. In such problems, the goal is to find a set of solutions that simultaneously optimize multiple objectives, where improving one objective often leads to the degradation of others. This is known as the Pareto optimization. By taking this advantage, NSGA-III could outperform benchmark problems; however, it finds difficulty when the number of solution parameters increases naturally.

This is a normal reaction to large-scale optimization problems since many solutions must be tuned.

To address the issue efficiently and efficiently, a novel ranking multi-objective center-based sampling can provide solutions closer to golden regions. The suggested approach is the same one that was researched for the NSGA-II algorithm. The algorithm starts with an initial population that has NP candidate solutions. It is desired to have $NC\%$ of the population as center-based solutions. After generating the $NP - NC$ offspring solutions from the NP population, the ranking multi-objective center-based sampling approach takes the offspring solutions into calculating centroids based on derived groups. As the budget in each generation should be controlled equal to NP, the selection operator finds $NP - NC$ offspring generated by NSGA-III operators and NC center-based solutions accounting as NP for the population size. Similar to the research done on NSGA-II, the center-based solutions are calculated and added to the population to speed up NSGA-III's performance on issues involving numerous objectives. As the NSGA-III matches the solutions to the reference points, the center-based sampling potentially fills the gaps between the reference points that are not associated with any solutions.

Experimental Results:

In order to investigate the performance of the proposed algorithm versus NSGA-III, comprehensive experiments have been conducted. The proposed ranking multi-objective center-based NSGA-III was applied and tested on MaF benchmark functions proposed in the CEC-2017 multi-objective competition with M=2, 3, 5, 10, and 15 objectives.

Firstly, the originally proposed dimension setting for MaF problems will be analyzed. Looking at Tables A.11, A.12, A.13, A.14, and A.15, it can be seen that the proposed Ranking Center-based NSGA-III won 5, 6, 7, 4 problems out of 15 problems, respectively. To be fair, it needs more experiments to evaluate the performance of the proposed center-based scheme on NSGA-III.

Table 4.3: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-NSGA-III scheme on original dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**			**M=15**		
	w	l	t	w	l	t	w	l	t	w	l	t	w	l	t
RC-NSGA-III	**5**	5	3	**6**	6	3	**7**	4	4	**4**	6	5	**0**	12	3

Table 4.4: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-NSGA-III scheme on $D = 1000$ dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**			**M=15**		
	w	l	t	w	l	t	w	l	t	w	l	t	w	l	t
RC-NSGA-III	**9**	2	1	**12**	0	1	**10**	0	3	**8**	1	4	**0**	5	8

To consolidate, Table 4.3 shows the win, tie, lose ratio gathered from Tables A.11, A.12, A.13, A.14, and A.15.

In the second part, the MaF benchmark with all fixed 1000D dimensions will be analyzed. The purpose of this study is to show how the NSGA-III algorithm could handle large-scale many-objective problems. Looking at Table A.16 for M=2, the number of wins for the proposed scheme on NSGA-III rose to 9 whereas in Table A.11 was 5. This result clearly shows the effectiveness of a center-based scheme on large-scale optimization problems as it has been proven before. On the other sets of objectives such as M =3, 5, and 10, it can be seen in Tables A.16, A.17, A.18, and A.19 that number of wins increased significantly by 6, 3, and 4, respectively. Only on M=15 number of objectives in Table A.20, the proposed scheme did not change the results after increasing the D to 1000.

To consolidate, Table 4.4 shows the win, tie, lose ratio gathered from Tables A.16, A.17, A.18, and A.19, and A.20. The comparison between the results obtained from MaF benchmark optimization on large-scale and original dimensions shows that the proposed algorithm wins more than 50% of problems. The best result can be seen on M = 5 objectives, where the number of wins is 12 out of 13 problems meaning almost every functions where outperformed by the proposed scheme on NSGA-III algorithm.

4.6 Case Study Three: Ranking Center-based Multi-objective Evolutionary Algorithm based on Decomposition

In this section, MOEA/D algorithm is studied to analyze the impact of adding center-based solutions to its population. Each generation of MOEA/D consists of applying two parts, namely, generation and replacement of offspring solutions with the parents in the population. Considering the optimized population as a suitable source to find clusters, multiple center-based solutions from the internal clusters can be calculated. The proposed ranking multi-objective approach would be one of the ways to divide the population into NC clusters. In contrast to previous studies such as NSGA-II and NSGA-III, the selection operations are applied immediately after each offspring is generated. Therefore, the selected and center-based solutions construct the whole population.

Experimental Results:

By looking at Tables A.21 and A.22, one can say that the proposed concept almost failed on the M=2 and M=3 objectives. When the dimensions of solutions in the problem increase to 1000, it is harder to find the optimal Pareto front. By looking at Tables A.26 and A.27, it can be seen that the number of wins over the MOEA/D by the proposed ranking center-based MOEA/D are increased to 9 and 8 for M=2 and M=3, respectively.

In a further analysis of the higher number of objectives, such as M=5, M=10, and M=15, one can see a similar pattern in the performance of Ranking Center-based MOEAD. As a macro analysis, by looking at Tables A.23, A.24, and A.25, the number of wins over the parent algorithm, MOEAD, was 3 for M=5 and M=10 and 4 for M=15 which shows poor performance.

To recapitulate, an accumulative Table 4.5 is provided to show the win, tie, and

Table 4.5: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-MOEA/D scheme on original dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**			**M=15**		
	w	l	t	w	l	t	w	l	t	w	l	t	w	l	t
RC-MOEA/D	**0**	5	8	**2**	9	4	**3**	8	4	**3**	5	7	**4**	6	5

Table 4.6: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-MOEA/D scheme on $D = 1000$ dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**			**M=15**		
	w	l	t	w	l	t	w	l	t	w	l	t	w	l	t
RC-MOEA/D	**9**	1	5	**8**	3	2	**6**	6	1	**4**	4	5	**5**	6	2

lose counts from Tables A.21, A.22, A.23, A.24, and A.25 which are showing IGD results on the MaF benchmark problems with original dimensions. In addition, another accumulative Table 4.6 shows the IGD results on the MaF benchmark problems with $D = 1000$ dimensions. By comparing both tables, it can be understood that the proposed scheme has done much more better on large-scale search spaces. For instance, the proposed RC-MOEA/D resulted better than MOEA/D statistically on 9 problems with large dimensions of solutions, whereas it could not beat the MOEA/D on the original dimensions. It proves that center-based sampling enhanced the convergence speed of the MOEA/D algorithm and improved its exploration capability.

On the other similar tests on D=1000, the large-scale dimensionality could be addressed by center-based solutions. The total number of wins for M=5 is 6 in the contest against the parent algorithm shown in Table A.28.

4.7 Case Study Four: Ranking Center-based Strength Pareto Evolutionary Algorithm 2

This section proposes the Ranking Center-based SPEA2 (RC-SPEA2) algorithm for solving many-objective optimization benchmark problems. The proposed strategy is applied at the offspring generation stage, where the mating parents produce offspring. In order to keep the function evaluation budget for every generation, the offspring generation size must be controlled. To do that, the SPEA2's operators are controlled to generate $NP - NC$ solutions. Following that, the proposed ranking center-based approach is applied to the recently generated $NP - NC$ solutions. The ranking strategy finds the possible fixed-size NC clusters that could calculate the NC center-based solutions in the regions. Afterwards, the $NP - NC$ offspring solutions in addition to NC center-based solutions, create NP offspring solutions equal to the size of the population. At the end of each generation, the specific selection operator in SPEA2 algorithm is used to find the best NP population from $2 \times NP$ solutions as the combination of offspring and previous population sets. The key is to keep everything in the algorithm as simple as possible. Considering the simplicity, the performance of the proposed strategy has to be analyzed in order to find effectiveness.

Experimental Results:

The proposed algorithm was evaluated on CEC-2017 MaF benchmark problems with the originally proposed dimensions and the large-scale dimension of 1000. Five series of experiments have been conducted to compare the proposed schemes against the classical SPEA2. The main difference between these five experiments is the number of function objectives. As can be seen from Table A.31, the proposed scheme for SPEA2 algorithm outperforms the classical SPEA2 on M=2 objectives in five functions when the dimensions are in the original format. Despite the number of losses, the proposed strategy only

Table 4.7: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-SPEA2 scheme on original dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**		
	w	l	t	w	l	t	w	l	t	w	l	t
RC-SPEA2	**5**	7	1	**5**	9	1	**3**	10	2	**4**	10	1

Table 4.8: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-SPEA2 scheme on $D = 1000$ dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**		
	w	l	t	w	l	t	w	l	t	w	l	t
RC-SPEA2	**10**	1	1	**11**	0	2	**8**	3	2	**10**	2	1

lost the competition against SPEA2 in function MaF7 once. Considering the dimensions increase to 1000, Table A.35 shows that the number of wins rose to 10, resulting in proof of the effectiveness of the proposed algorithm on large-scale optimization problems. Similar to the experiments on original dimensions, the function MaF7 is likewise lost by the proposed ranking center-based SPEA2 against its parent algorithm.

Similar steps were taken for $M = 3$, $M = 5$, and $M = 10$ objectives where the proposed algorithm performed better on large-scale dimensions. Looking at Tables A.32, A.33, and A.34, the number of outperformed functions for the proposed ranking center-based SPEA2 against SPEA2 algorithm were 5, 3, and 4, respectively. Having the same order of number of objectives, one can see that the functions that were tied in the original dimensions are won in the dimension $D = 1000$. For instance, looking at Table A.36, on $M = 3$, the previously tied problems MaF1, MaF3-6, and MaF10-13 were won by the proposed ranking center-based SPEA2 algorithm in the contest against SPEA2 algorithm on $D = 1000$ dimensions. In the other Tables A.37 and A.38, the number of wins are shown as 8 and 10 out of 13 numbers of total functions, resulting in 5 and 6 improvements in the performance of the proposed RCSPEA2 against its parent SPEA2 algorithm, respectively.

To encapsulate the tables in the Appendix for the comparison between RC-SPEA2 and SPEA2, two accumulative Tables 4.8 and 4.7 are provided. The number of wins is doubled by leveraging the number of dimensions in the solutions of MaF benchmark problems. For instance, the proposed RC-SPEA2 won 10 times out of 13 problems from MaF benchmark problems, resulting in a 77% win ratio.

4.8 Case Study Five: Ranking Center-based Multi-objective Particle Swarm Optimization

This section studies the famous PSO algorithm on many-objective optimization problems for the proposed center-based strategy. Prior to the previous knowledge on operators in PSO, one knows that other than the population of particles, there are Pbest particles that are the best particles from each generation. Thus, there is a great opportunity to get the most advantage from these selected solutions to calculate center-based solutions. All that is needed is a technique to find clusters that could find well-separated solutions based on objective values. In this direction, the simple proposed ranking clustering method is applied to the Pbest solutions to find NC clusters. The proposed Ranking Center-based MOPSO begins with a randomly initialized Population. The algorithm is designed to generate exactly NP number of particles in each generation. In order to keep the size NP, the PSO's operators are controlled to generate $NP - NC$ number of particles. Later, the Pbest set is selected from the top $NP - NC$ Pbest set and $NP - NC$ newly generated particles, resulting in $NP - NC$ number of solutions. The top $NP - NC$ Pbest set is found by the proposed average ranking technique used to cluster the solutions. In the next stage, the Pbest solutions are used to be ranked by the proposed ranking strategy. The sorted order of particles in Pbest set will provide NC clusters. Each cluster is considered to find its center-based solution by the simple average calculation of the variables. After gathering NC center-based solutions and appending them to Pbest solutions, a second average ranking-based selection process is necessary to

Table 4.9: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-MOPSO scheme on original dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**		
	w	l	t	w	l	t	w	l	t	w	l	t
RC-MOPSO	**7**	5	1	**10**	5	0	**8**	4	3	**7**	4	4

sort Pbest solutions. In summary, the key here is to infuse the MOPSO algorithm with center-based solutions to enhance the quality and closeness of solutions to the optimal Pareto-front in any black-box optimization problems.

Experimental Results:

The effectiveness of the proposed RC-MOPSO algorithm is analyzed with comprehensive experiments on CEC-2017 MaF benchmark problems with the originally proposed dimensions and the large-scale dimension of 1000. Four sets of tests have been run to compare the proposed methods to the traditional MOPSO. The number of function objectives is the main distinction between these four experiments. Looking at Table A.39, for M=2, RC-MOPSO outperforms MOPSO algorithm for 7 functions (MaF3, MaF5, MaF6-MaF10, MaF14-15). While, for large-scale optimization, four more functions were won by the proposed algorithm resulting in 11 wins in 12 MaF functions (MaF1-MaF12, and MaF15). In general, one can see from the Tables A.40, A.41 and A.42 that the number of outperformed functions by the proposed RC-MOPSO algorithm is 10, 8, and 7 for M=3, M=5, and M=10 objectives, respectively. The intersection between them is 5 functions namely, MaF1, MaF6-8, and MaF14.

On the larger scale of dimensions, acceptable results are shown for M=2, 3, 5, and 10 objectives. To be specific, on M=3 objectives shown in Table A.44, it indicates that RC-MOPSO has a better result than the MOPSO algorithm on eight functions (MaF1-3, MaF5-7, MaF12, and MaF15). In summary, Tables A.45 and A.46 the number of wins for M=5 and M=10 is 6 which share the identical functions (MaF1, MaF5, MaF13-MaF15).

Table 4.10: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-MOPSO scheme on $D = 1000$ dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**		
	w	l	t	w	l	t	w	l	t	w	l	t
RC-MOPSO	**11**	1	0	**8**	3	2	**6**	0	7	**6**	1	6

4.9 Case Study Six: Ranking Center-based Generalized Differential Evolution 3

Within the context of population-based algorithms, it is impossible to skip the differential evolution algorithm, which has been successfully studied on large-scale single-objective optimization problems. This last section evaluates the proposed ranking center-based strategy using the Generalized Differential Evolution 3 algorithm. The proposed RC-GDE3 algorithm is designed to have a similar process to the previously proposed ranking center-based evolutionary algorithms such as NSGA-II and NSGA-III. In this scheme, the GDE3 begins the optimization by population initialization. The specialized DE operators for multi-objective optimization are responsible for generating offspring solutions from the current population. Before the greedy selection in GDE3, there was an excellent possibility of augmenting offspring solutions with center-based solutions. Sometimes, newly generated offspring solutions are placed in the out-of-population region, resulting in a broader search for space. Although they might be dominated by the non-dominated sorting and crowding distance methods, it is a must to take advantage of them. If a gap between solutions or a part of the solutions is generated close to each other, the center of regions should be investigated. One possible convenient and less expansive investigation method is the center-based strategy. Therefore, the offspring solutions are used to apply the proposed ranking center-based strategy. The ranking clustering approach is designed to sort the average of the solutions' ranks on every objective. The clusters are organized in equal sizes with a set of sorted solutions. In this study, the number of clusters is set as

Table 4.11: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-GDE3 scheme on original dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**		
	w	l	t	w	l	t	w	l	t	w	l	t
RC-GDE3	**8**	4	1	**8**	5	2	**7**	8	0	**2**	12	1

NC=10, meaning the aimed number of center-based solutions. Each cluster represents its center-based solution by calculating the average of consisting solutions. The simplicity makes the implementation fast and the cost even cheaper than the operators in the evolutionary algorithm.

Experimental Results:

This section provides a comprehensive analysis of the performance of the proposed RC-GDE3 algorithm compared to the GDE3 algorithm. The experiments were carried out on many-objective CEC-2017 MaF benchmark problems, which have 15 problems with five different objectives $M = 2$, 3, 5, 10, and 15. Although the MaF benchmark dynamically changes the dimension of the solution based on the given number of objectives, this thesis has investigated another series of problems with constant dimensions for the sake of large-scale global optimization. Therefore, the experiments consist of two parts as follows: 1) Original dimensions and 2) $D = 1000$ dimensions.

Starting with the originally proposed dimensions, Table 4.11 provides an accumulative win/tie/lose ratio over all the objectives. The results clearly demonstrate that the proposed RC-GDE3 algorithm has won more than 50% of the problems. Only in $M = 10$ objectives did the proposed RC-GDE3 show no significant improvement over the parent algorithm in 12 out of 15 problems. For more information on the details of separate problems results, Tables A.47, A.48, A.49, and A.50 presents mean and standard deviation of IGD values over 31 independent runs.

In the second part, the proposed RC-GDE3 and GDE3 algorithms are evaluated on

Table 4.12: Accumulative win (w), tie (t), lose (l) ratio of proposed RC-GDE3 scheme on $D = 1000$ dimensions over $M = 2$, $M = 3$, $M = 5$, $M = 10$, and $M = 15$ objective numbers.

Scheme	**M=2**			**M=3**			**M=5**			**M=10**		
	w	l	t	w	l	t	w	l	t	w	l	t
RC-GDE3	**7**	4	1	**8**	5	0	**6**	7	0	**5**	5	3

large-scale MaF optimization problems with $D = 1000$. The center-based sampling at population level shows enhancement when it is applied to large-scale problems. In this case study, Table 4.12 declares that the proposed RC-GDE3 successfully addressed its weaknesses on $M = 10$ objectives by decreasing the number of losses to 5 from 12 and increasing the number of wins to 5 from 2 against the GDE3 algorithm. The experiments carried out on other objectives show slight changes but no significance. Overall, the proposed RC-GDE3 won exactly 50% of problems on the larger scale of many-objective optimization problems where the dimension of solutions is $D = 1000$.

4.10 Summary

In this chapter, six population-based algorithms are studied, and the proposed scheme is applied to all the algorithms to solve CEC-2017 MaF benchmark problems. To summarize the results, Tables 4.13 and 4.14 provide a comprehensive outlook on the $w/t/l$ ratio.

Table 4.13: Summary of win, tie, and lose ratio collected from all tables for CEC-2017 MaF benchmark problems on $D = 1000$ dimensions.

