Multi-Guide Particle Swarm Optimization for Large-Scale Multi-Objective Optimization Problems

Amirali Madani

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Department of Computer Science
Brock University
St. Catharines, Ontario

Abstract

Multi-guide particle swarm optimization (MGPSO) is a novel metaheuristic for multi-objective optimization based on particle swarm optimization (PSO). MGPSO has been shown to be competitive when compared with other state-of-the-art multi-objective optimization algorithms for low-dimensional problems. However, to the best of the author's knowledge, the suitability of MGPSO for high-dimensional multi-objective optimization problems has not been studied. One goal of this thesis is to provide a scalability study of MGPSO in order to evaluate its efficacy for high-dimensional multi-objective optimization problems. It is observed that while MGPSO has comparable performance to state-of-the-art multi-objective optimization algorithms, it experiences a performance drop with the increase in the problem dimensionality. Therefore, a main contribution of this work is a new scalable MGPSO-based algorithm, termed cooperative co-evolutionary multi-guide particle swarm optimization (CCMGPSO), that incorporates ideas from cooperative PSOs. A detailed empirical study on well-known benchmark problems comparing the proposed improved approach with various state-of-the-art multi-objective optimization algorithms is done. Results show that the proposed CCMGPSO is highly competitive for high-dimensional problems.

Acknowledgements

I would like to extend my gratitude to the following people:

- Professor Beatrice Ombuki-Berman for her excellent guidance during the process of this work.
- Professor Andries Engelbrecht for his collaboration and valuable comments.
- My mom, my dad, and my sister for always being supportive of me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many real-life problems involve two or more (usually conflicting) objectives. Such problems, referred to as multi-objective optimization problems, are observed in many real-life instances, such as space missions [97] or cloud computing [52]. In these problems, the existing objectives are often conflicting, meaning that optimizing one objective would typically result in a less optimal value for at least one of the others. For example, in [97], the cost and the time of a space mission were modeled as a multi-objective optimization problem, such that a mission with low cost would take longer to complete and vice versa.

Population-based multi-objective evolutionary algorithms (MOEAs) have gained a lot of popularity for solving multi-objective optimization problems. There is a plethora of different MOEAs available for multi-objective optimization [18, 54, 69, 87, 94, 95]. Needless to say, different MOEAs use different approaches for solving multi-objective optimization problems. Some algorithms such as the non-dominated sorting genetic algorithm (NSGA-II) [18] are based on Pareto-dominance[1], or some algorithms such as MOEA/D [87] use an aggregated value of all objectives as the fitness measure. Although there is no formal definition, some papers have defined large-scale multi-objective problems as problems that have more than 100 decision variables [75] [47]. Unfortunately, with the increase in the number of decision variables, most MOEAs lose performance. In fact, some papers have empirically shown that the majority of existing MOEAs cannot solve large-scale multi-objective optimization problems efficiently [47] [75] [88]. As the number of decision variables in multi-objective optimization problems increases, MOEAs are less likely to explore the larger search spaces efficiently due to either premature convergence or converging to a region that is too large to explore [55].

[1]The concept of Pareto-dominance is discussed in Section 2.3.1

It must be added that the aforementioned performance deterioration over large-scale problems with many decision variables is not exclusive to multi-objective optimization, or any specific algorithm. In fact, Van den Bergh and Engelbrecht [77] described this phenomenon as the "*curse of dimensionality*" [77], from which most stochastic optimization algorithms (regardless of the number of objectives) suffer. As the search space gets bigger, a bigger volume is added on top of an already complex optimization problem. When solving these problems, optimization algorithms update their guides by which they control their individuals based on fitness improvements of some kind. However, due to a phenomenon known as "*two steps forward, one step back*" [77], this update in variable vectors could put some dimensions in the correct direction, while moving the other decision variables in a less optimal one (two steps forward, one step back). Therefore, the performance of optimization algorithms is gradually lost as the number of decision variables increases.

In order to tackle large-scale problems, different methods have been proposed over the years. The idea of cooperative coevolution (CC) for large-scale single-objective optimization was first introduced by Potter and De Jong [60] in the form of a new algorithm termed the cooperative coevolutionary genetic algorithm (CCGA). This algorithm divides the decision variables into smaller subgroups where each subgroup is optimized by an independent genetic algorithm (GA) [82]. Inspired by the CC framework, Van den Bergh and Engelbrecht [77] proposed two major cooperative approaches based on particle swarm optimization (PSO) [37], termed cooperative split particle swarm optimization (CPSO-S_k) and cooperative hybrid particle swarm optimization (CPSO-H_k). In CPSO-S_k, the decision variables are classified into k groups, where each group is optimized using PSO and a context vector is used to construct full-dimensional solutions out of the smaller groups. In CPSO-H_k, in addition to the aforementioned k groups of decision variables, a full-dimensional swarm is also used, such that the full-dimensional swarm and the low-dimensional subswarms cooperate with each other by exchanging solutions.

Looking to add some dynamic elements into the CPSO-S_k, the the cooperative coevolutionary particle swarm optimization (CCPSO) algorithm [43] introduced the frequent regrouping of the decision variables throughout the search, as opposed to CPSO-S_k where the variables are randomly grouped only once at the beginning. The main motivation behind CCPSO was to constantly regroup the decision variables, in hopes of optimizing the interacting ones together as a group at some point during the search. Looking to improve upon CCPSO's success, CCPSO2 [44] was proposed. In addition to the previously discussed constant regrouping, CCPSO2 also continuously

changes the size of each decision variable group.

The CC framework, in spite of its success in solving large-scale problems, can be very expensive in terms of the computational cost (the number of function evaluations). More recently, contribution based-approaches have gained popularity [58] [56] [83] [84]. The focal idea of these approaches is to save computational budget by allocating more function evaluations to variable groups that have contributed more to the search, rather than equally distributing this budget among all groups as in regular CC-based approaches. Experimental results [58] have shown that using contribution data (both historical and dynamic) can save computational budget especially on imbalanced problems where different dimensions affect the objective value in unequal amounts.

Moreover, many different approaches have been proposed in hopes of improving the performance of multi-objective evolutionary algorithms (MOEAs) on large-scale problems. Similar to a lot of the previously discussed algorithms, some methods are based on decomposition and classify the decision variables into different groups [47] [88] [6]. For example, the work in [88] uses a k-means method for classifying different variables and optimizing them separately. On the other hand, some approaches do not rely on decomposition and optimize the large-scale variables of a problem as they are. For example, the large-scale multi-objective optimization algorithm based on the competitive swarm optimizer (LMOCSO) was proposed by Tian *et al.* [75]. This algorithm is heavily inspired by the competitive swarm optimizer (CSO) [7], where competitions are used to improve the overall diversity by pairing different solutions together and making the worse solution learn from the better one. Some non-decomposition approaches for large-scale multi-objective problems (LSMOPs) propose novel offspring generation methods which can be incorporated into all applicable MOEAs that produce new offspring (such as the genetic algorithm) instead of constantly updating a single population (such as the particle swarm optimization). For example, He *et al.* [24] proposed an adaptive offspring generation framework, termed DGEA. DGEA uses two kinds of direction vectors[2] for offspring generation, one for convergence and the other for diversity improvement. Experimental results in [24] showed competitive performance of the DGEA framework on large-scale problems, when compared with five state-of-the-art approaches.