Dimensions	M=2			M=3			M=5			M=10			M=15			Sum		
Algorithms	w	t	l	w	t	l	w	t	l	w	t	l	w	t	l	w	t	l
RC-NSGA-II vs. NSGA-II	4	7	1	6	6	3	6	7	2	**9**	5	1	**8**	6	1	**33**	31	8
RC-NSGA-III vs. NSGA-III	5	5	3	6	6	3	7	4	4	4	6	5	0	12	3	**22**	33	18
RC-MOEA/D vs. MOEA/D	0	5	8	2	9	4	3	8	4	3	5	7	4	6	5	12	33	28
RC-SPEA2 vs. SPEA2	5	7	1	5	9	1	3	10	2	4	10	1	3	10	2	**20**	46	7
RC-MOPSO vs. MOPSO	7	5	1	**10**	5	0	**8**	4	3	7	4	4	**8**	2	5	**40**	20	13
RC-GDE3 vs. GDE3	**8**	4	1	8	5	1	7	8	0	2	12	1	2	13	0	**27**	42	3

From Table 4.13, one can understand that the proposed scheme was successful with 30 problems out of 72 problems (40%) on average for original dimensions. The most successful algorithm is RC-MOPSO, with 40 wins and 13 losses, and the bottom-performing algorithm is RC-MOEA/D, with 12 wins and 28 losses out of 72 problems.

Table 4.14: Summary of win, tie, and lose ratio collected from all tables for CEC-2017 MaF benchmark problems on $D = 1000$ dimensions.

Dimensions	M=2			M=3			M=5			M=10			M=15			Sum		
Algorithms	w	t	l	w	t	l	w	t	l	w	t	l	w	t	l	w	t	l
RC-NSGA-II vs. NSGA-II	9	2	1	11	1	1	**10**	0	3	9	3	1	**8**	3	2	**47**	9	8
RC-NSGA-III vs. NSGA-III	9	2	1	**12**	0	1	**10**	0	3	8	1	4	0	5	8	**39**	8	17
RC-MOEA/D vs. MOEA/D	7	0	5	8	3	2	6	6	1	4	4	5	5	6	2	**30**	19	15
RC-SPEA2 vs. SPEA2	10	1	1	11	0	2	8	3	2	**10**	2	1	–	–	–	**39**	6	6
RC-MOPSO vs. MOPSO	**11**	1	0	8	3	2	6	0	7	6	1	6	–	–	–	**31**	5	15
RC-GDE3 vs. GDE3	7	4	1	8	5	0	6	7	0	5	5	3	–	–	–	**26**	21	4

On the other condition, when the dimensions are set to $D = 1000$, different patterns can be seen from Table 4.14. In this case, the proposed RC-NSGA-II showed a significantly better winning ratio in comparison to the tests on the original dimensions.

In conclusion, the proposed Ranking Center-based Sampling at the population level on multi- and many-objective optimization could outperform the classic population-based algorithm.

Chapter 5

Conclusion and Future Direction

5.1 Conclusion Remarks

Several works have investigated the importance of center-based sampling in the search space of black-box problems. This is because center-based solutions generally are closer to optimal solutions compared to randomly generated points. These characteristics make them valuable to be utilized in the optimization process. This thesis introduced an investigation into the utilization of center-based sampling at the population level. The center-based concept has shown to be promising in solving both single- and many-objective optimization problems. According to the fact that the advantage of center-based sampling was investigated previously, this thesis represents a comprehensive experimental investigation. The main contribution of this thesis was two proposed center-based schemes for both single- and many-objective population-based optimization algorithms.

For single-objective optimization, the Clustering Center-based strategy was proposed and tested in five different case studies, namely, the DE, GA, PSO, ABC, and CMA-ES. A single hyper-parameter is used for clustering center-based sampling to control number of center-based solutions NC injected into the population at each generation. Three NC values (i.e., 5, 10, 20) are investigated by the conducted experiments on the DE algorithm. Overall, the proposed algorithm showed close performances based on the win/tie/lose ratio. However, the error in $NC = 20$ was the highest. This stems from the smaller clusters, which aim to search the smaller regions similar to local search strategies. On the contrary, the $NC = 5$ center-based solutions do the global search since the size of clusters is five times smaller but could not beat the $NC = 10$ performance. Although the structure in the search space of the problem is inevitable, the average number of center-based solutions $NC = 10$ has an average performance on all the problems concluding based on experiments. Therefore, for the rest of the experiments, the parameter NC is set to 10, and crucial improvements were successfully achieved. In brief, the proposed algorithm outperformed the classical algorithms, such as DE and PSO, in almost all tests regarding solution accuracy on CEC 2017 benchmark functions, which consist of 29 functions with three dimensions $D = 30$, 50, and 100.

For many-objective optimization, the average Ranking Center-based sampling was proposed on six well-known specially designed optimization algorithms such as NSGA-II and NSGA-III, MOEA/D, SPEA2, MOPSO, and GDE3. The proposed average ranking algorithm was tested on the mentioned optimization algorithm and compared based on CEC-2017 many-objective optimization benchmark functions (MaF). Two series of experiments are carried out to analyze the performance of center-based sampling on the original dimensions proposed by benchmark and large-scale $D = 1000$ dimensions. In addition, each series was tested on five objectives to investigate different levels of hardness. The experimental results confirmed that the proposed center-based sampling scheme could significantly improve the performance of the classical algorithms. It is interesting to see these results indicate that utilizing center-based scheme at population level is promising when the dimension of solutions is large. Although large-scale problems can challenge the scalability of optimization algorithms, the proposed approach is applicable to any problem and performs efficiently regardless of degrading factors such as D dimensions and M objectives. This arises from the clustering technique that is specially designed for grouping solutions in objective space. Since the number of objectives is small (i.e., $M \leq 15$), the complexity and accuracy of clustering in this space are more effortless than in the decision space.

Overall, the presented findings demonstrated the effectiveness and even greater accomplishment of the suggested clustering center-based and average ranking center-based in the exploration and exploitation stages of the optimization process.

5.2 Future Direction

Although this thesis proposed several center-based sampling schemes at the population level, meta-heuristic optimization techniques still have a lot of room for development. In this thesis, the intention is to show the effectiveness of utilizing center-based solutions in the population directly by solving various problems. Mainly, the proposed algorithms in this thesis experimented on real-world mathematical benchmark optimization problems, and it has been successful. Due to the fact that meta-heuristic algorithms have shown good performance in a wide range of problems, such as neural network training, data mining, and pattern recognition, one can solve them utilizing the proposed center-based sampling at the population level in meta-heuristic algorithms.

Furthermore, there are other ways to generate center-based solutions efficiently and effectively to fill the gaps in the exploration and eliminate solutions stuck in local optima at the population level in the exploitation. Recommended approaches to try are as follows:

- It would be interesting to investigate the distance between solutions in a center-based infused and a classic optimizer. By this end, one can analyze the spread and spacing in the benefit of reaching closer to global optima in the search space.
- During optimization stages, the number of center-based solutions NC or size of clusters SC parameter can be changed adaptively based on convergence rate. This

will control the algorithm to switch the trade-off between global search (exploration) and local search (exploitation).

- There are many unsupervised clustering techniques to group unlabeled data using distance or any relational metrics. Inspired by them, one can utilize them in grouping individuals in a population for center-based sampling.
- A new adaptive parameter setting for algorithms to solve any black-box problem could be proposed by center-based sampling.

Bibliography

[1] H. Hiba, S. Rahnamayan, A. Asilian Bidgoli, A. Ibrahim, and R. khosroshahli, "A comprehensive investigation on novel center-based sampling for large-scale global optimization," *Swarm and Evolutionary Computation*, vol. 73, p. 101105, 2022.

[2] R. Khosrowshahli, S. Rahnamayan, and A. A. Bidgoli, "Clustering center-based differential evolution," in *2022 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2022.

[3] R. Khosrowshahli, S. Rahnamayan, and A. A. Bidgoli, "Ranking center-based nsga-ii," in *2023 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2023.

[4] R. Khosrowshahli and S. Rahnamayan, "Block differential evolution," in *2023 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2023.

[5] R. Khosrowshahli, S. Rahnamayan, A. A. Bidgoli, and M. Makrehchi, "Self-supervised learning using noisy-latent augmentation," in *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1–8, IEEE, 2023.

[6] S. Rahnamayan and G. G. Wang, "Center-based sampling for population-based algorithms," in *2009 IEEE Congress on Evolutionary Computation*, pp. 933–938, 2009.

[7] S. J. Mousavirad and S. Rahnamayan, "A novel center-based differential evolution algorithm," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[8] Z. Beheshti and S. M. Shamsuddin, "A review of population-based meta-heuristic algorithm," *International Journal of Advances in Soft Computing and Its Applications*, vol. 5, pp. 1–35, 03 2013.

[9] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[10] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, 1995.

[12] D. Karaboga and B. Basturk, "Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems," in *International fuzzy systems association world congress*, pp. 789–798, Springer, 2007.

[13] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, p. 341–359, Dec. 1997.

[14] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *2007 IEEE Congress on Evolutionary Computation*, pp. 4661–4667, 2007.

[15] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Gsa: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009. Special Section on High Order Fuzzy Sets.

[16] S. Mahdavi, S. Rahnamayan, and K. Deb, "Center-based initialization of cooperative co-evolutionary algorithm for large-scale optimization," pp. 3557–3565, 07 2016.

[17] S. Rahnamayan, G. G. Wang, and M. Ventresca, "An intuitive distance-based explanation of opposition-based sampling," *Applied Soft Computing*, vol. 12, no. 9, pp. 2828–2839, 2012.

[18] S. Tsutsui and A. Ghosh, "A study on the effect of multi-parent recombination in real coded genetic algorithms," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pp. 828–833, 1998.

[19] S. Sharma, T. K. Sharma, M. Pant, J. Rajpurohit, and B. Naruka, "Centroid mutation embedded shuffled frog-leaping algorithm," *Procedia Computer Science*, vol. 46, pp. 127–134, 2015. Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace Island Resort, Kochi, India.

[20] H. Hiba, S. Mahdavi, and S. Rahnamayan, "Differential evolution with center-based mutation for large-scale optimization," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2017.

[21] H. Hiba, A. Ibrahim, and S. Rahnamayan, "Large-scale optimization using center-based differential evolution with dynamic mutation scheme," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3189–3196, 2019.

[22] H. Hiba, M. El-Abd, and S. Rahnamayan, "Improving shade with center-based mutation for large-scale optimization," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1533–1540, 2019.

[23] S. Kukkonen and J. Lampinen, "Gde3: the third evolution step of generalized differential evolution," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 443–450 Vol.1, 2005.

[24] H. Salehinejad and S. Rahnamayan, "Effects of centralized population initialization in differential evolution," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2016.

[25] S. J. Mousavirad and S. Rahnamayan, "Cenpso: A novel center-based particle swarm optimization algorithm for large-scale optimization," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2066–2071, 2020.

[26] Y. Xu, L. Wang, and L. Li, "An effective hybrid algorithm based on simplex search and differential evolution for global optimization," in *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence* (D.-S. Huang, K.-H. Jo, H.-H. Lee, H.-J. Kang, and V. Bevilacqua, eds.), (Berlin, Heidelberg), pp. 341–350, Springer Berlin Heidelberg, 2009.

[27] R. A. Khanum and M. A. Jan, "Centroid-based initialized jade for global optimization," in *2011 3rd Computer Science and Electronic Engineering Conference (CEEC)*, pp. 115–120, 2011.

[28] S. J. Mousavirad and S. Rahnamayan, "A novel center-based differential evolution algorithm," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[29] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, p. 341, 1997.

[30] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992.

[31] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*, vol. 1, pp. 69–93, Elsevier, 1991.

[32] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pp. 69–73, 1998.

[33] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[34] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[35] I. Loshchilov, "Cma-es with restarts for solving cec 2013 benchmark problems," in *2013 IEEE Congress on Evolutionary Computation*, pp. 369–376, 2013.

[36] I. Loshchilov, "A computationally efficient limited memory CMA-ES for large scale optimization," *CoRR*, vol. abs/1404.5520, 2014.

[37] Q. Li, T. Liu, H. He, K. Wang, and S. Gao, "A covariance matrix adaptation evolution strategy-based manta ray foraging optimization," in *2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCISISIS)*, pp. 1–6, 2022.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[39] K. Deb and H. Jain, “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*, vol. 18, no. 4, pp. 577–601, 2014.

[40] I. Das and J. Dennis, “Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems,” *SIAM Journal on Optimization*, vol. 8, 07 2000.

[41] Q. Zhang and H. Li, “Moea/d: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[42] H. Ishibuchi, N. Akedo, and Y. Nojima, “Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems,” *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 264–283, 2015.

[43] E. Zitzler, M. Laumanns, and L. Thiele, “Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization,” vol. 3242, 01 2001.

[44] C. Coello Coello and M. Lechuga, “Mopso: a proposal for multiple objective particle swarm optimization,” in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No.02TH8600)*, vol. 2, pp. 1051–1056 vol.2, 2002.

[45] C. A. C. Coello, *Evolutionary algorithms for solving multi-objective problems.* Springer, 2007.

[46] X.-S. Yang, *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.

[47] R. Lewis and K. Smith-Miles, "A heuristic algorithm for finding cost-effective solutions to real-world school bus routing problems," *Journal of Discrete Algorithms*, vol. 52, pp. 2–17, 2018.

[48] B. Kiraz, A. A. Bidgoli, H. Ebrahimpour-komleh, and S. Rahnamayan, "A novel collective crossover operator for genetic algorithms," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 4204–4209, 2020.

[49] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization," November 2016. Available Online: https://github.com/P-N-Suganthan/CEC2017-BoundContrained.

[50] Sentewolf, "Concept of directional optimization in cma-es algorithm," 2008.

[51] S. J. Mousavirad and S. Rahnamayan, "Cenpso: A novel center-based particle swarm optimization algorithm for large-scale optimization," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2066–2071, 2020.

[52] R. Cheng, M. Li, Y. Tian, X. Zhang, Y. Jin, and X. Yao, "Benchmark functions for the cec'2017 competition on evolutionary many-objective optimization," 01 2017.

[53] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 1, pp. 825–830 vol.1, 2002.

[54] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.

[55] N. Hansen and A. Ostermeier, “Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation,” pp. 312 – 317, 06 1996.

[56] A. Iorio and X. Li, “Solving rotated multi-objective optimization problems using differential evolution,” vol. 3339, pp. 861–872, 12 2004.

[57] K. Deb, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints,” *Evolutionary Computation, IEEE Transactions on*, vol. 18, pp. 577–601, 08 2014.

Appendices

Appendix A

Multi-objective Optimization Table Results

A.1 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

Table A.1: Comparison of IGD results on RC-NSGA-II, KC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**AveRank on Offspring**	**AveRank on ParetoFront**	**K-means on Offspring**	**K-means on ParetoFront**	**NSGAII**
MaF1	4.6006e-3 (2.53e-4) =	4.5773e-3 (1.91e-4) =	4.6430e-3 (2.28e-4) =	**4.5372e-3 (1.39e-4) =**	4.6029e-3 (1.48e-4)
MaF2	2.6921e-3 (1.12e-4) +	2.6714e-3 (1.02e-4) +	2.6553e-3 (8.18e-5) +	**2.6452e-3 (9.62e-5)** +	2.7827e-3 (1.30e-4)
MaF3	2.4039e+0 (2.56e+0) =	3.1494e+0 (4.72e+0) =	4.1489e+0 (6.38e+0) =	**2.2373e+0 (3.74e+0)** +	5.8957e+0 (1.11e+1)
MaF4	7.1986e-1 (7.71e-1) =	8.6758e-1 (9.40e-1) =	7.7644e-1 (8.45e-1) =	9.1410e-1 (1.22e+0) =	**6.9684e-1 (8.75e-1)**
MaF5	7.8679e-1 (9.87e-1) =	**2.7255e-1 (6.79e-1)** +	3.3704e-1 (7.45e-1) =	5.2979e-1 (8.86e-1) =	4.6567e-1 (8.47e-1)
MaF6	**4.9647e-3 (1.50e-4) =**	4.9963e-3 (2.12e-4) =	4.9934e-3 (1.50e-4) =	4.9980e-3 (1.77e-4) =	4.9847e-3 (1.46e-4)
MaF7	**5.3882e-3 (2.02e-4) =**	5.3952e-3 (2.30e-4) =	5.4558e-3 (1.71e-4) +	2.0060e-2 (7.86e-2) -	1.9443e-2 (7.87e-2)
MaF10	1.4583e-1 (4.49e-2) -	1.3205e-1 (2.69e-2) =	1.5318e-1 (6.24e-2) -	1.2292e-1 (2.42e-2) =	**1.2071e-1 (2.61e-2)**
MaF11	**1.3546e-2 (7.02e-4) =**	2.0605e-2 (3.58e-2) =	1.3733e-2 (8.13e-4) =	2.2484e-2 (3.54e-2) -	1.3755e-2 (9.16e-4)
MaF12	2.3692e-2 (2.06e-3) +	2.3621e-2 (1.71e-3) +	2.3420e-2 (2.10e-3) +	**2.3279e-2 (1.99e-3)** +	3.8873e-2 (5.04e-2)
MaF14	**2.4873e+0 (1.14e+0)** +	2.6747e+0 (1.02e+0) +	3.4743e+0 (1.30e+0) +	4.7528e+0 (1.73e+0) =	4.9265e+0 (2.19e+0)
MaF15	**1.0273e-1 (1.66e-2)** +	1.2087e-1 (2.07e-2) +	1.0568e-1 (1.97e-2) +	1.5625e-1 (5.88e-2) +	3.4540e-1 (6.80e-2)
+/-/=	**4/1/7**	**5/0/7**	**5/1/6**	**4/2/6**	