There has also been a growing interest in large-scale multi-objective optimization methods based on problem reformulation in recent years. Such approaches typically involve deriving a lower-dimensional problem out of a large-scale one, such that solv-

[2]A direction vector is used for guiding new solutions

ing the former would result in optimizing the latter. For example, Zille *et al.* [91] proposed the weighted optimization framework (WOF). WOF uses a transformation function for converting an arbitrary LSMOP into a low-dimensional problem, and then optimizes both the original problem and the transformed one for predefined intervals. As another example, He *et al.* [25] proposed the large-scale multi-objective optimization framework based on problem reformulation, termed LSMOF. LSMOF locates some reference directions that are controlled by a set of weight variables. This results in a reformulated single-objective[3] problem with fewer decision variables. This is done in hopes of eventually locating the global best by optimizing the weight variables associated with different reference directions.

As discussed before, CC-based approaches can be quite costly, especially if the problems are classified into many small subgroups. As a result, some researchers have proposed novel ways of utilizing the CC framework for solving LSMOPs. For example, Antonio *et al.* [2], proposed the operational decomposition (OD) framework which aims to improve the crossover operations in MOEAs that have them. In OD, decision variables are still divided into smaller subgroups, but these groups are only used during the crossover phase. Instead of using two full-dimensional solutions as parents in a crossover operation, OD uses low-dimensional parts of large-scale solutions, utilizing the CC framework with no additional function evaluations.

Multi-guide particle swarm optimization (MGPSO) [64] [63] is a novel metaheuristic that adapts the particle swarm optimization [37] for multi-objective optimization. In MGPSO, each objective is optimized using a single-objective PSO and these subswarms interact with each other using an external archive of trade-off solutions. In its original paper, MGPSO showed competitive performance on the WFG [30] and ZDT [92] test suites when put up against other MOEAs such as NSGA-II [71], SMPSO [54], and OMOPSO [69]. However, there experiments were all conducted on low-dimensional problems. Therefore, the performance of MGPSO on large-scale problems has to be further investigated. Recently, Steenkamp and Engelbrecht [70] studied the scalability of MGPSO to many objectives and observed a competitive performance from MGPSO when compared with other state-of-the-art approaches for many-objective optimization. On the other hand, and to the best of the author's knowledge, MGPSO's scalability to many decision variables remains unexplored. Due to this lack of study with reference to MGPSO and many decision variables, MGPSO's potential weaknesses in solving large-scale problems have also not been addressed

[3]LSMOF also reduces the number of objectives, as well as the number of dimensions by assigning a fitness measure to a set of multi-objective solutions.

properly. With this in mind, the contributions of this thesis are presented in the following section.

1.1 Main Contributions

The main contributions of this thesis are as follows:

- Perform a literature review of the previously proposed approaches for large-scale single objective optimization, PSO-based approaches for multi-objective optimization, evolutionary approaches for large-scale multi-objective optimization, and finally other relevant recent advances in evolutionary algorithms for single- and multi-objective optimization.
- Perform a scalability study of MGPSO and four other state-of-the-art PSO-based approaches for multi-objective optimization on the WFG test suite for 24, 50, 100, 500, and 1000 decision variables to detect the potential weaknesses of MGPSO in solving large-scale problems.
- Propose a new MGPSO-based algorithm, incorporating cooperative strategies, for large-scale multi-objective optimization termed cooperative co-evolutionary multi-guide particle swarm optimization (CCMGPSO).
- Perform a detailed empirical study on well-known benchmark problems comparing the proposed improved MGPSO approach with various state-of-the-art multi-objective optimization algorithms to determine the competitiveness of the proposed CCMGPSO.

1.2 Thesis Structure

The structure of this thesis is outlined below:

- **Chapter 2** covers background information on the basics of particle swarm optimization (PSO), PSO-based approaches for large-scale single-objective optimization, the basics of multi-objective optimization, benchmark suites for multi-objective optimization, large-scale multi-objective optimization and some evolutionary algorithms proposed for large-scale multi-objective problems.

- **Chapter 3** is dedicated to a scalability study of PSO-based approaches for large-scale multi-objective optimization as a main contribution of this thesis. More specifically, a scalability study involving MGPSO and four more PSO-based approaches, namely optimized multi-objective particle swarm optimization (OMOPSO) [69], speed-constrained multi-objective particle swarm optimization (SMPSO) [54], multi-objective particle swarm optimization with multiple search strategies (MMOPSO) [45], and competitive mechanism-based multi-objective particle swarm optimization (CMOPSO) [89] is conducted on the Walking Fish Group (WFG) [30] [31] test suite for 24, 50, 100, 500, and 1000 decision variables for two to three objectives to see how well each one of the algorithms scales as the number of decision variables is increased.

- **Chapter 4** proposes a new MGPSO-based algorithm for large-scale multi-objective optimization, termed cooperative co-evolutionary multi-guide particle swarm optimization (CCMGPSO) and inspired by previous algorithms for large-scale single- and multi-objective optimization, to address the shortcomings of MGPSO as detected in the scalability study of Chapter 3.

- **Chapter 5** is dedicated to a comparative empirical study involving CCMGPSO, and six state-of-the art algorithms including the best-performing algorithm as detected in the scalability study given in Chapter 3. The results indicate that CCMGPSO is highly competitive.

- **Chapter 6** includes the concluding remarks of this thesis and some potential avenues of future work.

Chapter 2

Background

This section covers the necessary background information for this thesis.

2.1 Particle Swarm optimization

Particle swarm optimization, introduced by Kennedy and Eberhart in 1995 [37], is a stochastic population-based single-objective optimization algorithm that aims to simulate the social behavior of birds in a flock. Let n_s and n_x be the swarm size and the number of dimensions respectively. In a run of PSO, n_s different particles are first initialized inside an n_x-dimensional search space representing the objective that is being optimized. Each particle contains a memory of a *personal best* position (found by itself) and a *neighborhood best* position (found by the particle's neighborhood according to predefined structure of neighbors) and during each iteration updates its velocity and position. Later in 1998, Shi and Eberhart [67], in order to enhance the trade-off between PSO's exploration and exploitation, incorporated the inertia weight ω into the initial velocity update formula which resulted in the following:

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 \mathbf{r}_{1i}(t)(\mathbf{y}_i(t) - \mathbf{x}_i(t)) + c_2 \mathbf{r}_{2i}(t)(\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t)) \tag{2.1}$$

After updating the particle's velocity, the position is updated using:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2.2}$$

where t is the time step (the number of iteration), i is the index of the particle, ω is the inertia weight, $\mathbf{r}_1$ and $\mathbf{r}_2$ are random vectors sampled from $[0,1]^{n_x}$, c_1 and c_2 are cognitive and social acceleration coefficients respectively. The position, velocity,

personal best position, and neighborhood best position vectors of particle i are indicated by $\mathbf{x}_i$, $\mathbf{v}_i$, $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$ respectively. Moreover, the amount of influence that the current velocity, the personal best position and the global best position have on the next velocity is controlled by ω, c_1, and c_2 respectively.