Table A.2: Comparison of IGD results on RC-NSGA-II, KC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**AveRank on Offspring**	**AveRank on ParetoFront**	**K-means on Offspring**	**K-means on ParetoFront**	**NSGAII**
MaF1	**5.4298e-2 (1.91e-3)** +	5.5980e-2 (1.73e-3) +	5.7157e-2 (3.48e-3) =	**5.8115e-2 (3.39e-3) =**	5.7089e-2 (1.96e-3)
MaF2	**4.2579e-2 (1.88e-3)** +	4.2808e-2 (2.01e-3) +	4.6777e-2 (3.96e-3) =	**4.7838e-2 (3.68e-3) =**	4.7556e-2 (4.67e-3)
MaF3	3.8963e+0 (4.60e+0) =	4.8822e+0 (5.47e+0) -	3.3354e+0 (5.30e+0) =	**3.5522e+0 (6.21e+0) =**	**1.5844e+0 (2.41e+0)**
MaF4	1.1806e+0 (1.27e+0) =	1.4204e+0 (1.63e+0) =	1.5858e+0 (1.85e+0) =	1.2786e+0 (1.78e+0) =	**1.0877e+0 (1.53e+0)**
MaF5	8.8531e-1 (1.57e+0) =	**4.4365e-1 (8.25e-1) =**	5.9507e-1 (1.15e+0) =	6.0021e-1 (1.14e+0) =	4.4933e-1 (8.24e-1)
MaF6	**5.2832e-3 (2.44e-4) =**	5.2118e-3 (2.24e-4) =	**5.1325e-3 (1.91e-4) =**	5.2215e-3 (2.79e-4) =	5.2069e-3 (2.48e-4)
MaF7	**1.0632e-1 (8.15e-2) -**	9.8681e-2 (6.72e-2) -	8.6682e-2 (5.71e-3) -	1.0260e-1 (6.90e-2) -	**8.4633e-2 (4.93e-2)**
MaF8	8.8368e-2 (3.21e-3) +	**8.7842e-2 (3.39e-3)** +	9.5217e-2 (5.16e-3) +	1.0517e-1 (1.84e-2) =	**1.0281e-1 (1.08e-2)**
MaF9	**2.5958e-1 (1.91e-1)** +	2.9334e-1 (8.09e-2) =	3.2258e-1 (1.80e-1) =	3.3981e+0 (1.34e+1) -	3.2066e-1 (2.04e-1)
MaF10	2.4504e-1 (2.89e-2) -	2.4631e-1 (2.61e-2) -	2.3712e-1 (2.08e-2) -	**2.3780e-1 (1.81e-2) -**	**2.2783e-1 (2.14e-2)**
MaF11	**2.1751e-1 (1.02e-2) =**	2.1782e-1 (1.11e-2) =	2.2174e-1 (1.24e-2) =	2.2277e-1 (1.46e-2) =	2.1979e-1 (9.46e-3)
MaF12	**2.7844e-1 (1.48e-2) =**	**2.7786e-1 (1.14e-2) =**	2.8074e-1 (1.71e-2) =	2.8080e-1 (1.54e-2) =	2.8001e-1 (2.13e-2)
MaF13	**1.1704e-1 (1.03e-2)** +	1.1815e-1 (8.67e-3) =	1.1787e-1 (1.20e-2) =	1.3805e-1 (1.89e-2) -	1.2344e-1 (1.35e-2)
MaF14	2.3374e+0 (6.98e-1) -	2.6762e+0 (6.75e-1) -	2.7332e+0 (7.41e-1) -	3.4652e+0 (1.03e+0) -	**1.3536e+0 (3.80e-1)**
MaF15	4.8155e-1 (1.00e-1) +	**4.5533e-1 (1.09e-1)** +	5.8311e-1 (1.12e-1) +	6.9501e-1 (1.13e-1) +	1.0726e+0 (2.25e-1)
+/-/=	**6/3/6**	**4/4/7**	**2/3/10**	**1/5/9**	

A.2 Non-dominated Sorting Genetic Algorithm III (NSGA-III)

Table A.3: Comparison of IGD results on RC-NSGA-II, KC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**AveRank on Offspring**	**AveRank on ParetoFront**	**K-means on Offspring**	**K-means on ParetoFront**	**NSGAII**
MaF1	**1.6654e-1 (5.61e-3)** +	1.6779e-1 (3.49e-3) +	1.6968e-1 (5.16e-3) +	**1.6840e-1 (6.15e-3)** +	1.7819e-1 (6.30e-3)
MaF2	**1.3749e-1 (5.23e-3)** +	1.3955e-1 (5.06e-3) +	1.5036e-1 (6.41e-3) =	**1.4944e-1 (8.31e-3)** =	1.5250e-1 (5.42e-3)
MaF3	3.6171e+1 (3.36e+1) =	2.8817e+1 (2.90e+1) =	3.7225e+1 (4.96e+1) =	**6.1243e+1 (1.21e+2)** =	**5.3593e+1 (5.89e+1)**
MaF4	5.8642e+0 (9.57e+0) =	4.9291e+0 (6.57e+0) =	3.5524e+0 (3.73e+0) =	3.7017e+0 (4.09e+0) =	**3.8481e+0 (5.77e+0)**
MaF5	2.3913e+0 (8.68e-2) =	**2.4024e+0 (7.54e-2)** =	2.3790e+0 (7.66e-2) =	2.3845e+0 (8.98e-2) =	2.4148e+0 (8.27e-2)
MaF6	**5.9207e-3 (3.97e-4)** =	5.9053e-3 (3.42e-4) =	**5.9463e-3 (4.17e-4)** =	5.6865e-3 (3.90e-4) +	6.0225e-3 (4.72e-4)
MaF7	**4.9028e-1 (5.20e-2) -**	4.9707e-1 (6.73e-2) -	6.3081e-1 (9.11e-2) -	6.5540e-1 (9.49e-2) -	**3.9726e-1 (1.96e-2)**
MaF8	1.6034e-1 (7.47e-3) +	**1.5768e-1 (7.45e-3)** +	1.6680e-1 (9.44e-3) =	1.7201e-1 (1.27e-2) =	**1.6674e-1 (8.20e-3)**
MaF9	**2.1180e-1 (2.21e-2)** +	2.5939e-1 (3.68e-2) +	6.1725e-1 (2.30e-1) =	3.2270e-1 (5.63e-2) +	7.4073e-1 (2.70e-1)
MaF10	8.5562e-1 (6.41e-2) =	8.4580e-1 (4.60e-2) =	8.6592e-1 (5.56e-2) =	**8.8994e-1 (4.83e-2)** =	**8.7033e-1 (5.13e-2)**
MaF11	**8.9631e-1 (7.01e-2)** =	9.1964e-1 (8.89e-2) =	8.7463e-1 (8.74e-2) +	8.4443e-1 (6.51e-2) +	9.0834e-1 (7.19e-2)
MaF12	**1.3427e+0 (6.08e-2)** +	**1.3328e+0 (2.71e-2)** +	1.3341e+0 (4.67e-2) +	1.3504e+0 (5.16e-2) +	1.3892e+0 (5.78e-2)
MaF13	**2.1151e-1 (3.48e-2)** =	2.1021e-1 (2.71e-2) =	2.1234e-1 (3.18e-2) =	2.1939e-1 (2.75e-2) -	2.0556e-1 (3.07e-2)
MaF14	7.1105e+0 (2.41e+0) -	7.0370e+0 (2.30e+0) -	8.3346e+0 (1.95e+0) -	7.7027e+0 (2.24e+0) -	**5.4603e+0 (1.87e+0)**
MaF15	5.1320e+0 (1.82e+0) +	**4.6470e+0 (9.15e-1)** +	6.0906e+0 (1.57e+0) +	5.0719e+0 (1.67e+0) +	2.0693e+1 (4.21e+0)
+/-/=	**6/2/7**	**6/2/7**	**4/2/9**	**6/3/6**	

Table A.4: Comparison of IGD results on RC-NSGA-II, KC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**AveRank on Offspring**	**AveRank on ParetoFront**	**K-means on Offspring**	**K-means on ParetoFront**	**NSGAII**
MaF1	**2.8978e-1 (7.60e-3)** +	2.9450e-1 (8.94e-3) +	3.0719e-1 (1.08e-2) +	**3.0822e-1 (1.22e-2)** +	3.4261e-1 (1.21e-2)
MaF2	**1.8768e-1 (2.39e-3)** +	1.8966e-1 (2.59e-3) +	1.8801e-1 (2.76e-3) +	**1.8753e-1 (2.81e-3)** +	1.9520e-1 (2.96e-3)
MaF3	9.3789e+6 (4.39e+7) =	2.1858e+8 (1.21e+9) =	3.2792e+8 (1.81e+9) =	**1.5480e+6 (1.48e+6)** =	**2.5229e+6 (5.96e+6)**
MaF4	7.5158e+1 (4.27e+0) +	7.4916e+1 (4.43e+0) +	7.8650e+1 (5.77e+0) +	8.0248e+1 (5.00e+0) =	**8.1850e+1 (5.01e+0)**
MaF5	1.1843e+2 (9.05e+0) =	**1.2038e+2 (9.35e+0)** =	1.2206e+2 (9.93e+0) =	1.2155e+2 (1.05e+1) =	1.2179e+2 (9.58e+0)
MaF6	**7.0861e-3 (8.44e-4)** +	6.9623e-3 (6.24e-4) +	**6.8638e-3 (5.57e-4)** +	7.0511e-3 (7.10e-4) +	3.5678e-1 (1.55e-2)
MaF7	**6.8394e+0 (2.39e+0) -**	6.8830e+0 (2.34e+0) -	7.6721e+0 (2.22e+0) -	8.6273e+0 (2.44e+0) -	**2.4023e+0 (4.58e-1)**
MaF8	2.4413e-1 (1.02e-2) +	**2.3831e-1 (9.30e-3)** +	2.5331e-1 (1.44e-2) =	2.6509e-1 (1.80e-2) -	**2.5670e-1 (1.12e-2)**
MaF9	**4.2301e+1 (5.02e+1)** +	2.2772e+1 (3.17e+1) +	8.6968e+1 (7.91e+1) =	9.5329e+1 (1.04e+2) =	1.0536e+2 (1.00e+2)
MaF10	1.8777e+0 (1.22e-1) =	1.8954e+0 (8.15e-2) =	1.9150e+0 (9.50e-2) =	**1.9166e+0 (1.06e-1)** =	**1.8863e+0 (1.07e-1)**
MaF11	**1.8623e+0 (1.15e-1)** +	1.9035e+0 (1.30e-1) +	1.8507e+0 (1.07e-1) +	1.8031e+0 (8.83e-2) +	1.9720e+0 (1.10e-1)
MaF12	**5.5200e+0 (9.94e-2)** =	**5.5335e+0 (8.51e-2)** =	5.5047e+0 (8.81e-2) =	5.5142e+0 (8.29e-2) =	5.5357e+0 (6.62e-2)
MaF13	**2.9349e-1 (1.24e-1)** =	2.9241e-1 (1.32e-1) =	3.0300e-1 (2.46e-1) =	2.8988e-1 (8.28e-2) -	2.6596e-1 (7.62e-2)
MaF14	1.0936e+1 (2.53e+0) +	1.2281e+1 (2.39e+0) =	1.0440e+1 (2.88e+0) +	9.7060e+0 (2.62e+0) +	**1.2212e+1 (2.91e+0)**
MaF15	2.1830e+1 (4.85e+0) +	**1.9717e+1 (5.70e+0)** +	2.1727e+1 (4.32e+0) +	1.8749e+1 (4.67e+0) +	7.0761e+1 (9.91e+0)
+/-/=	**9/1/5**	**8/1/6**	**7/1/7**	**6/3/6**	

A.3 Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D)

Table A.5: Comparison of IGD results on RC-NSGA-II, KC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 15$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**AveRank on Offspring**	**AveRank on ParetoFront**	**K-means on Offspring**	**K-means on ParetoFront**	**NSGAII**
MaF1	**3.3370e-1 (7.78e-3)** +	3.3302e-1 (6.96e-3) +	3.4027e-1 (1.31e-2) +	**3.4033e-1 (1.44e-2)** +	4.0515e-1 (1.58e-2)
MaF2	**1.8986e-1 (2.36e-3)** +	1.9179e-1 (3.80e-3) +	1.8893e-1 (3.42e-3) +	**1.8844e-1 (3.69e-3)** +	1.9787e-1 (2.76e-3)
MaF3	2.1180e+6 (1.61e+6) =	3.2604e+7 (1.62e+8) =	1.7872e+6 (1.15e+6) +	**1.0065e+7 (4.31e+7)** =	**2.8972e+6 (1.64e+6)**
MaF4	2.2072e+3 (3.04e+2) +	3.2923e+3 (5.71e+3) -	2.6193e+3 (2.38e+3) -	2.3213e+3 (2.61e+2) +	**2.5007e+3 (3.52e+2)**
MaF5	2.6249e+3 (3.13e+2) =	**2.7117e+3 (2.72e+2)** =	2.6995e+3 (2.61e+2) =	2.7069e+3 (3.46e+2) =	2.7484e+3 (3.20e+2)
MaF6	**1.2154e-1 (1.68e-1)** +	5.6952e-2 (1.31e-1) +	**1.4282e-1 (2.12e-1)** +	1.4654e-1 (1.81e-1) +	5.0288e-1 (1.89e-1)
MaF7	**1.5122e+1 (3.59e+0) -**	1.4302e+1 (3.78e+0) -	1.6887e+1 (4.98e+0) -	1.7289e+1 (4.03e+0) -	**8.0516e+0 (2.83e+0)**
MaF8	3.0149e-1 (1.19e-2) +	**2.9987e-1 (1.47e-2)** +	3.1180e-1 (1.62e-2) =	3.2110e-1 (1.89e-2) =	**3.1290e-1 (1.66e-2)**
MaF9	**3.6951e+0 (1.83e+0)** +	1.2195e+0 (4.10e-1) +	7.0037e+0 (3.88e+0) +	1.8834e+0 (5.33e-1) +	1.6678e+1 (1.05e+1)
MaF10	2.5309e+0 (1.41e-1) =	2.5398e+0 (1.41e-1) =	2.4855e+0 (1.26e-1) +	**2.5137e+0 (1.27e-1)** =	**2.5480e+0 (1.38e-1)**
MaF11	**2.5714e+0 (3.57e-1)** +	2.8012e+0 (6.09e-1) =	2.6941e+0 (4.17e-1) =	2.7315e+0 (5.69e-1) =	2.7283e+0 (4.27e-1)
MaF12	**9.5840e+0 (1.14e-1)** =	**9.5777e+0 (1.57e-1)** =	9.5228e+0 (1.37e-1) =	9.5788e+0 (1.30e-1) =	9.5490e+0 (1.70e-1)
MaF13	**3.5026e-1 (1.74e-1)** =	2.9421e-1 (8.91e-2) =	3.0981e-1 (8.22e-2) =	3.7247e-1 (1.45e-1) =	3.7861e-1 (2.89e-1)
MaF14	8.7388e+0 (6.90e+0) =	1.0519e+1 (6.45e+0) =	1.0115e+1 (6.54e+0) =	9.5058e+0 (6.65e+0) =	**1.1906e+1 (7.02e+0)**
MaF15	4.0312e+1 (1.36e+1) +	**4.0166e+1 (1.44e+1)** +	4.0132e+1 (1.21e+1) +	3.5798e+1 (9.72e+0) +	9.3157e+1 (1.29e+1)
+/-/=	**8/1/6**	**6/2/7**	**7/2/6**	**6/1/8**	

A.4 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

Table A.6: Comparison of IGD results on RC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**M**	**D**	**RC-NSGA-II**	**NSGA-II**
MaF1	2	1000	**1.3873e+0 (1.50e-1)** +	1.0350e+1 (4.64e-1)
MaF2	2	1000	**2.1300e-1 (2.46e-2)** +	2.2239e+0 (1.43e-1)
MaF3	2	1000	**1.0613e+9 (1.29e+8)** +	2.4893e+9 (3.25e+8)
MaF4	2	1000	**4.5697e+4 (2.58e+3)** +	6.8504e+4 (2.28e+3)
MaF5	2	1000	**1.0260e+1 (3.26e+0)** +	3.8632e+1 (1.35e+1)
MaF6	2	1000	**1.8951e+2 (2.40e+1)** +	1.4675e+3 (6.88e+1)
MaF7	2	1000	3.7129e+0 (1.65e-1) -	**2.5756e+0 (9.52e-2)**
MaF10	2	1000	1.3623e+0 (3.46e-2) =	**1.3479e+0 (2.07e-2)**
MaF11	2	1001	**2.2327e-1 (3.56e-3)** +	3.2537e-1 (8.06e-3)
MaF12	2	1000	**9.0117e-2 (5.77e-3)** +	3.5236e-1 (1.04e-2)
MaF14	2	1000	2.7853e+1 (4.82e-1) =	**2.7323e+1 (1.31e+0)**
MaF15	2	1000	**1.3515e+0 (5.67e-2)** +	2.4780e+0 (1.06e-1)
+/=/-			**9/2/1**	

Table A.7: Comparison of IGD results on RC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**M**	**D**	**RC-NSGA-II**	**NSGA-II**
MaF1	3	1000	**2.2264e+0 (2.99e-1)** +	2.0563e+1 (1.21e+0)
MaF2	3	1000	**1.8572e-1 (1.46e-2)** +	2.5648e+0 (2.10e-1)
MaF3	3	1000	**2.1618e+9 (1.73e+8)** +	3.3773e+9 (2.88e+8)
MaF4	3	1000	**1.6755e+5 (7.27e+3)** +	2.0571e+5 (4.59e+3)
MaF5	3	1000	**1.4398e+1 (6.68e+0)** +	4.2596e+1 (6.73e+0)
MaF6	3	1000	**3.4094e+2 (4.20e+1)** +	1.8029e+3 (1.00e+2)
MaF7	3	1000	7.3090e+0 (2.70e-1) -	**3.6533e+0 (1.51e-1)**
MaF10	3	1000	**1.5005e+0 (1.06e-2)** =	1.5033e+0 (7.95e-3)
MaF11	3	1000	**3.7965e-1 (1.26e-2)** +	5.5835e-1 (1.12e-2)
MaF12	3	1000	**3.0921e-1 (1.15e-2)** +	8.7457e-1 (2.52e-2)
MaF13	3	1000	**5.1282e-1 (8.84e-3)** +	1.3187e+0 (9.34e-2)
MaF14	3	1000	**1.5115e+1 (4.75e+0)** +	2.3240e+1 (4.81e+0)
MaF15	3	1000	**3.2435e+0 (3.68e-1)** +	8.6008e+0 (1.01e+0)
+/=/-			**11/1/1**	

Table A.8: Comparison of IGD results on RC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	**M**	**D**	**RC-NSGA-II**	**NSGA-II**
MaF1	5	1000	**3.3294e+0 (4.00e-1)** +	5.5508e+1 (4.26e+0)
MaF2	5	1000	**2.4174e-1 (3.04e-2)** +	1.6747e+0 (7.10e-2)
MaF3	5	1000	**3.3863e+12 (5.27e+12)** +	1.1283e+13 (1.51e+13)
MaF4	5	1000	9.0179e+5 (3.65e+4) -	**7.3745e+5 (3.94e+4)**
MaF5	5	1000	**1.9528e+1 (4.60e+0)** +	5.0608e+1 (2.11e+1)
MaF6	5	1000	**4.3166e+2 (8.97e+1)** +	2.0024e+3 (8.88e+1)
MaF7	5	1000	9.5225e+0 (4.04e-1) -	**6.4213e+0 (3.47e-1)**
MaF10	5	1000	**1.9472e+0 (4.15e-3)** +	1.9584e+0 (6.37e-3)
MaF11	5	1000	**7.3703e-1 (2.89e-1)** +	7.9379e-1 (1.49e-1)
MaF12	5	1000	**1.2084e+0 (7.42e-3)** +	1.4752e+0 (2.67e-2)
MaF13	5	1000	**7.9862e-1 (3.49e-2)** +	3.4268e+0 (2.82e-1)
MaF14	5	1000	1.6776e+1 (1.60e+1) -	**1.5897e+1 (3.33e+0)**
MaF15	5	1000	**5.0806e+0 (3.32e-1)** +	1.4463e+1 (2.08e+0)
+/=/-			**10/0/3**	

Table A.9: Comparison of IGD results on RC-NSGA-II and NSGA-II algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	M	D	**RC-NSGA-II**	**NSGA-II**
MaF1	10	1000	**3.8474e+0 (4.29e-1)** +	6.5791e+1 (4.52e+0)
MaF2	10	1000	**6.5815e-1 (6.78e-2)** +	2.0970e+0 (1.16e-1)
MaF3	10	1000	3.3061e+14 (1.84e+15) =	**8.2314e+13 (4.41e+14)**
MaF4	10	1000	2.7978e+7 (2.90e+6) =	**2.7237e+7 (2.69e+6)**
MaF5	10	1000	**3.8233e+2 (1.06e+1)** +	3.8696e+2 (1.33e+1)
MaF6	10	1000	**4.4348e+3 (8.05e+2)** +	7.9249e+3 (2.57e+2)
MaF7	10	1000	3.4362e+1 (5.36e-1) -	**3.1959e+1 (5.75e-1)**
MaF10	10	1000	**3.0655e+0 (6.95e-2)** =	3.0961e+0 (7.06e-2)
MaF11	10	1001	**1.8205e+0 (1.14e-1)** +	2.1604e+0 (1.46e-1)
MaF12	10	1000	**5.4633e+0 (1.20e-1)** +	5.6223e+0 (7.69e-2)
MaF13	10	1000	**8.8884e-1 (3.80e-2)** +	3.8839e+0 (4.16e-1)
MaF14	10	1000	**2.0860e+1 (1.83e+0)** +	2.2129e+1 (2.86e+0)
MaF15	10	1000	**1.1830e+1 (1.34e+0)** +	6.6779e+1 (4.99e+0)
+/=/-			**9/3/1**	

Table A.10: Comparison of IGD results on RC-NSGA-II and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 15$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-II algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-II), respectively.

Problem	M	D	**RC-NSGA-II**	**NSGA-II**
MaF1	15	1000	**4.9303e+0 (5.75e-1)** +	9.0813e+1 (7.02e+0)
MaF2	15	1000	**6.2637e-1 (7.77e-2)** +	1.5123e+0 (7.10e-2)
MaF3	15	1000	1.4595e+15 (7.50e+15) =	**2.0363e+14 (7.10e+14)**
MaF4	15	1000	9.4763e+8 (1.11e+8) -	**8.4856e+8 (8.64e+7)**
MaF5	15	1000	7.1027e+3 (4.88e+2) =	**7.0953e+3 (4.71e+2)**
MaF6	15	1000	**5.3666e+3 (8.94e+2)** +	7.8544e+3 (2.55e+2)
MaF7	15	1000	5.3734e+1 (6.98e-1) -	**5.2127e+1 (1.05e+0)**
MaF10	15	1000	**3.9940e+0 (7.58e-2)** =	4.0229e+0 (7.88e-2)
MaF11	15	1000	**2.4307e+0 (1.09e-1)** +	2.8250e+0 (1.32e-1)
MaF12	15	1000	**9.4494e+0 (1.31e-1)** +	9.5569e+0 (1.85e-1)
v **MaF13**	15	1000	**9.9446e-1 (4.42e-2)** +	4.7838e+0 (4.61e-1)
MaF14	15	1000	**2.3751e+1 (1.75e+0)** +	2.5181e+1 (1.54e+0)
MaF15	15	1000	**1.7431e+1 (2.35e+0)** +	9.0426e+1 (5.13e+0)
+/=/-			**8/3/2**	

Table A.11: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	RC-NSGA-III	NSGA-III
MaF1	**3.5728e-3 (1.78e-6)** +	3.5742e-3 (4.11e-6)
MaF2	**2.0222e-3 (1.58e-5)** +	2.0375e-3 (2.41e-5)
MaF3	**6.8441e+0 (1.05e+1)** =	9.8601e+0 (1.87e+1)
MaF4	**1.0488e+0 (9.75e-1)** =	1.3028e+0 (1.24e+0)
MaF5	4.6298e-1 (8.48e-1) =	**3.3421e-1 (7.46e-1)**
MaF6	**4.0544e-3 (5.06e-5)** =	4.0716e-3 (5.73e-5)
MaF7	5.2873e-3 (1.16e-4) −	**5.1317e-3 (9.65e-5)**
MaF10	2.4242e-1 (8.31e-2) −	**2.0137e-1 (6.95e-2)**
MaF11	1.6261e-2 (1.16e-3) −	**1.5135e-2 (7.18e-4)**
MaF12	**2.2496e-2 (2.25e-3)** +	2.4027e-2 (2.47e-3)
MaF13	**9.3026e-2 (7.81e-3)** =	9.3280e-2 (7.62e-3)
MaF14	**3.0007e+0 (7.41e-1)** +	6.5509e+0 (2.35e+0)
MaF15	**1.4985e-1 (2.01e-2)** +	3.2177e-1 (8.12e-2)
$+/=/-$	5/5/3	

Table A.12: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	RC-NSGA-III	NSGA-III
MaF1	**5.8947e-2 (1.38e-3)** +	6.1022e-2 (2.01e-3)
MaF2	**3.5235e-2 (6.98e-4)** +	3.6081e-2 (6.59e-4)
MaF3	1.0151e+0 (1.79e+0) =	**8.1175e-1 (1.59e+0)**
MaF4	1.8959e+0 (2.10e+0) =	**1.4368e+0 (1.94e+0)**
MaF5	9.5065e-1 (1.05e+0) −	**3.5668e-1 (3.81e-1)**
MaF6	**1.3195e-2 (1.15e-3)** +	1.4793e-2 (1.39e-3)
MaF7	9.9779e-2 (7.83e-2) −	**8.6662e-2 (5.11e-2)**
MaF8	**1.0099e-1 (4.32e-3)** +	1.2599e-1 (2.16e-2)
MaF9	**6.4110e-2 (3.33e-3)** +	7.6340e-2 (1.99e-2)
MaF10	2.3655e-1 (2.55e-2) =	**2.2374e-1 (2.65e-2)**
MaF11	1.6300e-1 (1.14e-3) =	**1.6287e-1 (1.45e-3)**
MaF12	**2.2346e-1 (1.40e-3)** =	2.2398e-1 (1.93e-3)
MaF13	**9.3026e-2 (7.81e-3)** =	9.3280e-2 (7.62e-3)
MaF14	2.3541e+0 (8.31e-1) −	**1.0401e+0 (2.39e-1)**
MaF15	**5.4750e-1 (9.42e-2)** +	6.5509e-1 (1.46e-1)
+/ = /−	6/6/3	

A.5 Multi-objective Particle Swarm Optimization (MOPSO)

Table A.13: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**RC-NSGA-III**	**NSGA-III**
MaF1	**1.8327e-1 (1.08e-2)** +	2.2942e-1 (2.00e-2)
MaF2	**1.3315e-1 (3.46e-3)** +	1.4139e-1 (4.09e-3)
MaF3	**2.5176e-1 (4.62e-1)** +	1.6734e+0 (3.28e+0)
MaF4	6.0323e+0 (9.00e+0) =	**4.4369e+0 (1.84e+0)**
MaF5	2.8422e+0 (9.77e-1) =	**2.6496e+0 (9.07e-1)**
MaF6	**4.8674e-2 (1.56e-2)** +	6.0712e-2 (1.43e-2)
MaF7	3.9670e-1 (1.76e-2) -	**3.8623e-1 (1.31e-2)**
MaF8	**2.3958e-1 (2.43e-2)** +	2.5596e-1 (2.63e-2)
MaF9	5.3264e-1 (1.96e-1) =	5.9856e-1 (2.33e-1)
MaF10	4.9395e-1 (2.09e-2) -	**4.8215e-1 (1.77e-2)**
MaF11	**4.9907e-1 (3.58e-3)** +	5.0150e-1 (3.59e-3)
MaF12	1.1982e+0 (3.38e-3) =	1.2018e+0 (7.77e-3)
MaF13	**2.7021e-1 (3.25e-2)** +	3.0315e-1 (5.40e-2)
MaF14	1.1064e+1 (1.39e+1) -	**3.8482e+0 (1.79e+0)**
MaF15	1.9450e+0 (4.02e-1) -	**1.3665e+0 (2.13e-1)**
+/=/-	7/4/4	

Table A.14: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**RC-NSGA-III**	**NSGA-III**
MaF1	3.3121e-1 (7.01e-3) -	**3.1789e-1 (6.59e-3)**
MaF2	2.8549e-1 (2.89e-2) =	2.7148e-1 (2.41e-2)
MaF3	2.3430e+2 (3.69e+2) -	**1.0904e+2 (2.95e+2)**
MaF4	1.5698e+2 (1.54e+1) =	1.5261e+2 (1.52e+1)
MaF5	**1.3390e+2 (6.41e+0)** +	**1.3631e+2 (1.95e+0)**
MaF6	**6.9977e-2 (3.59e-2)** +	2.9861e-1 (1.22e-1)
MaF7	1.8147e+0 (3.14e-1) -	**1.6787e+0 (3.84e-1)**
MaF8	5.5985e-1 (1.08e-1) -	**4.6350e-1 (7.85e-2)**
MaF9	1.1120e+0 (3.82e-1) =	2.1575e+0 (2.19e+0)
MaF10	1.5092e+0 (2.01e-1) =	1.4887e+0 (1.48e-1)
MaF11	**1.3994e+0 (7.17e-2)** +	1.4854e+0 (1.06e-1)
MaF12	5.7891e+0 (1.24e-1) =	5.8117e+0 (1.07e-1)
MaF13	**3.6210e-1 (4.79e-2)** +	4.8659e-1 (1.11e-1)
MaF14	9.6251e+0 (4.34e+0) =	1.0541e+1 (5.52e+0)
MaF15	3.6923e+0 (7.53e-1) -	**2.3534e+0 (1.34e+0)**
+/=/-	4/6/5	

Table A.15: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 15$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**RC-NSGA-III**	**NSGA-III**
MaF1	**3.6804e-1 (9.56e-3)** =	3.7197e-1 (8.87e-3)
MaF2	3.2556e-1 (5.81e-2) =	**3.1535e-1 (4.63e-2)**
MaF3	1.3371e+2 (3.20e+2) -	**9.5378e+0 (2.07e+1)**
MaF4	5.6781e+3 (4.82e+2) =	**5.6120e+3 (3.23e+2)**
MaF5	4.7883e+3 (3.43e+1) =	**4.7518e+3 (1.95e+2)**
MaF6	**4.1976e-1 (1.80e-1)** =	4.5611e-1 (1.76e-1)
MaF7	**5.7169e+0 (6.59e-1)** =	5.8233e+0 (6.20e-1)
MaF8	9.2279e-1 (1.39e-1) -	**8.1734e-1 (1.25e-1)**
MaF9	**1.4823e+0 (4.37e-1)** =	1.7047e+0 (2.27e+0)
MaF10	**2.0779e+0 (1.38e-1)** =	2.1125e+0 (1.92e-1)
MaF11	2.8588e+0 (5.61e-1) =	**2.5945e+0 (6.05e-1)**
MaF12	**1.1589e+1 (2.49e-1)** =	1.1620e+1 (2.73e-1)
MaF13	**8.4357e-1 (1.87e-1)** =	8.7011e-1 (2.14e-1)
MaF14	8.9352e+0 (9.55e+0) -	**4.9105e+0 (5.20e+0)**
MaF15	**1.4825e+1 (2.30e+1)** =	2.8090e+1 (3.19e+1)
+/=/-	0/12/3	

Table A.16: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**M**	**D**	**RC-NSGA-III**	**NSGA-III**
MaF1	2	1000	**1.3873e+0 (1.50e-1)** +	1.0350e+1 (4.64e-1)
MaF2	2	1000	**2.1300e-1 (2.46e-2)** +	2.2239e+0 (1.43e-1)
MaF3	2	1000	**1.0613e+9 (1.29e+8)** +	2.4893e+9 (3.25e+8)
MaF4	2	1000	**4.5697e+4 (2.58e+3)** +	6.8504e+4 (2.28e+3)
MaF5	2	1000	**1.0260e+1 (3.26e+0)** +	3.8632e+1 (1.35e+1)
MaF6	2	1000	**1.8951e+2 (2.40e+1)** +	1.4675e+3 (6.88e+1)
MaF7	2	1000	3.7129e+0 (1.65e-1) -	**2.5756e+0 (9.52e-2)**
MaF10	2	1000	1.3623e+0 (3.46e-2) =	**1.3479e+0 (2.07e-2)**
MaF11	2	1001	**2.2327e-1 (3.56e-3)** +	3.2537e-1 (8.06e-3)
MaF12	2	1000	**9.0117e-2 (5.77e-3)** +	3.5236e-1 (1.04e-2)
MaF14	2	1000	2.7853e+1 (4.82e-1) =	**2.7323e+1 (1.31e+0)**
MaF15	2	1000	**1.3515e+0 (5.67e-2)** +	2.4780e+0 (1.06e-1)
+/=/-			9/2/1	

A.6 Generalized Differential Evolution 3 (GDE3)

Table A.17: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**RC-NSGA-III**	**NSGA-III**
MaF1	**2.1262e+0 (1.70e-1)** +	1.9255e+1 (9.34e-1)
MaF2	**1.8700e-1 (1.48e-2)** +	1.7412e+0 (1.10e-1)
MaF3	**3.0379e+11 (5.55e+11)** +	1.7226e+12 (3.19e+12)
MaF4	**1.4455e+5 (8.00e+3)** +	1.8739e+5 (6.51e+3)
MaF5	**1.1609e+1 (3.72e+0)** +	3.6955e+1 (7.11e+0)
MaF6	**2.8555e+2 (2.98e+1)** +	1.5395e+3 (8.39e+1)
MaF7	6.6946e+0 (2.74e-1) -	**4.0173e+0 (1.58e-1)**
MaF10	**1.4855e+0 (6.21e-3)** +	1.4948e+0 (5.66e-3)
MaF11	**3.2629e-1 (5.46e-2)** +	5.0494e-1 (5.07e-2)
MaF12	**2.5230e-1 (7.09e-3)** +	7.1521e-1 (1.99e-2)
MaF13	**5.5390e-1 (1.16e-2)** +	1.0874e+0 (6.20e-2)
MaF14	**9.4600e+0 (3.96e-1)** +	1.4111e+1 (5.74e-1)
MaF15	**2.4452e+0 (2.28e-1)** +	5.7141e+0 (4.17e-1)
+/=/-	12/0/1	

Table A.18: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**RC-NSGA-III**	**NSGA-III**
MaF1	**3.3294e+0 (4.00e-1)** +	5.5508e+1 (4.26e+0)
MaF2	**2.4174e-1 (3.04e-2)** +	1.6747e+0 (7.10e-2)
MaF3	**3.3863e+12 (5.27e+12)** +	1.1283e+13 (1.51e+13)
MaF4	9.0179e+5 (3.65e+4) -	**7.3745e+5 (3.94e+4)**
MaF5	**1.9528e+1 (4.60e+0)** +	5.0608e+1 (2.11e+1)
MaF6	**4.3166e+2 (8.97e+1)** +	2.0024e+3 (8.88e+1)
MaF7	9.5225e+0 (4.04e-1) -	**6.4213e+0 (3.47e-1)**
MaF10	**1.9472e+0 (4.15e-3)** +	1.9584e+0 (6.37e-3)
MaF11	**7.3703e-1 (2.89e-1)** +	7.9379e-1 (1.49e-1)
MaF12	**1.2084e+0 (7.42e-3)** +	1.4752e+0 (2.67e-2)
MaF13	**7.9862e-1 (3.49e-2)** +	3.4268e+0 (2.82e-1)
MaF14	1.6776e+1 (1.60e+1) -	**1.5897e+1 (3.33e+0)**
MaF15	**5.0806e+0 (3.32e-1)** +	1.4463e+1 (2.08e+0)
+/=/-	10/0/3	

Table A.19: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**RC-NSGA-III**	**NSGA-III**
MaF1	**2.6062e+0 (6.14e-1)** +	6.1401e+1 (1.05e+1)
MaF2	**3.7468e-1 (8.69e-2)** +	9.7274e-1 (9.98e-2)
MaF3	**8.2266e+13 (4.55e+14)** +	2.2717e+16 (7.78e+16)
MaF4	1.6419e+7 (5.50e+6) -	**1.3636e+7 (1.92e+6)**
MaF5	**2.0572e+2 (2.66e+1)** +	2.9441e+2 (1.69e+1)
MaF6	**7.0629e+2 (1.48e+2)** +	3.2723e+3 (6.26e+2)
MaF7	2.2101e+1 (1.14e+0) -	**1.9547e+1 (1.88e+0)**
MaF10	**3.0414e+0 (2.31e-1)** =	3.0853e+0 (3.77e-1)
MaF11	3.3883e+0 (1.19e+0) -	**1.8178e+0 (4.24e-1)**
MaF12	5.7813e+0 (4.70e-2) -	**5.7220e+0 (3.16e-2)**
MaF13	**1.0560e+0 (1.26e-1)** +	3.2901e+0 (3.37e-1)
MaF14	**1.4385e+1 (1.21e+0)** +	2.1146e+1 (4.23e+0)
MaF15	**1.0559e+1 (2.25e+0)** +	3.9101e+1 (7.79e+0)
+/=/-	8/1/4	

Table A.20: Comparison of IGD results on RC-NSGA-III and NSGA-III algorithms on CEC-2017 MaF benchmark problems for $M = 15$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-NSGA-III algorithm performs either better than, worse than, or similarly to the comparable algorithm (NSGA-III), respectively.