2.2 Particle Swarm Optimization for Large-Scale Single-Objective Optimization

This section covers some PSO-based approaches for large-scale single-objective optimization since some of these algorithms inspired the proposed algorithm in this thesis (fully discussed in Chapter 4).

2.2.1 Cooperative Coevolution

The idea of cooperative coevolutionary evolutionary algorithms (CCEAs) was first proposed by Potter and De Jong [60] in the form of a new algorithm termed cooperative coevolutionary genetic algorithm (CCGA). In their algorithm, Potter and De Jong divided an n_x-dimensional problem to n_x one-dimensional problems, where each problem was optimized by a GA subpopulation. The fitness of each one-dimensional individual was calculated by forming an n_x-dimensional vector using its value and other values from other subpopulations.

2.2.2 Cooperative Particle Swarm Optimization

Proposed by Engelbrecht and Van den Bergh [77], cooperative particle swarm optimization (CPSO) aims to address the shortcomings of the stochastic optimization algorithms (such as PSO) in solving large-scale problems. Also referred to as the "*curse of dimensionality*", this phenomenon is a limitation of these algorithms in large-scale environments. This means that, since in these algorithms decision vectors are evaluated once all decision variables are updated, the optimizer can be deceived by an objective value obtained by improvements in two decision variables but a worse value for another decision variable ("*two steps forward, one step back*"). In order to address this issue, CPSO proposes a decomposition framework to optimize different decision variables separately. CPSO splits any given n_x-dimensional problem to n_x one-dimensional problems. Also referred to as split-CPSO (CPSO-S), this algorithm uses a context vector for evaluating the one-dimensional solutions. A context vector

Algorithm 1 CPSO-S [77]

procedure CPSO-S
  Initialize n_x one-dimensional swarms
  Randomly initialize an n_x-dimensional context vector
  Define $\mathbf{b}(j, z) = (P_1.\hat{\mathbf{y}}, P_2.\hat{\mathbf{y}}, \ldots, P_{j-1}.\hat{\mathbf{y}}, z, P_{j+1}.\hat{\mathbf{y}}, \ldots, P_{n_x}.\hat{\mathbf{y}})$
  for each sub-swarm $j = 1, \ldots n_x$ **do**
    for each particle $i = 1, \ldots, n_s$ **do**
      if $f(\mathbf{b}(j, P_j.x_i) < f(\mathbf{b}(j, P_j.\mathbf{y}_i)$ **then**
        $P_j.\mathbf{y}_i \leftarrow P_j.x_i$
      end if
      if $f(\mathbf{b}(j, P_j.\mathbf{y}_i) < f(\mathbf{b}(j, P_j.\hat{\mathbf{y}})$ **then**
        $P_j.\hat{\mathbf{y}} \leftarrow P_j.\mathbf{y}_i$
      end if
    end for
    for each particle $i = 1, \ldots, n_s$ **do**
      Update particle's velocity using Eq. (2.1)
      Update particle's position using Eq. (2.2)
    end for
  end for
end procedure

is an n_x-dimensional vector that is initialized randomly and later used to combine the best positions found by different subswarms. The context vector holds the best values found for each dimension by its corresponding subswarm, and is used to evaluate low-dimensional solutions by putting them in their corresponding dimensions of the context vector. The context vector is then updated if this results in an improved fitness value.

The pseudo-code of CPSO-S is presented in Algorithm 1. $\mathbf{b}(j, z)$ is a function for evaluating the low-dimensional solutions. This function builds an n_x-dimensional vector using the global best vectors of all subswarms except the j-th one. For the j-th subswarm, the vector z (a prospective low-dimensional position vector) is used.

2.2.3 CPSO-S$_k$ and CPSO-H$_k$

Within the same paper [77], Engelbrecht and Van den Bergh also proposed CPSO-S$_k$ and CPSO-H$_k$. These algorithms are extensions to CPSO-S, CPSO-S$_k$ is CPSO-S with k groups of decision variables. Therefore, CPSO-S$_k$ has k subswarms, each having $\frac{n_x}{k}$ decision variables, compared to CPSO's n_x and one respectively. CPSO-H$_k$ is an extension to CPSO-S$_k$, where an n_x-dimensional swarm is also added to the algorithm. The n_x-dimensional and $\frac{n_x}{k}$-dimensional swarms cooperate with each

Algorithm 2 CCPSO [43]

```
1: procedure CCPSO
2:     Initialize k n_x/k one-dimensional swarms;
3:     Randomly initialize an n_x-dimensional context vector
4:     while the stopping criterion is not met do
5:         for each sub-swarm j = 1, ... n_x do
6:             for each particle i = 1, ..., n_s do
7:                 if f(b(j, P_j.x_i) < f(b(j, P_j.y_i) then
8:                     P_j.y_i ← P_j.x_i
9:                 end if
10:                if f(b(j, P_j.y_i) < f(b(j, P_j.ŷ) then
11:                    P_j.ŷ ← P_j.y_i
12:                end if
13:            end for
14:            for each particle i = 1, ..., n_s do
15:                Update particle's velocity using Eq. (2.1)
16:                Update particle's position using Eq. (2.2)
17:            end for
18:        end for
19:        Initialize a k-dimensional PSO for weight optimization;
20:        Optimize ŷ for a few iterations using the weight population, update ŷ if
   possible;
21:    end while
22: end procedure
```

other in different ways; for example, before each iteration, one of the n_x-dimensional particles is replaced by the context vector.

2.2.4 Cooperative Coevolving Particle Swarm Optimization

Proposed by Li and Yao [43], cooperative coevolving particle swarm optimization (CCPSO) aims to improve CPSO-S$_k$'s performance by constantly regrouping the decision variables, in hopes of optimizing interacting variables (if any) together as a group. In CPSO-S, CPSO-S$_k$, and CPSO-H$_k$, the random grouping is employed once at the very beginning, and the resulting groups are kept the same for the entirety of the optimization process. However, CCPSO constantly regroups the decision variables before each iteration. Due to its previous success [85] [86], CCPSO also pairs random grouping with adaptive weighting.

The pseudo-code of CCPSO is presented in Algorithm 2. Every CCPSO iteration starts with a random permutation of the decision variables followed by a run of the CPSO-S$_k$. After this, a dynamic weighting mechanism is employed. In this scheme, a

k-dimensional population is initialized as a set of weight vectors to further optimize the context vector. This is done by multiplying each decision variable in each of the k groups by a weight variable that represents the group. The weight values are chosen such that, if multiplied by them, none of the decision variables of a group will exceed their lower or upper bounds. This scheme is applied in hopes of further improving the best found solution.