Problem	**RC-NSGA-III**	**NSGA-III**
MaF1	8.5229e+1 (9.95e+0) -	**6.0962e+1 (1.27e+1)**
MaF2	7.3751e-1 (7.95e-2) -	**6.8342e-1 (7.28e-2)**
MaF3	1.0909e+12 (1.41e+12) =	**6.0527e+11 (4.87e+11)**
MaF4	6.6522e+8 (1.09e+8) -	**5.3115e+8 (6.84e+7)**
MaF5	**6.9526e+3 (7.00e+2)** =	7.0519e+3 (4.38e+2)
MaF6	4.4805e+3 (2.58e+2) -	**4.0161e+3 (3.01e+2)**
MaF7	4.4521e+1 (1.64e+0) -	**3.9952e+1 (1.31e+0)**
MaF10	5.4024e+0 (8.23e-1) -	**4.9576e+0 (5.65e-1)**
MaF11	5.4818e+0 (1.92e+0) =	**4.7734e+0 (2.28e+0)**
MaF12	1.1653e+1 (3.59e-1) =	**1.1517e+1 (2.80e-1)**
MaF13	8.7234e+0 (6.06e-1) -	**5.1461e+0 (4.54e-1)**
MaF14	2.8487e+1 (4.65e+0) -	**2.6622e+1 (8.93e+0)**
MaF15	**7.5185e+1 (1.28e+1)** =	7.8864e+1 (1.34e+1)
+/=/-	0/5/8	

Table A.21: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	RC-MOEA/D	MOEA/D
MaF1	1.1524e-2 (1.11e-3) −	**3.5736e-3 (1.55e-6)**
MaF2	6.6455e-3 (9.03e-4) −	**2.9070e-3 (1.80e-4)**
MaF3	4.9682e+0 (6.30e+0) =	**4.1499e+0 (5.38e+0)**
MaF4	2.7700e+0 (2.41e+0) =	**2.0166e+0 (1.33e+0)**
MaF5	6.8516e-1 (9.28e-1) −	**4.6397e-1 (8.48e-1)**
MaF6	1.2742e-1 (1.50e-1) −	**4.3348e-3 (1.92e-4)**
MaF7	2.5575e-1 (2.24e-1) −	**1.5509e-1 (2.05e-1)**
MaF10	2.8585e-1 (5.81e-2) =	**2.6940e-1 (6.01e-2)**
MaF11	1.3431e-1 (8.28e-2) −	**8.3345e-2 (4.35e-2)**
MaF12	8.5296e-2 (6.01e-2) =	**7.0799e-2 (3.30e-2)**
MaF13	1.2549e-1 (3.59e-2) −	**1.1744e-1 (3.98e-2)**
MaF14	6.8495e-1 (8.32e-2) =	**6.6155e-1 (8.86e-2)**
MaF15	1.5319e-1 (1.59e-2) −	**1.4513e-1 (1.53e-2)**
+/ = /−	0/5/8	

Table A.22: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	RC-MOEA/D	MOEA/D
MaF1	**6.4846e-2 (7.79e-4)** +	7.0476e-2 (7.00e-6)
MaF2	**4.0335e-2 (1.20e-3)** =	4.0797e-2 (8.53e-4)
MaF3	3.2067e-1 (6.55e-1) −	**1.3748e-1 (3.95e-1)**
MaF4	1.6449e+0 (4.45e-1) =	**1.5557e+0 (4.53e-1)**
MaF5	1.9710e+0 (1.81e+0) −	**1.2008e+0 (1.57e+0)**
MaF6	**8.3431e-2 (1.58e-1)** +	9.1596e-2 (1.18e-1)
MaF7	1.8964e-1 (1.64e-1) −	**1.5502e-1 (2.50e-3)**
MaF8	2.6376e-1 (1.56e-1) =	**2.3413e-1 (1.25e-1)**
MaF9	1.6573e-1 (7.52e-2) =	**1.4906e-1 (6.68e-2)**
MaF10	3.2812e-1 (3.15e-2) =	**3.2742e-1 (2.76e-2)**
MaF11	2.5312e-1 (1.61e-2) =	**2.4869e-1 (1.63e-2)**
MaF12	**2.9859e-1 (4.59e-2)** =	3.0318e-1 (4.50e-2)
MaF13	1.2549e-1 (3.59e-2) −	**1.1744e-1 (3.98e-2)**
MaF14	**6.1416e-1 (1.28e-1)** =	6.2941e-1 (1.55e-1)
MaF15	4.5353e-1 (1.43e-1) =	**4.2397e-1 (1.30e-1)**
+/ = /−	2/9/4	

Table A.23: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	**M**	**D**	**RC-MOEA/D**	**MOEAD**
MaF1	5	14	**2.3646e-1 (3.88e-2)** =	2.3812e-1 (2.76e-2)
MaF2	5	14	**1.2479e-1 (1.10e-3)** +	1.3462e-1 (1.36e-3)
MaF3	5	14	1.3039e-1 (6.50e-3) -	**1.2699e-1 (7.51e-3)**
MaF4	5	14	1.1846e+1 (1.05e+0) =	**1.1568e+1 (1.16e+0)**
MaF5	5	14	**7.7119e+0 (1.35e+0)** +	9.6627e+0 (1.52e+0)
MaF6	5	14	2.1159e-1 (2.03e-1) =	**1.4620e-1 (1.97e-1)**
MaF7	5	24	**9.0907e-1 (2.25e-1)** +	1.0422e+0 (1.78e-1)
MaF8	5	2	4.0731e-1 (7.42e-2) -	**3.7369e-1 (5.40e-2)**
MaF9	5	2	2.0607e-1 (5.46e-2) -	**1.8614e-1 (6.06e-2)**
MaF10	5	14	8.3150e-1 (4.08e-2) =	**8.2719e-1 (1.60e-2)**
MaF11	5	14	8.2200e-1 (2.50e-2) =	**8.1327e-1 (1.87e-2)**
MaF12	5	14	**1.9697e+0 (1.77e-1)** =	1.9980e+0 (1.70e-1)
MaF13	5	5	2.3384e-1 (1.25e-1) =	**2.3083e-1 (1.07e-1)**
MaF14	5	100	8.2872e-1 (1.85e-1) =	**8.1190e-1 (1.89e-1)**
MaF15	5	100	7.6294e-1 (8.00e-2) -	**7.0407e-1 (9.18e-2)**
+/=/-			3/8/4	

Table A.24: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	M	D	RC-MOEA/D	MOEAD
MaF1	10	19	**5.2465e-1 (2.74e-3)** +	5.3479e-1 (6.40e-4)
MaF2	10	19	4.4124e-1 (7.23e-3) -	**3.7645e-1 (4.62e-3)**
MaF3	10	19	1.5775e-1 (7.66e-4) -	**1.4044e-1 (5.57e-4)**
MaF4	10	19	**4.5325e+2 (1.89e+1)** =	4.5450e+2 (1.92e+1)
MaF5	10	19	**2.4851e+2 (3.92e+1)** +	3.0350e+2 (1.57e+0)
MaF6	10	19	4.7652e-1 (2.70e-1) =	**3.1253e-1 (2.67e-1)**
MaF7	10	29	**1.9830e+0 (1.95e-1)** =	1.9868e+0 (2.41e-1)
MaF8	10	2	1.1703e+0 (5.15e-2) -	**1.1324e+0 (2.47e-2)**
MaF9	10	2	**8.5429e-1 (4.13e-2)** +	8.9088e-1 (5.45e-1)
MaF10	10	19	1.7426e+0 (5.59e-2) -	**1.6973e+0 (7.15e-2)**
MaF11	10	19	1.9109e+0 (1.74e-2) -	**1.8525e+0 (1.78e-2)**
MaF12	10	19	9.7714e+0 (6.00e-1) =	**9.5376e+0 (4.25e-1)**
MaF13	10	5	1.0730e+0 (1.37e-1) -	**9.6296e-1 (1.47e-1)**
MaF14	10	200	7.3506e-1 (1.24e-1) =	**7.0554e-1 (1.48e-1)**
MaF15	10	200	1.0598e+0 (4.98e-2) -	**1.0351e+0 (4.21e-2)**
+/=/-			3/5/7	

Table A.25: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 15$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	M	D	RC-MOEA/D	MOEAD
MaF1	15	24	**6.4385e-1 (5.06e-3)** +	6.4951e-1 (7.45e-3)
MaF2	15	24	**5.6430e-1 (3.59e-2)** =	5.7466e-1 (3.00e-2)
MaF3	15	24	1.3938e-1 (3.27e-4) -	**1.3657e-1 (5.01e-4)**
MaF4	15	24	**1.5941e+4 (9.55e+1)** =	1.5970e+4 (7.68e+1)
MaF5	15	24	**6.6127e+3 (8.39e+2)** +	7.3258e+3 (1.26e-1)
MaF6	15	24	6.7053e-1 (1.52e-1) -	**5.3000e-1 (2.40e-1)**
MaF7	15	34	6.2269e+0 (8.55e-1) =	**6.1873e+0 (8.04e-1)**
MaF8	15	2	1.6398e+0 (5.02e-2) -	**1.6116e+0 (3.02e-2)**
MaF9	15	2	4.0891e+0 (3.98e+0) -	**3.1558e+0 (3.91e+0)**
MaF10	15	24	**2.2771e+0 (5.93e-2)** =	2.3365e+0 (1.25e-1)
MaF11	15	24	2.4499e+0 (1.69e-2) -	**2.4397e+0 (1.85e-2)**
MaF12	15	24	**1.4546e+1 (2.19e+0)** +	1.6258e+1 (1.30e+0)
MaF13	15	5	1.3886e+0 (2.20e-1) =	**1.3458e+0 (2.22e-1)**
MaF14	15	300	8.8822e-1 (1.99e-1) =	**8.6211e-1 (2.31e-1)**
MaF15	15	300	**1.1900e+0 (2.11e-2)** +	1.2095e+0 (2.60e-2)
+/=/-			4/6/5	

Table A.26: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	**M**	**D**	**RC-MOEA/D**	**MOEA/D**
MaF1	2	1000	**1.3360e+1 (4.21e+0)** +	2.4576e+1 (2.34e+0)
MaF2	2	1000	**2.6920e+0 (5.81e-1)** +	5.1483e+0 (3.22e-1)
MaF3	2	1000	1.7917e+9 (7.47e+8) =	**1.6699e+9 (6.10e+8)**
MaF4	2	1000	5.8955e+4 (2.10e+3) =	**5.8258e+4 (2.02e+3)**
MaF5	2	1000	**8.9892e+1 (4.07e+1)** +	1.1031e+2 (2.99e+1)
MaF6	2	1000	**1.9415e+3 (5.43e+2)** +	3.3956e+3 (2.65e+2)
MaF7	2	1000	**6.4388e+0 (2.88e-1)** =	6.4829e+0 (2.92e-1)
MaF10	2	1000	1.9716e+0 (1.94e-1) =	**1.8581e+0 (2.90e-1)**
MaF11	2	1001	**5.0395e-1 (5.02e-2)** +	5.9681e-1 (5.13e-2)
MaF12	2	1000	**5.1728e-1 (7.58e-2)** +	7.5062e-1 (2.62e-2)
MaF14	2	1000	**2.8885e+1 (1.87e+1)** =	2.9967e+1 (1.57e+1)
MaF15	2	1000	**2.9174e+0 (4.70e-1)** +	3.8549e+0 (5.31e-1)
+/=/-			7/0/5	

Table A.27: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	**M**	**D**	**RC-MOEA/D**	**MOEAD**
MaF1	3	1000	**1.3205e+1 (3.67e+0)** +	2.6700e+1 (2.21e+0)
MaF2	3	1000	**1.5590e+0 (4.02e-1) +**	2.9583e+0 (2.68e-1)
MaF3	3	1000	9.0238e+8 (5.53e+8) =	**8.6429e+8 (4.40e+8)**
MaF4	3	1000	1.1024e+5 (6.74e+3) -	**9.8174e+4 (4.27e+3)**
MaF5	3	1000	**7.4704e+1 (3.40e+1)** +	9.9528e+1 (3.08e+1)
MaF6	3	1000	**1.5631e+3 (3.94e+2)** +	2.1619e+3 (2.24e+2)
MaF7	3	1000	6.9541e+0 (3.66e-1) -	**6.4942e+0 (2.76e-1)**
MaF10	3	1000	1.5944e+0 (2.37e-1) =	**1.5119e+0 (9.38e-2)**
MaF11	3	1000	**6.1485e-1 (6.96e-2)** +	7.3902e-1 (1.26e-1)
MaF12	3	1000	**5.9797e-1 (7.16e-2)** +	9.0155e-1 (4.27e-2)
MaF13	3	1000	**1.6687e+0 (4.67e-1)** +	2.3600e+0 (2.30e-1)
MaF14	3	1000	1.8176e+1 (1.52e+1) =	**1.7325e+1 (9.81e+0)**
MaF15	3	1000	**3.4593e+0 (9.86e-1)** +	4.8576e+0 (5.61e-1)
+/=/-			8/3/2	

Table A.28: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	**M**	**D**	**RC-MOEA/D**	**MOEAD**
MaF1	5	1000	**4.4803e+0 (2.61e+0)** +	3.8720e+1 (7.46e+0)
MaF2	5	1000	**5.2685e-1 (1.12e-1)** +	8.8446e-1 (1.99e-1)
MaF3	5	1000	2.5833e+8 (1.20e+8) =	**2.4383e+8 (8.77e+7)**
MaF4	5	1000	**1.9758e+5 (1.45e+4)** =	1.9777e+5 (8.93e+3)
MaF5	5	1000	**3.7363e+1 (2.88e+1)** =	3.8446e+1 (3.01e+1)
MaF6	5	1000	**9.4145e+2 (2.63e+2)** +	1.8145e+3 (1.48e+2)
MaF7	5	1000	1.1777e+1 (5.58e-1) -	**1.0741e+1 (4.80e-1)**
MaF10	5	1000	**1.9091e+0 (2.63e-2)** =	1.9189e+0 (2.87e-2)
MaF11	5	1000	**1.1502e+0 (2.90e-1)** +	1.3005e+0 (3.60e-1)
MaF12	5	1000	2.1178e+0 (1.70e-1) =	**2.0542e+0 (1.58e-1)**
MaF13	5	1000	**1.4990e+0 (2.08e-1)** +	1.5988e+0 (9.16e-2)
MaF14	5	1000	9.5632e+0 (3.88e+0) =	**8.0477e+0 (3.03e+0)**
MaF15	5	1000	**2.0296e+0 (7.27e-1)** +	4.7496e+0 (7.35e-1)
+/=/-			6/6/1	

Table A.29: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	**M**	**D**	**RC-MOEA/D**	**MOEAD**
MaF1	10	1000	**1.7874e+0 (1.49e+0)** +	3.4998e+0 (1.21e+0)
MaF2	10	1000	8.1769e-1 (2.27e-2) =	**8.1316e-1 (2.65e-2)**
MaF3	10	1000	**5.2618e+7 (9.62e+6)** +	6.5265e+7 (8.69e+6)
MaF4	10	1000	**3.4443e+6 (4.18e+5)** =	3.6668e+6 (9.71e+4)
MaF5	10	1000	3.2126e+2 (1.02e+2) -	**3.0463e+2 (5.19e+0)**
MaF6	10	1000	**2.7649e+2 (1.52e+2)** +	2.1398e+3 (2.80e+2)
MaF7	10	1000	8.6216e+0 (1.92e+0) -	**5.0190e+0 (1.31e+0)**
MaF10	10	1000	**2.4539e+0 (5.30e-2)** +	2.6150e+0 (3.48e-2)
MaF11	10	1001	2.7092e+0 (9.28e-1) -	**2.5938e+0 (9.63e-1)**
MaF12	10	1000	1.0153e+1 (5.49e-1) =	**1.0031e+1 (9.53e-1)**
MaF13	10	1000	1.4505e+0 (1.47e-1) -	**1.3874e+0 (1.31e-1)**
MaF14	10	1000	1.8216e+0 (5.88e-1) -	**1.5233e+0 (2.46e-1)**
MaF15	10	1000	1.1485e+0 (5.57e-2) =	**1.1188e+0 (6.70e-2)**
+/=/-			4/4/5	

Table A.30: Comparison of IGD results on RC-MOEA/D and MOEA/D algorithms on CEC-2017 MaF benchmark problems for $M = 15$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-MOEA/D algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOEA/D), respectively.

Problem	**M**	**D**	**RC-MOEA/D**	**MOEAD**
MaF1	15	1000	**3.4694e+1 (2.59e+1)** +	8.6747e+1 (5.38e+1)
MaF2	15	1000	8.4299e-1 (3.55e-2) =	**8.4345e-1 (2.73e-2)**
MaF3	15	1000	**4.8227e+7 (9.82e+6)** +	5.4048e+7 (4.40e+6)
MaF4	15	1000	**1.0788e+8 (1.24e+7)** +	1.1467e+8 (3.48e+6)
MaF5	15	1000	7.3275e+3 (8.96e+0) =	**7.3266e+3 (2.09e+0)**
MaF6	15	1000	**3.0354e+0 (6.54e-1)** +	5.8734e+0 (1.28e+0)
MaF7	15	1000	3.2694e+1 (1.82e+0) -	**2.9713e+1 (1.95e+0)**
MaF10	15	1000	**3.4555e+0 (6.02e-2)** +	3.5424e+0 (3.80e-2)
MaF11	15	1001	**4.4059e+0 (2.37e+0)** =	5.7150e+0 (3.61e+0)
MaF12	15	1000	1.3465e+1 (2.50e+0) =	**1.2743e+1 (2.40e+0)**
MaF13	15	1000	1.4893e+0 (3.33e-1) =	**1.4009e+0 (3.19e-1)**
MaF14	15	1000	1.4892e+0 (3.30e-1) -	**1.3249e+0 (2.72e-1)**
MaF15	15	1000	**1.2598e+0 (2.41e-2)** =	1.2616e+0 (3.20e-2)
+/=/-			5/6/2	

Table A.31: Comparison of IGD results on RC-SPEA2 and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	RC-SPEA2	SPEA2
MaF1	**3.7977e-3 (4.25e-5)** +	3.8295e-3 (4.52e-5)
MaF2	**2.1550e-3 (3.02e-5)** +	2.1950e-3 (3.59e-5)
MaF3	5.3834e+0 (5.51e+0) =	**2.0737e+0 (2.80e+0)**
MaF4	1.1552e+0 (9.55e-1) =	**9.0401e-1 (1.04e+0)**
MaF5	5.9218e-1 (9.21e-1) =	**3.3469e-1 (7.46e-1)**
MaF6	**4.1443e-3 (4.40e-5)** =	4.1740e-3 (7.67e-5)
MaF7	1.9149e-2 (7.88e-2) −	**4.8035e-3 (9.78e-5)**
MaF10	1.4519e-1 (3.44e-2) =	**1.4149e-1 (4.51e-2)**
MaF11	**1.2544e-2 (7.53e-4)** +	1.8574e-2 (3.60e-2)
MaF12	**2.1863e-2 (1.58e-3)** +	2.3216e-2 (1.64e-3)
MaF13	8.5291e-2 (5.21e-3) =	**8.3602e-2 (5.62e-3)**
MaF14	4.3286e+0 (1.93e+0) =	**4.0719e+0 (1.45e+0)**
MaF15	**8.6889e-2 (1.34e-2)** +	3.0780e-1 (4.78e-2)
$+/=/-$	5/7/1	

Table A.32: Comparison of IGD results on RC-SPEA2 and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	RC-SPEA2	SPEA2
MaF1	**4.1801e-2 (4.87e-4)** =	4.1889e-2 (4.60e-4)
MaF2	**3.1347e-2 (1.41e-3)** +	3.2230e-2 (1.25e-3)
MaF3	2.8111e+0 (9.15e+0) =	**1.0041e+0 (2.21e+0)**
MaF4	1.7216e+0 (2.41e+0) =	**7.8970e-1 (9.23e-1)**
MaF5	9.6332e-1 (1.04e+0) =	**8.8772e-1 (7.87e-1)**
MaF6	4.1167e-3 (3.66e-5) =	**4.0952e-3 (3.38e-5)**
MaF7	**6.1097e-2 (1.38e-3)** +	6.9384e-2 (5.15e-2)
MaF8	**6.6176e-2 (9.82e-4)** +	7.1633e-2 (1.81e-2)
MaF9	**9.7080e-1 (8.45e-1)** +	1.2472e+0 (5.71e-1)
MaF10	1.6839e-1 (1.53e-2) =	**1.6500e-1 (1.66e-2)**
MaF11	**1.7062e-1 (5.14e-3)** =	1.7109e-1 (3.54e-3)
MaF12	**2.1689e-1 (3.18e-3)** =	2.1818e-1 (4.55e-3)
MaF13	8.5291e-2 (5.21e-3) =	**8.3602e-2 (5.62e-3)**
MaF14	2.1181e+0 (6.24e-1) −	**1.1545e+0 (3.62e-1)**
MaF15	**3.5755e-1 (6.88e-2)** +	5.7638e-1 (1.83e-1)
+/ = /−	5/9/1	

Table A.33: Comparison of IGD results on RC-SPEA2 and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	**M**	**D**	**RC-SPEA2**	**SPEA2**
MaF1	5	14	**1.4218e-1 (1.82e-3)** =	1.4302e-1 (2.07e-3)
MaF2	5	14	**1.2603e-1 (2.61e-3)** +	1.3005e-1 (2.92e-3)
MaF3	5	14	**2.3405e+1 (4.11e+1)** =	3.3617e+1 (5.76e+1)
MaF4	5	14	**5.0269e+0 (6.92e+0)** +	7.4914e+0 (9.38e+0)
MaF5	5	14	2.8521e+0 (1.52e+0) =	**2.4214e+0 (6.59e-1)**
MaF6	5	14	4.1231e-3 (3.44e-5) -	**4.0862e-3 (2.66e-5)**
MaF7	5	24	3.0652e-1 (8.53e-3) =	**3.0413e-1 (8.53e-3)**
MaF8	5	2	1.1248e-1 (1.34e-3) =	**1.1193e-1 (1.38e-3)**
MaF9	5	2	**1.5639e-1 (9.44e-3)** =	1.6724e-1 (5.14e-2)
MaF10	5	14	6.2722e-1 (4.91e-2) =	**6.2077e-1 (2.41e-2)**
MaF11	5	14	**7.9526e-1 (4.38e-2)** =	7.9592e-1 (2.48e-2)
MaF12	5	14	**1.1523e+0 (2.01e-2)** =	1.1561e+0 (1.95e-2)
MaF13	5	5	**3.6954e-1 (3.19e-1)** =	3.7244e-1 (3.43e-1)
MaF14	5	100	2.7398e+1 (1.94e+1) -	**2.1048e+1 (4.39e+1)**
MaF15	5	100	**1.5371e+1 (6.30e+0)** +	2.6145e+1 (5.13e+0)
+/=/-			3/10/2	

Table A.34: Comparison of IGD results on RC-SPEA2 and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	**M**	**D**	**RC-SPEA2**	**SPEA2**
MaF1	10	14	**3.3290e-1 (1.09e-2)** +	3.4733e-1 (8.29e-3)
MaF2	10	14	**1.9436e-1 (4.36e-3)** =	1.9608e-1 (5.23e-3)
MaF3	10	14	3.0189e+12 (6.70e+11) =	**2.7884e+12 (6.35e+11)**
MaF4	10	14	**8.4517e+1 (3.36e+1)** =	8.7940e+1 (4.84e+1)
MaF5	10	14	2.3882e+2 (4.19e+1) =	**2.3544e+2 (2.69e+1)**
MaF6	10	14	**3.0326e-2 (1.46e-1)** +	1.7760e+2 (8.64e+1)
MaF7	10	24	**1.7838e+0 (7.61e-2)** =	1.8256e+0 (9.76e-2)
MaF8	10	2	1.7286e-1 (2.09e-3) -	**1.7131e-1 (2.07e-3)**
MaF9	10	2	3.7749e+0 (2.50e+0) =	**2.8577e+0 (1.94e+0)**
MaF10	10	14	**1.9541e+0 (1.03e-1)** =	1.9631e+0 (1.36e-1)
MaF11	10	14	**3.0215e+0 (4.23e-1)** +	3.2644e+0 (3.37e-1)
MaF12	10	14	**5.0735e+0 (8.78e-2)** =	5.0802e+0 (8.47e-2)
MaF13	10	5	9.3101e-1 (6.56e-1) =	**9.2233e-1 (6.04e-1)**
MaF14	10	100	6.6798e+4 (4.34e+4) =	**6.4511e+4 (4.33e+4)**
MaF15	10	100	**1.2673e+2 (9.20e+1)** +	2.0496e+2 (6.37e+1)
+/=/-			4/10/1	

Table A.35: Comparison of IGD results on RC-SPEA2 and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	**M**	**D**	**RC-SPEA2**	**SPEA2**
MaF1	2	1000	**1.7635e+0 (2.46e-1)** +	1.2474e+1 (7.77e-1)
MaF2	2	1000	**2.4410e-1 (2.53e-2)** +	2.7098e+0 (1.83e-1)
MaF3	2	1000	**8.3433e+8 (8.32e+7)** +	1.3073e+9 (9.15e+7)
MaF4	2	1000	**4.4902e+4 (3.22e+3)** +	6.2905e+4 (3.03e+3)
MaF5	2	1000	**8.5532e+0 (2.67e+0)** +	4.3527e+1 (3.01e+0)
MaF6	2	1000	**2.3846e+2 (2.54e+1)** +	1.8728e+3 (1.33e+2)
MaF7	2	1000	3.7760e+0 (1.47e-1) -	**3.1414e+0 (1.45e-1)**
MaF10	2	1000	1.2899e+0 (1.64e-2) =	**1.2856e+0 (1.67e-2)**
MaF11	2	1001	**2.3601e-1 (3.56e-2)** +	3.3775e-1 (3.88e-2)
MaF12	2	1000	**1.0184e-1 (5.93e-3)** +	3.5605e-1 (1.24e-2)
MaF14	2	1000	**2.7650e+1 (5.97e-1)** +	3.2533e+1 (1.26e+0)
MaF15	2	1000	**1.4547e+0 (7.63e-2)** +	2.9667e+0 (1.45e-1)
+/=/-			10/1/1	

Table A.36: Comparison of IGD results on RC-SPEA2 and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	**M**	**D**	**RC-SPEA2**	**SPEA2**
MaF1	3	1000	**2.5137e+0 (4.39e-1)** +	1.8652e+1 (1.05e+0)
MaF2	3	1000	**1.9217e-1 (1.49e-2)** +	1.8010e+0 (1.03e-1)
MaF3	3	1000	**2.7209e+16 (4.07e+16)** +	5.6525e+16 (8.62e+16)
MaF4	3	1000	**1.8226e+5 (6.27e+3)** +	2.1653e+5 (6.23e+3)
MaF5	3	1000	**1.2729e+1 (6.10e+0)** +	5.3260e+1 (3.33e+1)
MaF6	3	1000	**3.4998e+2 (4.15e+1)** +	1.6997e+3 (1.11e+2)
MaF7	3	1000	5.1275e+0 (2.10e-1) -	**4.0809e+0 (1.89e-1)**
MaF10	3	1000	**1.4876e+0 (4.64e-3)** +	1.4993e+0 (4.33e-3)
MaF11	3	1000	**3.4249e-1 (5.58e-2)** +	5.2015e-1 (6.81e-2)
MaF12	3	1000	**2.4616e-1 (9.63e-3)** +	8.2833e-1 (1.92e-2)
MaF13	3	1000	**5.0025e-1 (9.37e-3)** +	1.3041e+0 (7.17e-2)
MaF14	3	1000	7.6447e+1 (2.90e+1) -	**6.1018e+1 (3.45e+1)**
MaF15	3	1000	**3.5639e+0 (3.84e-1)** +	7.5624e+0 (4.71e-1)
+/=/-			11/0/2	

Table A.37: Comparison of IGD results on RC-SPEA2 and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	**M**	**D**	**RC-SPEA2**	**SPEA2**
MaF1	5	1000	**3.2532e+0 (5.04e-1)** +	3.6792e+1 (1.82e+0)
MaF2	5	1000	**3.4570e-1 (5.13e-2)** +	3.1315e+0 (2.01e-1)
MaF3	5	1000	3.0219e+18 (8.32e+17) -	**1.4337e+18 (6.73e+17)**
MaF4	5	1000	**1.0396e+6 (4.11e+4)** =	1.0500e+6 (3.78e+4)
MaF5	5	1000	**2.5859e+1 (1.04e+1)** +	1.1560e+2 (1.88e+1)
MaF6	5	1000	**6.5690e+3 (9.91e+2)** +	9.3592e+3 (2.93e+2)
MaF7	5	1000	1.0699e+1 (2.69e-1) -	**9.6472e+0 (4.04e-1)**
MaF10	5	1000	**2.0630e+0 (3.02e-2)** =	2.0715e+0 (4.11e-2)
MaF11	5	1000	**1.0026e+0 (4.65e-2)** +	1.1244e+0 (1.18e-1)
MaF12	5	1000	**1.1669e+0 (1.06e-2)** +	1.5834e+0 (2.06e-2)
MaF13	5	1000	**8.2947e-1 (4.63e-2)** +	3.7060e+0 (3.87e-1)
MaF14	5	1000	**3.0809e+2 (5.67e+2)** =	1.5144e+3 (2.49e+3)
MaF15	5	1000	**1.4326e+1 (4.37e+0)** +	3.6475e+1 (4.01e+0)
+/=/-			8/3/2	

Table A.38: Comparison of IGD results on RC-SPEA2 and SPEA2 algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-SPEA2 algorithm performs either better than, worse than, or similarly to the comparable algorithm (SPEA2), respectively.

Problem	**M**	**D**	**RC-SPEA2**	**SPEA2**
MaF1	10	1000	**3.8365e+0 (6.00e-1)** +	6.8807e+1 (4.16e+0)
MaF2	10	1000	**1.5377e+0 (1.78e-1)** +	2.8231e+0 (1.17e-1)
MaF3	10	1000	**1.6325e+20 (3.17e+19)** +	1.8728e+20 (1.52e+19)
MaF4	10	1000	3.2986e+7 (1.94e+6) -	**3.1575e+7 (1.50e+6)**
MaF5	10	1000	6.6461e+3 (2.91e+3) =	**5.3764e+3 (2.75e+3)**
MaF6	10	1000	**1.4538e+4 (3.16e+2)** +	1.4889e+4 (3.56e+2)
MaF7	10	1000	**3.0859e+1 (8.70e-1)** +	3.1989e+1 (7.54e-1)
MaF10	10	1000	3.3794e+0 (1.44e-1) =	**3.3647e+0 (1.27e-1)**
MaF11	10	1000	**3.4387e+0 (4.62e-1)** +	3.6988e+0 (3.15e-1)
MaF12	10	1000	**4.9618e+0 (9.30e-2)** +	5.1058e+0 (6.53e-2)
MaF13	10	1000	**9.6429e-1 (3.67e-2)** +	3.7154e+0 (3.45e-1)
MaF14	10	1000	**3.3601e+4 (2.56e+4)** +	5.7447e+4 (3.03e+4)
MaF15	10	1000	**4.4473e+1 (1.62e+1)** +	1.2074e+2 (4.23e+1)
+/=/-			10/2/1	

Table A.39: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	RC-MOPSO	MOPSO
MaF1	8.6902e-3 (1.45e-3) =	**8.2694e-3 (1.58e-3)**
MaF2	4.7510e-3 (3.34e-4) =	**4.7382e-3 (4.59e-4)**
MaF3	**6.6385e+3 (1.74e+4)** +	9.5149e+3 (2.66e+4)
MaF4	3.8629e+1 (5.71e+1) −	**3.1318e+1 (1.89e+1)**
MaF5	**8.7338e-1 (9.81e-1)** +	1.4524e+0 (9.07e-1)
MaF6	**2.4239e-2 (1.96e-2)** +	1.0304e-1 (1.09e-1)
MaF7	**2.9061e+0 (5.39e-1)** +	4.0847e+0 (4.45e-1)
MaF10	**9.6040e-1 (1.25e-1)** +	1.2524e+0 (1.56e-1)
MaF11	9.1455e-2 (1.69e-2) =	**8.3040e-2 (1.78e-2)**
MaF12	**5.0208e-2 (6.84e-3)** =	5.0843e-2 (8.75e-3)
MaF13	1.6083e-1 (5.96e-2) =	**1.4642e-1 (2.90e-2)**
MaF14	**1.5673e+1 (3.64e+0)** +	1.9885e+1 (5.04e+0)
MaF15	**5.7587e-1 (8.56e-2)** +	6.4320e-1 (1.04e-1)
+/ = /−	7/5/1	