2.2.5 Cooperative Coevolving Particle Swarm Optimization 2

In hopes of improving CCPSO's performance, Li and Yao [44] proposed CCPSO2. CCPSO2 no longer uses the adaptive weighting scheme as in CCPSO. Instead, it uses a different decision variable grouping approach in the form of dynamic values for k. Unlike CCPSO's random regrouping before every iteration, CCPSO2 only regroups the decision variables if the fitness value of the context vector has not improved. For each random regrouping, the value s is randomly selected from the set $S = \{2, 5, 50, 100, 200\}$. Until the next regrouping, each group has s decision variables assigned to it, forming $k = \frac{n_x}{s}$ groups of decision variables in total. Instead of the traditional velocity-based position update mechanism, CCPSO2 updates the position using the following equation:

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{y}_i(t) + \mathcal{C}(1)|(\mathbf{y}_i(t) - \hat{\mathbf{y}}_i(t)| & if \;\; r < p \\ \hat{\mathbf{y}}_i(t) + \mathcal{N}(0,1)|(\mathbf{y}_i(t) - \hat{\mathbf{y}}_i(t)| & otherwise \end{cases} \tag{2.3}$$

where r is a random number in $[0, 1]$, with $\mathcal{C}(1)$ and $\mathcal{N}(0, 1)$ being two numbers sampled by Cauchy and Gaussian distributions respectively. Particle i's neighbourhood best is denoted by $\hat{\mathbf{y}}_i$ and is determined based on the fitness values of the particle itself, and its immediate left and right neighbours in a ring topology.

2.2.6 Adaptive Multi-Context Cooperative Coevolving Particle Swarm Optimization

Using CCPSO2 as a starting point, Tang et al. [72] proposed the adaptive multi-context cooperative coevolving PSO (AM-CCPSO). AM-CCPSO aims to improve CCPSO2's performance on non-separable and multi-modal problems. Unlike CPSO-S, CPSO-S$_k$, CPSO-H$_k$, CCPSO, and CCPSO2 which use a single context vector, AM-CCPSO uses a pool of context vectors. The motivation behind using multiple context

vectors is to evolve a set of context vectors over the duration of the optimization process. AM-CCPSO uses a roulette selection scheme to choose a context vector from this set based on their respective fitness values.

2.3 Multi-Objective Optimization

A vast majority of optimization problems consist of more than one objective [13] [50]. These objectives are usually conflicting with each other. In other words, multi-objective optimization problems (MOOPs) involve finding a set of optimal trade-offs between two or three problems or objectives [73]. If an optimization problem has more than three objectives, it is referred to as a many-objective optimization problem. The goals of a multi-objective optimization algorithm were defined by Zitzler [93] as:

1. finding solutions which are close to the true solutions,
2. finding solutions that are evenly spread out, and
3. maximizing the extent of the found solutions.

Generally speaking, MOOPs that have more than 100 decision variables are referred to as large-scale MOOPs [47] [75].

Multi-objective optimization problems are often encountered in real life. As an example, the work in [97] proposed an approach to optimize the (conflicting) objectives space mission cost and time as a multi-objective optimization problem. Midya *et al.* [52] proposed the hybrid adaptive particle swarm optimization (HAPSO) algorithm for solving a multi-objective problem related to task scheduling in the field of cloud computing. In the field of finance, multi-objective optimization has been used to optimize the accuracy and the length of market predictions [28, 62, 73]. Multi-objective optimization has been also been applied to the field of mechanics to minimize the equipment cost and the energy consumption [16, 36, 66].

2.3.1 Formal Definitions

More formally and in the context of a minimization problem, a MOOP is formulated as:

$$minf(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{n_m}(\mathbf{x})) \tag{2.4}$$

where n_m is the number of objectives and $\mathbf{x}$ is an n_x-dimensional vector called the **decision vector**, such that:

$$\mathbf{x} = (x_1, x_2, \ldots, x_{n_x}) \tag{2.5}$$

$f(\mathbf{x})$ is also referred to as the **objective vector**.

Definition 1. ***Pareto-dominance:*** *Decision vector* $\mathbf{x_1}$ *is said to dominate decision vector* $\mathbf{x_2}$ *(denoted by* $\mathbf{x_1} \prec \mathbf{x_2}$*) if and only if* $f_m(\mathbf{x_1}) \leq f_m(\mathbf{x_2})\ \forall m \in [1, n_m]$ *and* $\exists m \in [1, n_m]$ *such that* $f_m(\mathbf{x_1}) < f_m(\mathbf{x_2})$ *[64].*

Definition 2. ***Pareto-optimal:*** *In a given set* S *of decision vectors, decision vector* $\mathbf{x_1}$ *is said to be Pareto-optimal if there exists no* $\mathbf{x_2} \in S$ *such that* $\mathbf{x_2} \prec \mathbf{x_1}$ *[64].*

Definition 3. ***Pareto-optimal set:*** *A given set* S *of decision vectors is called a Pareto-optimal set if it only contains Pareto-optimal decision vectors [64].*

Definition 4. ***Pareto-optimal front:*** *A given set* F *of objective vectors is called a Pareto-optimal front (POF) if it only contains the corresponding objective vectors of a Pareto-optimal set [64].*

2.4 Multi-Objective Benchmark Suites

In order to evaluate an optimization algorithm and find out about its strengths and weaknesses, there exists a need for benchmark suites. Deb [14] noted that, for creating problems that are used to evaluate the performance of an algorithm, one needs to take certain characteristics into account. For example, these problems should make convergence difficult by adding deception and multi-modality, and they should make diversity challenging by adding convexity or non-convexity into the Pareto-optimal front. Three well-known benchmark suites were used in this thesis, namely the Zitzler-Deb-Thiele (ZDT) test suite [92], the Deb-Thiele-Laumanns-Zitzler (DTLZ) [19] test suite, and the Walking Fish Group (WFG) test suite [30] [31]. These test suites are all discussed in more detail in Appendix A.

2.5 Multi-Objective Optimization Performance Measures

In order to compare different multi-objective optimization algorithms, the existence of performance metrics is necessary. This section covers some of the commonly used

performance measures in the field of multi-objective optimization.

2.5.1 Spread

First introduced in [18], the spread measure quantifies the distances between the solutions in the POF. For a given set of objective vectors, spread (S) can be formulated as:

$$\Delta = \frac{d_l + d_f + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_l + d_f + (N-1)\bar{d}} \tag{2.6}$$

where

$$\bar{d} = \frac{1}{N-1} \sum_{k=1}^{N-1} d_k \tag{2.7}$$

where N is the number of solutions in the obtained front. To calculate the spread measure, first the true POF and the obtained POF (obtained by the multi-objective optimization algorithm) are sorted lexicographically. When sorted lexicographically, the first and the last solutions in the obtained POF are called the **extreme** solutions. Moreover, d_l and d_f are the distances between the extreme solutions in the true POF and their counterparts in the obtained POF. $\bar{d}$ is the average distance between each of two solutions in the sorted obtained POF (there are $N-1$ consecutive distances) and d_i is the distance between solutions i and $i+1$. Ideally and if the obtained POF contains the extreme solutions of the true POF and the solutions are equally spaced, $\Delta = 0$, otherwise $\Delta > 0$.