Table A.40: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	RC-MOPSO	MOPSO
MaF1	**5.9444e-2 (6.09e-3)** +	7.9482e-2 (1.87e-2)
MaF2	**4.0361e-2 (3.41e-3)** +	5.9361e-2 (1.69e-2)
MaF3	2.0606e+4 (3.38e+4) =	**1.8946e+4 (2.30e+4)**
MaF4	**7.1533e+1 (1.34e+2)** +	1.0122e+2 (5.59e+1)
MaF5	9.6353e-1 (1.33e+0) =	**8.6677e-1 (1.09e+0)**
MaF6	**2.5944e-2 (1.73e-2)** +	1.5408e-1 (1.34e-1)
MaF7	**5.6061e+0 (9.46e-1)** +	6.4942e+0 (6.65e-1)
MaF8	**8.1866e-2 (6.09e-3)** +	9.6739e-2 (1.86e-2)
MaF9	**8.9934e-2 (1.43e-2)** +	1.2058e-1 (8.01e-2)
MaF10	**1.6066e+0 (1.41e-1)** =	1.6115e+0 (1.10e-1)
MaF11	2.5312e-1 (1.66e-2) =	**2.4529e-1 (1.78e-2)**
MaF12	**4.2339e-1 (7.03e-2)** +	5.7131e-1 (1.27e-1)
MaF13	1.6083e-1 (5.96e-2) =	**1.4642e-1 (2.90e-2)**
MaF14	**1.3628e+1 (5.06e+0)** +	3.3400e+1 (3.81e+1)
MaF15	**1.3870e+0 (1.83e-1)** +	2.2687e+0 (6.14e-1)
+/ = /−	10/5/0	

Table A.41: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	**M**	**D**	**RC-MOPSO**	**MOPSO**
MaF1	5	14	**4.0535e-1 (1.27e-2)** +	4.5129e-1 (1.65e-2)
MaF2	5	14	3.4662e-1 (1.58e-2) -	**3.0349e-1 (5.35e-3)**
MaF3	5	14	**3.5907e+11 (1.55e+11)** +	5.0637e+11 (2.56e+11)
MaF4	5	14	4.3047e+2 (6.08e+2) =	**3.2909e+2 (3.02e+2)**
MaF5	5	14	**3.8821e+0 (3.73e-1)** +	1.0240e+1 (5.93e+0)
MaF6	5	14	**1.9506e-2 (8.75e-3)** +	8.5333e-2 (6.60e-2)
MaF7	5	24	**1.9820e+1 (6.75e-1)** +	2.0793e+1 (5.92e-1)
MaF8	5	2	**6.3886e-1 (1.06e-1)** +	9.3616e-1 (2.85e-2)
MaF9	5	2	**3.7211e-1 (1.15e-1)** =	4.7598e-1 (3.75e-1)
MaF10	5	14	2.3121e+0 (2.23e-1) =	**2.2975e+0 (1.10e-1)**
MaF11	5	14	5.7624e+0 (1.68e+0) -	**4.1164e+0 (1.00e+0)**
MaF12	5	14	4.8846e+0 (3.89e-1) -	**4.4684e+0 (1.54e-1)**
MaF13	5	5	2.6649e-1 (4.94e-2) =	**2.5772e-1 (6.39e-2)**
MaF14	5	100	**4.3861e+2 (2.85e+2)** +	9.6716e+2 (8.38e+2)
MaF15	5	100	**5.1673e+0 (1.14e+0)** +	8.5343e+0 (2.10e+0)
+/=/-			8/4/3	

Table A.42: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	**M**	**D**	**RC-MOPSO**	**MOPSO**
MaF1	10	19	**5.3785e-1 (1.61e-2)** +	5.6910e-1 (1.65e-2)
MaF2	10	19	6.4254e-1 (2.22e-2) -	**5.4494e-1 (1.87e-2)**
MaF3	10	19	6.0966e+10 (7.52e+10) =	**4.6344e+10 (4.35e+10)**
MaF4	10	19	**7.4302e+3 (4.79e+3)** +	1.0455e+4 (5.00e+3)
MaF5	10	19	**7.2283e+2 (1.59e+2)** +	8.6376e+2 (6.71e+1)
MaF6	10	19	**1.8996e-2 (9.78e-3)** +	6.5784e-2 (4.28e-2)
MaF7	10	29	**4.7462e+1 (1.08e+0)** +	5.0241e+1 (7.70e-1)
MaF8	10	2	**1.3669e+0 (1.24e-1)** +	1.4890e+0 (7.01e-2)
MaF9	10	2	**8.3462e+1 (9.16e+1)** =	1.0528e+2 (1.16e+2)
MaF10	10	19	4.3612e+0 (6.87e-1) -	**3.4327e+0 (2.78e-1)**
MaF11	10	19	1.3712e+1 (2.43e+0) -	**9.6330e+0 (1.12e+0)**
MaF12	10	19	1.2424e+1 (1.41e+0) -	**1.1103e+1 (8.22e-1)**
MaF13	10	5	3.2755e-1 (7.71e-2) =	**3.1933e-1 (7.50e-2)**
MaF14	10	200	**1.7504e+3 (1.33e+3)** +	5.9574e+3 (3.98e+3)
MaF15	10	200	**1.1191e+1 (3.56e+0)** =	1.3388e+1 (4.60e+0)
+/=/-			7/4/4	

Table A.43: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	**M**	**D**	**RC-MOPSO**	**MOPSO**
MaF1	2	1000	**1.7645e+0 (2.58e-1)** +	2.9700e+0 (3.25e-1)
MaF2	2	1000	**3.4276e-1 (4.07e-2)** +	5.4728e-1 (4.60e-2)
MaF3	2	1000	**9.3880e+16 (2.19e+17)** +	3.8386e+17 (4.76e+17)
MaF4	2	1000	**9.6827e+4 (1.41e+3)** +	1.0077e+5 (1.70e+3)
MaF5	2	1000	**2.2912e+1 (1.20e+1)** +	3.4582e+1 (1.05e+1)
MaF6	2	1000	**2.5405e+2 (4.09e+1)** +	4.3670e+2 (4.14e+1)
MaF7	2	1000	**7.3584e+0 (6.14e-2)** +	7.4859e+0 (6.76e-2)
MaF10	2	1000	**1.5214e+0 (4.80e-2)** +	1.7313e+0 (1.23e-1)
MaF11	2	1001	**2.6066e-1 (6.33e-3)** +	2.6683e-1 (6.72e-3)
MaF12	2	1000	**1.3424e-1 (9.02e-3)** +	1.4287e-1 (9.08e-3)
MaF14	2	1000	3.0401e+1 (3.12e+0) =	**3.5691e+1 (1.60e+1)**
MaF15	2	1000	**1.6188e+0 (7.78e-2)** +	1.7689e+0 (1.00e-1)
+/=/-			11/1/0	

Table A.44: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and $D = 1000$ dimensions. The symbols “+”, “−”, and “=” indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	M	D	RC-MOPSO	MOPSO
MaF1	3	1000	**2.0684e+0 (1.76e-1)** +	3.6650e+0 (4.83e-1)
MaF2	3	1000	**1.7361e-1 (1.23e-2)** +	2.0247e-1 (1.73e-2)
MaF3	3	1000	**2.2304e+18 (2.32e+18)** +	6.1993e+18 (3.84e+18)
MaF4	3	1000	**2.8539e+5 (6.10e+3)** =	2.9232e+5 (2.06e+4)
MaF5	3	1000	**1.7305e+1 (4.00e+0)** +	2.1470e+1 (7.24e+0)
MaF6	3	1000	**1.7992e+2 (2.04e+1)** +	2.9500e+2 (2.30e+1)
MaF7	3	1000	**1.1077e+1 (1.54e-1)** +	1.1352e+1 (2.05e-1)
MaF10	3	1000	2.1056e+0 (2.02e-1) -	**1.7753e+0 (8.35e-2)**
MaF11	3	1000	3.8001e-1 (8.39e-3) -	**3.7264e-1 (8.50e-3)**
MaF12	3	1000	**4.0975e-1 (5.00e-2)** +	6.1179e-1 (1.29e-1)
MaF13	3	1000	5.6510e-1 (2.25e-2) =	**5.6375e-1 (1.99e-2)**
MaF14	3	1000	**1.7862e+1 (5.86e+0)** =	3.6619e+1 (5.65e+1)
MaF15	3	1000	**2.3983e+0 (1.89e-1)** +	4.1514e+0 (8.68e-1)
+/=/-			8/3/2	

Table A.45: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and $D = 1000$ dimensions. The symbols “+”, “−”, and “=” indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	M	D	RC-MOPSO	MOPSO
MaF1	5	1000	**2.8682e+0 (1.85e-1)** +	5.1509e+0 (9.38e-1)
MaF2	5	1000	4.4610e-1 (5.13e-2) -	**3.4719e-1 (3.39e-2)**
MaF3	5	1000	1.1870e+19 (4.15e+18) -	**8.6921e+18 (1.10e+18)**
MaF4	5	1000	1.7385e+6 (1.39e+5) -	**1.4029e+6 (9.19e+4)**
MaF5	5	1000	**2.7598e+1 (2.49e+0)** +	3.1525e+1 (3.21e+0)
MaF6	5	1000	**2.1625e+2 (4.91e+1)** +	2.7349e+2 (5.18e+1)
MaF7	5	1000	2.3884e+1 (1.93e-1) -	**2.3732e+1 (2.28e-1)**
MaF10	5	1000	2.8291e+0 (2.96e-1) -	**2.4151e+0 (8.42e-2)**
MaF11	5	1000	5.3384e+0 (1.95e+0) -	**4.0948e+0 (9.64e-1)**
MaF12	5	1000	5.0417e+0 (8.03e-1) -	**4.4366e+0 (1.52e-1)**
MaF13	5	1000	1.0855e+0 (1.01e-1) +	**1.2823e+0 (1.60e-1)**
MaF14	5	1000	**8.0667e+2 (5.29e+2)** +	2.1873e+3 (1.24e+3)
MaF15	5	1000	**6.2592e+0 (5.18e-1)** +	7.7721e+0 (1.04e+0)
+/=/-			6/0/7	

Table A.46: Comparison of IGD results on RC-MOPSO and MOPSO algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-MOPSO algorithm performs either better than, worse than, or similarly to the comparable algorithm (MOPSO), respectively.

Problem	**M**	**D**	**RC-MOPSO**	**MOPSO**
MaF1	10	1000	**4.5886e+0 (2.59e-1)** +	8.2858e+0 (1.32e+0)
MaF2	10	1000	7.5751e-1 (2.88e-2) -	**6.2877e-1 (2.65e-2)**
MaF3	10	1000	1.1142e+19 (4.22e+18) -	**7.1314e+18 (1.54e+18)**
MaF4	10	1000	5.1034e+7 (3.88e+6) -	**4.1998e+7 (2.67e+6)**
MaF5	10	1000	**8.7883e+3 (4.17e+3)** +	1.2657e+4 (5.11e+3)
MaF6	10	1000	2.7009e+2 (5.18e+1) =	**2.5158e+2 (3.94e+1)**
MaF7	10	1000	**4.9495e+1 (4.54e-1)** +	5.0802e+1 (2.61e-1)
MaF10	10	1000	5.0126e+0 (5.87e-1) -	**3.4709e+0 (1.63e-1)**
MaF11	10	1000	1.3860e+1 (1.60e+0) -	**1.1319e+1 (1.53e+0)**
MaF12	10	1000	1.1351e+1 (1.92e+0) -	**1.0489e+1 (6.17e-1)**
MaF13	10	1000	1.6369e+0 (1.83e-1) +	**2.0324e+0 (3.34e-1)**
MaF14	10	1000	**2.4281e+3 (1.62e+3)** +	8.9894e+3 (3.65e+3)
MaF15	10	1000	**1.1421e+1 (1.54e+0)** +	1.3063e+1 (2.38e+0)
+/=/-			6/1/6	

Table A.47: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively.

Problem	RC-GDE3	GDE3
MaF1	**4.2901e-3 (8.04e-5)** +	5.0630e-3 (2.11e-4)
MaF2	**2.3635e-3 (3.68e-5)** +	2.7615e-3 (8.38e-5)
MaF3	8.0533e+3 (6.08e+3) =	**7.1398e+3 (7.07e+3)**
MaF4	5.2575e+1 (3.27e+1) =	**4.0938e+1 (3.03e+1)**
MaF5	**1.4405e-2 (5.42e-4)** +	1.6247e-2 (6.89e-4)
MaF6	**4.7255e-3 (1.33e-4)** +	5.8279e-3 (4.22e-4)
MaF7	**9.1897e-1 (2.35e-1)** =	1.0283e+0 (2.51e-1)
MaF10	**1.3106e+0 (4.37e-2)** +	1.3467e+0 (3.91e-2)
MaF11	**2.2864e-2 (4.00e-3)** +	3.7777e-2 (5.11e-3)
MaF12	**3.1524e-2 (2.10e-3)** +	1.0299e-1 (9.71e-2)
MaF13	**1.2315e-1 (1.57e-2)** =	1.2389e-1 (1.55e-2)
MaF14	4.4379e+0 (2.52e+0) −	**2.6668e+0 (2.11e+0)**
MaF15	**2.6935e-2 (1.89e-3)** +	2.8322e-2 (1.81e-3)
+/ = /−	8/4/1	

Table A.48: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively

Problem	RC-GDE3	GDE3
MaF1	**6.4515e-2 (2.26e-3)** +	7.5869e-2 (3.57e-3)
MaF2	**5.8721e-2 (3.16e-3)** +	6.5899e-2 (2.67e-3)
MaF3	**4.8997e+3 (5.74e+3)** =	6.3701e+3 (6.30e+3)
MaF4	9.0046e+1 (7.27e+1) =	**6.2537e+1 (6.53e+1)**
MaF5	**3.0958e-1 (1.31e-2)** +	3.4269e-1 (1.45e-2)
MaF6	**4.7886e-3 (1.01e-4)** +	5.0841e-3 (2.07e-4)
MaF7	1.0009e+0 (3.49e-1) −	**8.1425e-1 (2.94e-1)**
MaF8	**7.6660e-2 (1.97e-3)** +	7.8002e-2 (2.79e-3)
MaF9	8.5672e-2 (9.47e-3) =	**8.3060e-2 (8.12e-3)**
MaF10	**1.5626e+0 (3.82e-2)** +	1.5927e+0 (4.40e-2)
MaF11	**2.2448e-1 (6.27e-3)** +	2.4387e-1 (1.13e-2)
MaF12	**2.6774e-1 (6.66e-3)** +	3.8429e-1 (4.06e-2)
MaF13	**1.2315e-1 (1.57e-2)** =	1.2389e-1 (1.55e-2)
MaF14	7.4042e+0 (1.91e+0) −	**4.4196e+0 (2.21e+0)**
MaF15	3.3073e-1 (2.24e-1) =	**2.7005e-1 (4.58e-2)**
+/ = /−	8/5/2	

Table A.49: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively.

Problem	**M**	**D**	**RC-GDE3**	**GDE3**
MaF1	5	14	**2.1035e-1 (1.14e-2)** +	2.2842e-1 (9.86e-3)
MaF2	5	14	**1.6075e-1 (7.47e-3)** =	1.6515e-1 (9.28e-3)
MaF3	5	14	4.3490e+4 (1.32e+4) =	**3.9986e+4 (1.16e+4)**
MaF4	5	14	4.2054e+2 (4.62e+2) =	**3.1606e+2 (3.29e+2)**
MaF5	5	14	**4.2222e+0 (3.49e-1)** =	4.2400e+0 (2.88e-1)
MaF6	5	14	**5.2849e-3 (1.59e-4)** +	5.6485e-3 (6.60e-4)
MaF7	5	24	**9.9285e-1 (1.33e-1)** =	9.9572e-1 (1.77e-1)
MaF8	5	2	**1.4765e-1 (5.38e-3)** +	1.5198e-1 (7.55e-3)
MaF9	5	2	**1.8546e-1 (1.94e-2)** +	3.5075e-1 (1.22e-1)
MaF10	5	14	2.1271e+0 (5.93e-2) =	**2.1063e+0 (4.32e-2)**
MaF11	5	14	**9.7606e-1 (6.94e-2)** +	1.0567e+0 (9.77e-2)
MaF12	5	14	**1.5174e+0 (3.13e-2)** +	1.6583e+0 (6.46e-2)
MaF13	5	5	**2.2977e-1 (3.40e-2)** =	2.3894e-1 (4.23e-2)
MaF14	5	100	**9.5883e-1 (4.51e-16)** =	9.5883e-1 (4.51e-16)
MaF15	5	100	**4.8889e+0 (1.55e+0)** +	1.5811e+1 (4.26e+0)
+/=/-			**7/8/0**	

Table A.50: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and original dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively.

Problem	**M**	**D**	**RC-GDE3**	**GDE3**
MaF1	10	19	**3.6415e-1 (2.07e-2)** =	3.6904e-1 (1.83e-2)
MaF2	10	19	1.9992e-1 (4.64e-3) =	**1.9941e-1 (2.96e-3)**
MaF3	10	19	**1.4340e+5 (3.06e+5)** =	1.6499e+5 (3.34e+5)
MaF4	10	19	1.0473e+4 (9.79e+3) =	**9.2835e+3 (7.42e+3)**
MaF5	10	19	1.4266e+2 (1.72e+1) =	**1.3681e+2 (1.47e+1)**
MaF6	10	19	**4.8117e-1 (1.79e-1)** =	5.2178e-1 (3.18e-1)
MaF7	10	29	**2.0668e+0 (1.23e-1)** =	2.0901e+0 (1.42e-1)
MaF8	10	2	2.2801e-1 (7.30e-3) =	**2.2646e-1 (7.32e-3)**
MaF9	10	2	**7.4091e+1 (1.01e+2)** +	1.3596e+2 (1.21e+2)
MaF10	10	19	**3.1243e+0 (7.68e-2)** =	**3.1376e+0 (8.28e-2)**
MaF11	10	19	2.5799e+0 (1.87e-1) =	**2.5401e+0 (2.12e-1)**
MaF12	10	19	5.6659e+0 (6.69e-2) =	**5.6574e+0 (7.15e-2)**
MaF13	10	5	3.5884e-1 (1.05e-1) -	**3.0243e-1 (6.57e-2)**
MaF14	10	200	1.9142e+1 (4.15e+0) =	**1.9065e+1 (6.02e+0)**
MaF15	10	200	**1.1517e+1 (3.20e+0)** +	3.3877e+1 (9.01e+0)
+/=/-			**2/12/1**	

Table A.51: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 2$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively.