2.5.2 Inverted Generational Distance

Inverted Generational Distance (IGD) was introduced by Coello Coello and Reyes-Sierra [12] [68] and is defined as the following:

$$IGD = \frac{\sqrt{\sum_{k=1}^{|Q_{true}|} d_k^2}}{|Q_{true}|} \tag{2.8}$$

where $|Q_{true}|$ is the number of solutions inside the true POF and d_k is the distance between the kth solution in the true POF and its closest counterpart in the obtained POF. Smaller values for IGD indicate a closer POF to the true front and hence a better performance.

2.5.3 Hypervolume

The hypervolume measure (HV), introduced by Zitzler and Thiele [96], is used to measure the space covered by hypervolumes between each solution in the obtained POF and a predefined reference vector. The work in [96] uses the all-zero vector as the reference vector. In that case, if the problem is a minimization one, a smaller value for HV indicates a closer solution to the all-zero vector and hence a better performance. However, recent studies all use the nadir vector (consisting of the worst objectives in the Pareto-optimal front for each of the n_m objectives [64]) as the reference point. HV can potentially be deceiving when the Pareto-optimal front is not convex [78]. Additionally, picking an appropriate reference point for HV is not always easy. It has been shown that the results of HV comparisons between different algorithms depend on the location of the reference vector [35] [33]. Ishibuchi *et al.* [34] noted that some studies use a point slightly worse than the nadir vector so that the selected reference point dominates every point in the obtained front. This is particularly necessary when the optimization problem is more difficult to solve, and the obtained fronts are further away from the true fronts. As an example, Seada and Deb [65] used a point 1.01 times, Maltese *et al.* [49] used a point 1.1 times, and Wagner *et al.* [79] used a point 1.4 times worse than the nadir vector respectively. Ishibuchi *et al.* [34] also showed that when the problem has more than three objectives or the algorithm has a small population of individuals, a slightly worse point than the nadir vector may not always be suitable.

2.6 Particle Swarm Optimization for Multi-Objective Optimization

The original PSO algorithm was developed to solve single-objective optimization problems. However, a number of PSO variants have been developed to solve multi-objective optimization problems. This section presents a brief description of each of the five PSO-based multi-objective approaches studied in this paper.

2.6.1 Optimized Multi-Objective Particle Swarm Optimization

Coello and Sierra [69] proposed the optimized multi-objective particle swarm optimization (OMOPSO). OMOPSO uses a combination of Pareto-dominance and crowd-

ing distance [18] to update the external archive of non-dominated solutions. This algorithm splits the swarm into three parts, where one part is mutated by uniform mutation, the second part is mutated by non-uniform mutation, and the last part is not mutated at all. Uniform and non-uniform mutations are used to promote exploration and exploitation respectively. Unlike the majority of PSO algorithms, OMOPSO does not fix the values of ω, c_1 and c_2. Instead, ω is chosen randomly from $[0.1, 0.5]$, and c_1 and c_2 are randomly chosen from $[1.5, 2]$.

2.6.2 Speed-Constrained Multi-Objective Particle Swarm Optimization

Using OMOPSO as a stating point, Nebro *et al.* [54] proposed the speed-constrained multi-objective particle swarm optimization (SMPSO). SMPSO aims to solve the problem of particles' velocities becoming too high in multi-modal problems by adopting the constriction factor in [11] to limit the particles' velocities. SMPSO also differs from OMOPSO in the values for c_1 and c_2; in SMPSO these values are chosen randomly from $[1.5, 2.5]$.

2.6.3 Multi-Objective Particle Swarm Optimization with Multiple Search Strategies

Lin *et al.* [45] proposed the multi-objective particle swarm optimization with multiple search strategies (MMOPSO). MMOPSO uses the boundary intersection method [87] for the aggregation of all the objective values to select the personal best. MMOPSO has two search strategies, meaning that it uses the control parameter δ to choose between two velocity update equations, transforming (2.1) into:

$$\mathbf{v}_i(t+1) = \begin{cases} \omega\mathbf{v}_i(t) + c_1\mathbf{r}_{1i}(t)(\mathbf{y}_i(t) - \mathbf{x}_i(t)) & if \;\; r_3 < \delta \\ \omega\mathbf{v}_i(t) + c_2\mathbf{r}_{2i}(t)(\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t)) & otherwise \end{cases} \quad (2.9)$$

where δ is a parameter to control the exploration-exploitation trade-off, and r_3 is a number randomly chosen from $[0, 1]$. It is recommended to use $\delta \in [0.5, 0.9]$ to put more focus on exploitation [45]. Moreover, ω is chosen randomly from $[0.1, 0.5]$, with c_1 and c_2 being two numbers randomly chosen from $[1.5, 2]$. The global best position is chosen randomly from the archive.

2.6.4 Competitive Mechanism-Based Multi-Objective Particle Swarm Optimization

The competitive swarm optimizer was proposed by Cheng and Jin [7] for large-scale single-objective optimization. As its name suggests, CSO relies on competitions between different particles for optimization. In order to avoid premature convergence, CSO aims to improve the swarm diversity by randomly pairing two particles in a competition. In this competition, the particle with the worse fitness value is labeled as the loser and learns from the better-performing particle (the winner), which is directly transferred to the next generation's population. Looking to adapt the CSO concepts for multi-objective optimization, Zhang *et al.* [89] proposed the competitive mechanism-based multi-objective particle swarm optimization (CMOPSO). Unlike the majority of MOEAs, CMOPSO does not use any external archive and it relies solely on the local front.

The pseudo-code of CMOPSO is provided in Algorithm 3. In CMOPSO, first n_s particles are initialized as the $Swarm$. Then, during each iteration, the competitive process (Algorithm 4) is performed to create $Swarm^{'}$ which also has n_s particles. The competitive mechanism first selects γ (control parameter) elite particles by a combination of non-dominated sorting rankings and crowding distance [18] from the current swarm. Then, for each particle p_i, $i \in \{1, \ldots, n_s\}$ in $Swarm$, two particles are randomly selected from the elite set of size γ and the elite particle whose objective vector has a smaller angle with p_i's objective vector is the winner and p_i is the loser. For each winner-loser pair, the following equations are applied to create a new particle $p_i^{'}$:

$$\mathbf{v}_i^{'} = \mathbf{r}_{1i}\mathbf{v}_i + \mathbf{r}_{2i}(\mathbf{x}_w - \mathbf{x}_i) \tag{2.10}$$

$$\mathbf{x}_i^{'} = \mathbf{x}_i + \mathbf{v}_i^{'} \tag{2.11}$$

where $\mathbf{r}_1$ and $\mathbf{r}_2$ are two vectors randomly selected from $[0, 1]^{n_x}$, w is the index for the winner (the selected elite particle). The ordinary particles are indexed using $i, i^{'} \in \{1, \ldots, n_s\}$. Polynomial mutation is then applied to $\mathbf{x}_i^{'}$. Finally, the resulting $2n_s$ particles are truncated using the environmental selection from the strength Pareto evolutionary algorithm 2 (SPEA2) [95] to make up the n_s particles for the next iteration. This environmental selection fills up the next population by starting from the first non-dominated front and continuing to the others in a successive manner. When this approach reaches a non-dominated front that cannot be fit entirely inside

the population, it keeps removing overcrowded solutions from that front until the next population is full.

Algorithm 3 General framework of CMOPSO [89]

procedure CMOPSO
  Initialize N particles' position and velocity vectors as P and V respectively
  while stopping condition not satisfied **do**
    $P^{'} \leftarrow CompetitionBasedLearning(P, V)$
    $P \leftarrow EnvironmentalSelection(P, P^{'})$
  end while
end procedure

Algorithm 4 $CompetitionBasedLearning(P, V)$ [89]

procedure $CompetitionBasedLearning$
  Input: P (Current position vectors), V (current velocity vectors), L (elite particles set), γ (size of elite particle set)
  Output: $P^{'}$ (new position vectors)
  $L \leftarrow$ Select γ particles from P according to the front index and the crowding distance of each particle
  for each particle $p_i \in P$ **do**
    Choose particles a and b from P at random
    Between a and b, choose one particle whose objective vector has a smaller angle with that of p_i's as the winner (p_w)
    $v_i^{'} \leftarrow$ the updated velocity of p_i Eq. (2.10)
    $p_i^{'} \leftarrow$ the updated position of p_i Eq. (2.11)
    $P^{'} \leftarrow P^{'} \cup p_i^{'}$
  end for
  $P^{'} \leftarrow PolynomialMutation(P^{'})$
end procedure

2.7 Multi-Guide Particle Swarm Optimization

The multi-guide particle swarm optimization (MGPSO) [63, 64] is a new PSO-based approach for multi-objective optimization. As seen in Algorithm 5, in MGPSO each objective has its own separate subswarm and is optimized independently, meaning that the global best and personal best positions in each subswarm refer to the best values found for that specific objective. Because of this independence, a new guide, namely the archive guide, is introduced in MGPSO (hence the term multi-guide) and is used to establish communications between different subswarms through a bounded

archive. As a result, the velocity update equation in (2.1) is updated as:

$$\begin{aligned} \mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 \mathbf{r}_{1i}(t)(\mathbf{y}_i(t) - \mathbf{x}_i(t)) + \\ \lambda_i c_2 \mathbf{r}_{2i}(t)(\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t)) + (1-\lambda_i) c_3 \mathbf{r}_{3i}(\hat{\mathbf{a}}_i(t) - \mathbf{x}_i(t)) \end{aligned} \tag{2.12}$$

where $\hat{\mathbf{a}}_i$ and λ are referred to as the archive guide and the archive balance coefficient respectively. The archive guide, $\hat{\mathbf{a}}_i$, is chosen by a tournament selection on the archive. During each velocity update two or three solutions are chosen from the archive at random and the one with the biggest crowding distance value wins the tournament. Positive coefficient c_3 controls the amount of the archive guide's influence. The values of λ and $\mathbf{r}_3$ are chosen randomly from $[0, 1]$ and $[0, 1]^{n_x}$ respectively. The archive balance coefficient, λ, is initialized once for each particle at the very beginning and remains unchanged for the entire duration of the search. Recently, Erwin and Engelbrecht [20] explored dynamic adjustments of the archive balance coefficient. Contrary to the static λ in [64] (which was used in this paper as well), the work in [20] introduced five different dynamic approaches including updating λ randomly at each iteration, linearly decreasing λ from 1 to 0 and linearly increasing λ from 0 to 1. The results of [20] showed that the linearly increasing (LI) method outperformed all other methods on the 2-objective problems, enabling the MGPSO to obtain a highly diverse swarm at the early stages of the optimization process and exploit these regions slowly as λ increased. All particles are considered for entry to the archive, because MGPSO optimizes different objectives using separate subswarms and even if a particle fails to update its personal best, it might still be Pareto-optimal due to having good values for other objectives (it could be a boundary point of the Pareto-optimal set). The original MGPSO implementation used a crowding distance-based bounded archive.

2.8 Large-Scale Multi-Objective Optimization

Generally speaking, MOOPs that have more than 100 decision variables are referred to as large-scale MOOPs [47] [75]. Multi-objective optimization evolutionary algorithms (MOEAs) typically face many difficulties when solving large-scale problems. This can lead to a situation where a MOEA fails to explore the larger search space efficiently due to either premature convergence or converging to a region that is too large to explore [55]. In fact, it has been shown that the majority of MOEAs are incapable of solving large-scale MOOPs efficiently [47] [75] [88]. Therefore, in recent years many approaches have been proposed for large-scale multi- and many-objective

Algorithm 5 Multi-Guide Particle Swarm Optimization [64] [63]

procedure MGPSO(a, b)
  for each objective$m = 1, \ldots, n_m$ **do**
    create and initialize a swarm, S_m, of n_{sm} particles uniformly within a predefined hypercube of dimension n_x;
    Let f_m be the objective function;
    Le $S_m.y_i$ represent the personal best position of particle $S_m.x_i$, initialized to $S_m.x_i(0)$;
    Let $S_m.\hat{y}_i$ represent the neighborhood best position of particle $S_m.x_i$, initialized to $S_m.x_i(0)$;
    Initialize $S_m.v_i(0)$ to 0;
    Initialize $S_m.\lambda_i \sim U(0, 1)$;
  end for
  Let $t = 0$;
  while stopping condition is not true **do**
    for each objective $m = 1, \ldots, n_m$ **do**
      for each particle $i = 1, \ldots, S_m.n_s$ **do**
        if $f_m(S_m.x_i) < f_m(S_m.y_i)$ **then**
          $S_m.x_i = S_m.x_i(t)$
        end if
        for particles $\hat{i}$ with particle i in their neighborhood **do**
          if $f_m(S_m.y_i) < f_m(S_m.\hat{y}_{\hat{i}})$ **then**
            $S_m.\hat{y}_{\hat{i}} = S_m.y_i$
          end if
        end for
        Update the archive with the solution $S_m.x_i$;
      end for
    end for
    for each objective $m = 1, \ldots, n_m$ **do**
      for each particle $i = 1, \ldots, S_m.n_s$ **do**
        Select a solution, $S_m.\hat{a}_i(t)$ from the archive using the tournament selection;
        Update particle i's velocity using Eq. 2.12;
        Update particle i' position using Eq. 2.2;
      end for
    end for
    $t = t + 1$
  end while
end procedure

optimization. This section covers some of these approaches.