Problem	**M**	**D**	**RC-GDE3**	**GDE3**
MaF1	2	1000	**7.7968e-1 (5.83e-2)** +	2.2609e+0 (3.45e-1)
MaF2	2	1000	**1.2616e-1 (1.06e-2)** +	3.4146e-1 (4.03e-2)
MaF3	2	1000	**1.3994e+8 (9.55e+7)** =	1.5280e+8 (7.44e+7)
MaF4	2	1000	**1.3048e+4 (6.48e+3)** =	1.4790e+4 (5.14e+3)
MaF5	2	1000	**7.2008e+0 (6.58e-1)** +	1.5025e+1 (1.94e+0)
MaF6	2	1000	**1.0307e+2 (6.83e+0)** +	2.9063e+2 (3.52e+1)
MaF7	2	1000	7.1154e+0 (6.17e-2) -	**7.0222e+0 (8.83e-2)**
MaF10	2	1000	1.5653e+0 (3.13e-2) =	**1.5631e+0 (2.70e-2)**
MaF11	2	1001	**2.3813e-1 (6.72e-3)** +	2.8230e-1 (1.03e-2)
MaF12	2	1000	**8.5283e-2 (9.58e-3)** +	1.3110e-1 (3.21e-2)
MaF14	2	1000	2.7677e+1 (4.54e-1) =	**2.7656e+1 (6.28e-1)**
MaF15	2	1000	**1.4631e+0 (4.89e-2)** +	1.8226e+0 (9.66e-2)
+/=/-			**7/4/1**	

Table A.52: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 3$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively.

Problem	M	D	**RC-GDE3**	**GDE3**
MaF1	3	1000	**1.3233e+0 (8.12e-2)** +	3.9565e+0 (5.16e-1)
MaF2	3	1000	**1.6223e-1 (1.29e-2)** +	3.6489e-1 (3.78e-2)
MaF3	3	1000	3.8450e+8 (2.80e+8) =	**2.8673e+8 (2.61e+8)**
MaF4	3	1000	**2.2289e+4 (2.06e+4)** =	2.3811e+4 (2.22e+4)
MaF5	3	1000	**1.9629e+1 (4.05e+0)** +	2.9367e+1 (7.79e+0)
MaF6	3	1000	**1.6011e+2 (2.20e+1)** +	3.5490e+2 (5.29e+1)
MaF7	3	1000	1.0044e+1 (1.56e-1) =	**1.0001e+1 (2.13e-1)**
MaF10	3	1000	1.6545e+0 (1.99e-2) =	**1.6522e+0 (2.14e-2)**
MaF11	3	1000	**3.9939e-1 (1.29e-2)** +	4.5679e-1 (1.28e-2)
MaF12	3	1000	**3.2982e-1 (1.45e-2)** +	4.1100e-1 (4.30e-2)
MaF13	3	1000	**5.0712e-1 (9.17e-3)** +	5.5401e-1 (2.71e-2)
MaF14	3	1000	**1.2288e+1 (1.19e+0)** =	1.3275e+1 (2.63e+0)
MaF15	3	1000	**3.2098e+0 (3.05e-1)** +	6.4576e+0 (6.31e-1)
+/=/-			**8/5/0**	

Table A.53: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 5$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively.

Problem	M	D	**RC-GDE3**	**GDE3**
MaF1	5	1000	**1.9912e+0 (1.62e-1)** +	7.2690e+0 (1.04e+0)
MaF2	5	1000	**3.2403e-1 (3.44e-2)** +	5.6046e-1 (5.38e-2)
MaF3	5	1000	**5.9043e+8 (1.08e+8)** =	6.1131e+8 (3.65e+7)
MaF4	5	1000	1.0350e+5 (9.86e+4) =	**7.3643e+4 (7.99e+4)**
MaF5	5	1000	**1.3944e+2 (3.97e+1)** =	1.5961e+2 (3.84e+1)
MaF6	5	1000	**2.3333e+3 (9.59e+2)** =	2.5913e+3 (1.20e+3)
MaF7	5	1000	1.6597e+1 (3.15e-1) =	**1.6411e+1 (4.00e-1)**
MaF10	5	1000	2.1797e+0 (3.45e-2) =	**2.1772e+0 (3.36e-2)**
MaF11	5	1000	**1.1662e+0 (6.58e-2)** +	1.3709e+0 (1.17e-1)
MaF12	5	1000	**1.5692e+0 (7.73e-2)** +	1.7232e+0 (1.03e-1)
MaF13	5	1000	**8.7027e-1 (5.46e-2)** +	1.0047e+0 (7.07e-2)
MaF14	5	1000	**2.9559e+1 (3.78e+0)** =	3.0207e+1 (2.26e+0)
MaF15	5	1000	**6.5588e+0 (8.69e-1)** +	1.6044e+1 (2.90e+0)
+/=/-			**6/7/0**	

Table A.54: Comparison of IGD results on RC-GDE3 and GDE3 algorithms on CEC-2017 MaF benchmark problems for $M = 10$ number of objectives and $D = 1000$ dimensions. The symbols "+", "−", and "=" indicate that the RC-GDE3 algorithm performs either better than, worse than, or similarly to the comparable algorithm (GDE3), respectively.

Problem	M	D	**RC-GDE3**	**GDE3**
MaF1	10	1000	**2.9761e+0 (3.00e-1)** +	1.1021e+1 (1.47e+0)
MaF2	10	1000	**5.5679e-1 (5.38e-2)** +	7.8419e-1 (8.34e-2)
MaF3	10	1000	1.1043e+9 (1.58e+9) =	**7.7763e+8 (4.04e+8)**
MaF4	10	1000	7.0026e+6 (6.20e+6) -	**4.3496e+6 (1.71e+6)**
MaF5	10	1000	**4.1996e+2 (4.93e+1)** =	4.2801e+2 (6.61e+1)
MaF6	10	1000	2.6267e+3 (1.33e+3) -	**1.9944e+3 (1.66e+3)**
MaF7	10	1000	3.3213e+1 (1.02e+0) -	**3.2642e+1 (8.11e-1)**
MaF10	10	1000	3.2264e+0 (4.23e-2) =	**3.2146e+0 (4.27e-2)**
MaF11	10	1001	**2.7525e+0 (1.33e-1)** =	2.7593e+0 (1.40e-1)
MaF12	10	1000	**5.6127e+0 (6.27e-2)** +	5.6784e+0 (5.40e-2)
MaF13	10	1000	**1.1019e+0 (6.19e-2)** +	1.4002e+0 (1.20e-1)
MaF14	10	1000	**2.9133e+1 (3.00e+0)** =	2.9566e+1 (2.99e+0)
MaF15	10	1000	**1.0016e+1 (8.90e-1)** +	2.1527e+1 (3.62e+0)
+/=/-			**5/5/3**	

Appendix B

CEC-2017 Single-Objective Real-Parameter Numerical Optimization

Table B.1: Summary of the CEC'17 test functions. HyF is an abbreviation of Hybrid Functions.

	No.	Functions	Fi*=F;(x*)
Unimodal	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Zakharov Function	200
Simple Multimodal Functions	3	Shifted and Rotated Rosenbrock's Function	300
	4	Shifted and Rotated Rastrigin's Function	400
	5	Shifted and Rotated Expanded Scaffer's F6 Function	500
	6	Shifted and Rotated Lunacek Bi_Rastrigin Function	600
	7	Shifted and Rotated Non-Continuous Rastrigin's Function	700
	8	Shifted and Rotated Levy Function	800
	9	Shifted and Rotated Schwefel's Function	900
Hybrid Functions	10	Hybrid Function I (f_3, f_4, and f_5)	1000
	11	Hybrid Function 2 (f_{11}, f_{10}, and f_1)	1100
	12	Hybrid Function 3 (f_1, f_4, and f_7)	1200
	13	Hybrid Function 4 (f_{11}, f_{13}, f_{20}, and f_5)	1300
	14	Hybrid Function 5 (f_1, f_{18}, f_5, and f_4)	1400
	15	Hybrid Function 6 (f_6, f_{18}, f_4, and f_{10})	1500
	16	Hybrid Function 7 (f_{16}, f_{13}, f_{19}, f_{10}, and f_5)	1600
	17	Hybrid Function 8 (f_1, f_{13}, f_5, f_{18}, and f_{12})	1700
	18	Hybrid Function 9 (f_1, f_5, f_{19}, f_{14}, and f_6)	1800
	19	Hybrid Function 10 (f_{17}, f_{16}, f_{13}, f_5, f_{10}, and f_{20})	1900
Composition Functions	20	Composition Function 1 (f_4, f_{11}, and f_4)	2000
	21	Composition Function 2 (f_5, f_{15}, and f_{10})	2100
	22	Composition Function 3 (f_4, f_{13}, f_{13}, and f_5)	2200
	23	Composition Function 4 (f_{13}, f_{11}, f_{15}, and f_5)	2300
	24	Composition Function 5 (f_5, f_{17}, f_{13}, f_{12}, and f_4)	2400
	25	Composition Function 6 (f_6, f_{10}, f_{15}, f_4, and f_5)	2500
	26	Composition Function 7 (f_{18}, f_5, f_{10}, f_{11}, f_{10}, and f_4)	2600
	27	Composition Function 8 (f_{13}, f_{15}, f_{12}, f_4, f_{17}, and f_6)	2700
	28	Composition Function 9 (HyF 5, HyF 6 and HyF 7)	2800
	29	Composition Function 10 (HyF 5, HyF 8 and HyF 9)	2900
Search Range: [-100,100]D			

All test functions are minimization problems defined as follows:

$$Minf(x), x = [x_1, x_2, ..., x_D]^T$$

D: Dimensions 30, 50, and 100.

Search range: $[-100, 100]^D$

M_i: rotation matrix. Different rotation matrices are assigned to each function and each basic function.

MaxFES: $10000 \times D$. For simplicity, the D is always considered 30 and MaxFES=300,000

Definitions of the Basic Functions:

1. Bent Cigar Function

$$f_1(\boldsymbol{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \tag{B.1}$$

2. Zakharov Function

$$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4 \tag{B.2}$$

3. Rosenbrock's Function

$$f_4(\boldsymbol{x}) = \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right) \tag{B.3}$$

4. Rastrigin's Function

$$f_5(\boldsymbol{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \tag{B.4}$$

5. Expanded Schaffer's F6 Function

   Schaffer's F6 Function: $g(x,y) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x^2+y^2}\right)-0.5\right)}{(1+0.001(x^2+y^2))^2}$

$$f_6(\boldsymbol{x}) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_{D-1}, x_D) + g(x_D, x_1) \tag{B.5}$$

6. Lunacek bi-Rastrigin Function

$$f_8(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right) + f_{13}*$$

$$f = \mathbf{M}_1 \frac{5.12(x-o)}{100}, y_i = \begin{cases} x_i & \text{if } |x_i| \leq 0.5 \\ \text{round}(2x_i)/2 & \text{if } |x_i| > 0.5 \end{cases} \quad \text{for } i = 1, 2, \ldots, D$$

$$z = \mathbf{M}_1 \Lambda^{10} \mathbf{M}_2 T_{ayy}^{0.2}\left(T_{asz}(y)\right)$$

Where Λ^{α} : a diagonal matrix in D dimensions with the i^{th} diagonal element as $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}$ $i = 1, 2, \ldots, D$.

$$T_{asy}^{\beta}: \text{ if } x_i > 0, x_i = x_i^{1+\beta \frac{i-1}{D-1}\sqrt{x_i}}, \text{ for } i = 1, \ldots, D$$

T_{asz} : for $x_i = \text{sign}(x_i)\exp\left(\hat{x}_i + 0.049\left(\sin(c_1\hat{x}_i) + \sin(c_2\hat{x}_i)\right)\right)$, for $i = 1$ and $D^{[4]}$

$$\text{where } \hat{x}_i = \begin{cases} \log(|x_i|) & \text{if } x_i \neq 0 \\ 0 & \text{otherwise} \end{cases}, \text{sign}(x_i) = \begin{cases} -1 & \text{if } x_i < 0 \\ 0 & \text{if } x_i = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$c_1 = \begin{cases} 10 & \text{if } x_i > 0 \\ 5.5 & \text{otherwise} \end{cases}, \text{ and } c_2 = \begin{cases} 7.9 & \text{if } x_i > 0 \\ 3.1 & \text{otherwise} \end{cases}$$

7. Non-continuous Rotated Rastrigin's Function

$$f_9(\boldsymbol{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1} (w_i - 1)^2 \left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_D - 1)^2 \left[1 + \sin^2(2\pi w_D)\right] \tag{B.6}$$

Where $w_i = 1 + \frac{x_i - 1}{4}, \forall i = 1, \ldots, D$

8. Levy Function

$$f_{10}(\boldsymbol{x}) = 418.9829 \times D - \sum_{i=1}^{D} g\left(z_i\right), \quad z_i = x_i + 4.209687462275036\mathrm{e}+002$$

$$g\left(z_i\right) = \begin{cases} z_i \sin\left(\left|z_i\right|^{1/2}\right) & \text{if } \left|z_i\right| \leq 500 \\ \left(500 - \bmod\left(z_i, 500\right)\right)\sin\left(\sqrt{\left|500 - \bmod\left(z_i, 500\right)\right|}\right) - \frac{\left(z_i - 500\right)^2}{10000D} & \text{if } z_i > 500 \\ \left(\bmod\left(\left|z_i\right|, 500\right) - 500\right)\sin\left(\sqrt{\left|\bmod\left(\left|z_i\right|, 500\right) - 500\right|}\right) - \frac{\left(z_i + 500\right)^2}{10000D} & \text{if } z_i < -500 \end{cases} \tag{B.7}$$

9. Modified Schwefel's Function

$$f_{11}(\boldsymbol{x}) = \sum_{i=1}^{D} \left(10^6\right)^{\frac{i-1}{D-1}} \boldsymbol{x}_i^2 \tag{B.8}$$

10. High Conditioned Elliptic Function

$$f_{12}(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2 \tag{B.9}$$

11. Discus Function

$$f_{13}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right) + 20 + \mathrm{e} \tag{B.10}$$

12. Ackley's Function

$$f_1(\boldsymbol{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \tag{B.11}$$

13. Weierstrass Function

$$f_{14}(\boldsymbol{x}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k \left(x_i + 0.5\right)\right)\right] \right) - D \sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k \cdot 0.5\right)\right]$$
$$a = 0.5, b = 3, k\max = 20 \tag{B.12}$$

14. Griewank's Function

$$f_{15}(\boldsymbol{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{B.13}$$

15. Katsuura Function

$$f_{16}(\boldsymbol{x}) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{32} \frac{\left|2^j x_i - \text{round}\left(2^j x_i\right)\right|}{2^j}\right)^{\frac{10}{D^{12}}} - \frac{10}{D^2} \tag{B.14}$$

16. HappyCat Function

$$f_{17}(\boldsymbol{x}) = \left|\sum_{i=1}^{D} x_i^2 - D\right|^{1/4} + \left(0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right) / D + 0.5 \tag{B.15}$$

17. HGBat Function

$$f_{18}(\boldsymbol{x}) = \left|\left(\sum_{i=1}^{D} x_i^2\right)^2 - \left(\sum_{i=1}^{D} x_i\right)^2\right|^{1/2} + \left(0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right) / D + 0.5 \tag{B.16}$$

18. Expanded Griewank's plus Rosenbrock's Function

$$f_{19}(\boldsymbol{x}) = f_7\left(f_4\left(x_1, x_2\right)\right) + f_7\left(f_4\left(x_2, x_3\right)\right) + \ldots + f_7\left(f_4\left(x_{D-1}, x_D\right)\right) + f_7\left(f_4\left(x_D, x_1\right)\right) \tag{B.17}$$

19. Schaffer's F7 Function

$$f_{20}(\boldsymbol{x}) = \left[\frac{1}{D-1}\sum_{i=1}^{D-1}\left(\sqrt{s_i}\cdot\left(\sin\left(50.0s_i^{0.2}\right)+1\right)\right)\right]^2, s_i = \sqrt{x_i^2 + x_{i+1}^2} \tag{B.18}$$

Appendix C

CEC-2017 Many-Objective Optimization Benchmark Functions

Table C.1: The CEC-2017 MAOPs (MaF) benchmark function suite properties.

Problem	**Definitions**	**Properties**
MaF1	Modified inverted DTLZ1	Linear
MaF2	DTLZ2BZ	Concave
MaF3	Convex DTLZ3	Convex, multimodal
MaF4	Inverted badly-scaled DTLZ3	Concave, multimodal
MaF5	Convex badly-scaled DTLZ4	Convex, biased
MaF6	DTLZ5(I,M)	Concave, degenerate
MaF7	DTLZ7	Mixed, disconnected, multimodal
MaF8	Multi-Point Distance Minimization Problem	Linear, degenerate
MaF9	Multi-Line Distance Minimization Problem	Linear, degenerate
MaF10	WFG1	Mixed, biased
MaF11	WFG2	Convex, disconnected, nonseparable
MaF12	WFG9	Concave, nonseparable, biased, deceptive
MaF13	PF7	Concave, unimodal, nonseparable, degenerate
MaF14	LSMOP3	Linear, partially separable, large scale
MaF15	Inverted LSMOP8	Convex, partially separable, large scale

Maximum Number of Fitness Evaluations (FEs): $\max\{100000, 10000 \times D\}$

Number of Independent Runs: 31