2.8.1 Weighted Optimization Framework

Proposed by Zille *et al.* [91], the weighted optimization framework is a novel approach for large-scale multi-objective optimization. WOF is a general framework and can be paired with any population-based algorithm for MOOP. WOF uses a transformation function to reduce the many decision variables of a large-scale MOOP into a small group of weight variables. Then, both the weight variables and the original decision variables of the problem are optimized by the population-based algorithm that WOF is paired with (such as SMPSO [54] or NSGA-II [18]).

Similar to many CC-based approaches, the main goal of WOF is to reduce the number of decision variables; however, unlike the CC-based approaches, WOF does not rely on co-evolution. Instead, WOF is based on weighting different decision variables with reference to different objectives. This approach indeed brings up the questions of how to group the decision variables, and what transformation functions to use. The original paper of WOF [91] covered some grouping methods and transformation functions, which are discussed in the following subsections.

Transformation Functions

Let Z be an optimization problem, n_x the number of decision variables, $\mathbf{w}$ an arbitrary weight vector, $\mathbf{x}$ an arbitrary decision vector, γ the number of decision variable groups, and l the size of each group (such that $\gamma \times l = n_x$). The transformation function $\psi(\mathbf{w}, \mathbf{x})$ is defined as:

$$\psi(\mathbf{w}, \mathbf{x}) = (\overbrace{w_1 x_1, \ldots, w_1 x_l}^{w_1}, \ldots, \overbrace{w_\gamma x_{n_x - l + 1}, \ldots, w_\gamma x_{n_x}}^{w_\gamma}) \tag{2.13}$$

Therefore, the large-scale MOOP is transformed into a smaller problem, with $\gamma < n_x$ decision variables (weight variables). Let $g_1, g_2, \ldots, g_\gamma$ denote γ different groups of decision variables, such $g(x_i) = g_j$ is the group that the i-th decision variable (x_i) belongs to. Below are some of the existing transformation functions:

- **Product Transformation (ψ_1)**: This is a simple function that only multiplies the value of each decision variable in a group by its respective weight:

$$x_{i,new} = w_j . x_{i,old}, \quad w_j \in [0, 2] \tag{2.14}$$

One obvious disadvantage of this method is not taking the decision variable bounds into account. As mentioned in [91], in problems where decision variables can accept both positive and negative values, the product transformation only produces values of the same sign (positive or negative) as the original decision variable before transformation. In order to address this issue the following transformation functions were introduced:

- **p-Value Transformation (ψ_2)**: This transformation method was proposed to address the shortcomings of the product transformation and in order to make more parts of the decision space accessible to the weight optimizer:

$$x_{i,new} = x_{i,old} + p.(x_{i,max} - x_{i,min}).(w_j - 1.0), \quad w_j \in [0, 2], \quad p \in [0, 1] \quad (2.15)$$

where $x_{i,max}$ and $x_{i,min}$ denote the upper and lower bounds of the i-th variable respectively. Parameter p controls the amount of change, in terms of a percentage of the width of the original decision variable centered around the original value of the decision variable ($x_{i,old}$).

- **Interval-Intersection Transformation (ψ_3)**: One major disadvantage of the last two transformation functions is the use of variable repairs. The variable repair mechanism sets the value of the transformed decision variable to the upper bound or the lower bound of the decision variable if its value becomes greater than the upper bound or less than the lower bound respectively. This could put WOF at an unfair advantage over other algorithms when tested on problems where the global best is found near the boundary values of the decision variables. This is more specifically the case in the ZDT problems. Therefore, inspired by [43], ψ_3 is formulated as:

$$x_{i,new} = w_j.x_{i,old}, \quad w_j \in [min_{x_h \in g(x_i)}(x_{h,min}/x_h), max_{x_h \in g(x_i)}(x_{h,max}/x_h)] \quad (2.16)$$

A closer look at Eq. 2.16 reveals that the main body of ψ_3 is similar to ψ_1, only the domains of the weight variables are altered. This alteration is done so that no decision variable in a group exceeds its upper and lower bounds.

Grouping Methods

This section covers some possible decision variable grouping methods for WOF.

- **Random Grouping:** As its name suggests, this method randomly assigns each decision variables to one of the groups.
- **Linear Grouping:** This method groups the decision variables based on their natural order. Given n_x decision variables and γ decision variable groups, the linear grouping method assigns the first $\frac{n_x}{\gamma}$ decision variables to the first group, the second $\frac{n_x}{\gamma}$ decision variables to the second group and so on.
- **Ordered Grouping:** The ordered grouping method first sorts the decision variables (in a given decision vector) based on their values and then classifies them into γ groups by applying linear grouping to these sorted decision variable indices.

Algorithm 6 The main body of WOF [91]

1: **procedure** WOF
2: **Input:** Z (optimization problem), A (optimization algorithm), G (grouping mechanism), ψ (transformation function)
3: **Output:**S (population of individuals)
4: $S \leftarrow RandomInitialization(N)$
5: **while** number of evaluations used is less than δ.total number of evaluations **do**
6: $S \leftarrow A(Z, S, t_1)$ ▷ Optimize Z using algorithm A for t_1 evaluations with S as a starting population
7: $\{\mathbf{x_1'}, \ldots, \mathbf{x_q'}\} \leftarrow$ Select q solutions from the first non-dominated front of S using crowding distance values
8: **for** k= 1 to q **do**
9: $W_k \leftarrow WeightingOptimization(\mathbf{x_k'}, \mathbf{Z}, \mathbf{A}, \mathbf{G}, \psi)$
10: **end for**
11: $S \leftarrow updatePopulation(W_1, \ldots, W_q, S)$
12: **end while**
13: Keep optimizing Z on S using A until all evaluations are used.
14: **return** S;
15: **end procedure**

The pseudo-code of WOF is presented in Algorithm 6. During the first $\delta.maxIter$ iterations, first the MOOP is optimized using S as a starting population for a maximum of t_1 function evaluations. Then, q solutions are chosen from S using crowding distance values. This is done by selecting the solutions with the greatest crowding distance values from the first non-dominated front. On each of these selected solutions, the weighting optimization (Algorithm 7) is carried out. The weighting optimization first groups the decision variables of the solution using the chosen grouping method,

Algorithm 7 WeightingOptimization [91]

procedure WEIGHTINGOPTIMIZATION
  Input: $\mathbf{x_k'}$ (solution), Z (optimization problem), A (optimization algorithm), G (grouping mechanism), ψ (transformation function)
  Output:W_k (population of weights)
  Divide the n_x variables into γ groups using G
  Apply the transformation function ψ to the solution $\mathbf{x_k'}$ for a transformed problem
  Randomly initialize a population of weights W_k
  With W_k as a starting weight population, optimize the transformed problem for t_2 function evaluations
  return W_k;
end procedure

Algorithm 8 UpdatePopulation [91]

procedure UPDATEPOPULATION
  Input: $W_1, \ldots, W_q$ (weight populations), S (population)
  Output:W_k (population of weights)
  for $k = 1$ to q **do**
    $w_k \leftarrow$ Select one individual from W_k
    $S_k' \leftarrow$ Apply w_k to population S
  end for
  $S \leftarrow$ Perform non-dominated sorting on $S \cup \{S_k'\}_{k=1,\ldots,q}$
end procedure

then produces a random population of weights, and finally returns the weight population after optimizing it for t_2 function evaluations. After the weighting optimization is done for q times, the population is updated using Algorithm 8. In Algorithm 8, one weight vector is selected from each of the q weight populations (the solution with the greatest crowding distance value from the first non-dominated front) and applied to all n_x-dimensional solutions of the main population (S). All of the resulting solutions along with the original population (S) then go through the non-dominated sorting algorithm to make up the new population for the next iteration.

2.8.2 Large-Scale Many-Objective Optimization Problems by Covariance Matrix Adaptation Evolution

Diversity (position)-related variables are variables that when perturbed, generate solutions that are non-dominated with reference to each other. On the other hand, convergence (distance)-related variables generate solutions that are dominated one by one when perturbed. Please note that a decision variable can be both convergence- and diversity-related. More information regarding these two types of variables is found in Appendix A (Section A.1). Based on this classification of decision variables, Ma *et al.* [47] proposed the multi-objective evolutionary algorithm based on decision variable analyses (MOEA/DVA). MOEA/DVA uses a sample set of solutions (of a predefined sample size NCA) to classify the decision variables. For all of the solutions in the sample set, the variable x_i ($1 \leq i \leq n_x$) is perturbed, the resulting solutions (from this perturbation) go through a non-dominated sorting after being evaluated. If all of the new solutions are non-dominated with reference to each other, x_i is a diversity-related parameter. On the other hand, if all non-dominated fronts have exactly one solution (solutions are dominated one by one), x_i is a convergence-related variable. If none of the aforementioned conditions is satisfied, x_i is a mixed variable. In MOEA/DVA, mixed variables are added to the set of diversity-related variables. Once the decision variables are classified into these two sets, MOEA/DVA further classifies the convergence-related variables into smaller groups on the basis of the interactions between them.

The Covariance Matrix Adaptation Evolution Strategy

The covariance matrix adaptation evolution strategy (CMA-ES) [23] is a novel evolutionary algorithm that uses a multivariate normal distribution to update position

vectors using the following equation:

$$\mathbf{x}_k^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}\left(\mathbf{0}, \mathbf{C}^{(g)}\right) \text{ for } k = 1, 2, \cdots, \lambda \tag{2.17}$$

where g is the generation index, $\mathbf{m}^{(g)}$ denotes the mean values of the decision variables at generation g, $\sigma^{(g)}$ is the step size, λ is the population size, and $\mathcal{N}\left(\mathbf{0}, \mathbf{C}^{(g)}\right)$ is a multivariate normal distribution with mean of zero and covariance matrix $\mathbf{C}^{(g)}$. The mean of the decision variables is updated using:

$$\mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + \sum_{i=1}^{\mu} w_i \left(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}\right) \tag{2.18}$$

where $\mu \geq \lambda$ is the number of solutions to be selected from the population, w_i denotes the positive weight coefficient for the i-th selected solution, $\mathbf{x}_{i:\lambda}^{(g+1)}$ denotes the i-th best solution in the new population, as created by the position update equation. The covariance matrix is updated using:

$$\begin{aligned}\mathbf{C}^{(g+1)} = &\left(1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i\right)\mathbf{C}^{(g)} \\ &+ c_1 \mathbf{p}_c^{(g+1)} \mathbf{p}_c^{(g+1)^T} + c_\mu \sum_{i=1}^{\lambda} w_i \mathbf{y}_{i:\lambda}^{(g+1)} \left(\mathbf{y}_{i:\lambda}^{(g+1)}\right)^T\end{aligned} \tag{2.19}$$

where $\mathbf{y}_{i:\lambda}^{(g+1)} = \left(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}\right)/\sigma^{(g)}$, $c_1 = 2/n_x^2$, and $c_\mu = \min\left(1 - c_1, 1/\left(n_x^2 \sum_{i=1}^{\mu} w_i^2\right)\right)$.

S3-CMA-ES

In order to adapt CMA-ES for large-scale many-objective optimization, Chen *et al.* [6] proposed S3-CMA-ES. S3-CMA-ES uses the decision variable grouping approach of MOEA/DVA to classify the variables into convergence- and diversity-related variables. Then, similar to MOEA/DVA, the convergence-related variables (as detected in the previous step) are classified into smaller subgroups based on their interactions. This step returns one set of non-separable variables and a number of groups with interacting separable variables. Each group of the separable variables has a predefined size and includes a subset of all variables of a large-scale problem. The interacting variables are determined in the following way: For an n_m-objective minimization problem $F = \{f_1, f_2, \ldots, f_{n_m}\}$, decision variables x_i and x_j are interacting if scalars a, b, c, and d, a decision vector $\mathbf{x}$, and at least one objective function f_k can be found

such that:

$$f_k(\mathbf{x})|_{x_i=a,x_j=c} > f_k(\mathbf{x})|_{x_i=b,x_j=c} \tag{2.20}$$

, and:

$$f_k(\mathbf{x})|_{x_i=a,x_j=d} < f_k(\mathbf{x})|_{x_i=b,x_j=d} \tag{2.21}$$

where $f_k(\mathbf{x})|_{x_i=a,x_j=c}$ is the k-th objective value of decision vector $\mathbf{x}$ when the i-th and the j-th decision variables are replaced by a and c respectively.

After the decision variables are classified, M subpopulations are initialized in S3-CMA-ES. Each subpopulation is controlled by an independent CMA-ES, where all subpopulations have access to the diversity-related variables and optimize each group of convergence-related variables one by one. Using the sum of all objective values as the fitness measure, S3-CMA-ES uses the convergence threshold Δ to determine whether a subpopulation has converged. When all subpopulations have converged, the diversity-related variables are optimized. Then, the global best value of each subpopulation is set to infinity (so that the convergence threshold is no longer satisfied), and the convergence-related variables are optimized again. This process is repeated until all function evaluations have been consumed.

2.8.3 Large-Scale Multi-Objective Optimization Using The Competitive Swarm Optimizer

Tian *et al.* [75] proposed a large-scale multi-objective optimization algorithm based on the competitive swarm optimizer [7]. Similar to CMOPSO, LMOCSO relies on the competitions between particles where a particle with the worse fitness value learns from another particle with the better fitness value. Based on this competition, LMOCSO proposes the following velocity and equation equations:

$$\mathbf{v_l}(t+1) = r_0\mathbf{v_l}(t) + r_1\left(\mathbf{x_w}(t) - \mathbf{x_l}(t)\right) \tag{2.22}$$

$$\mathbf{x_l}(t+1) = \mathbf{x_l}(t) + \mathbf{v_l}(t+1) + r_0\left(\mathbf{v_l}(t+1) - \mathbf{v_l}(t)\right) \tag{2.23}$$

where l and w are the indices for the loser and the winner particles respectively, with r_0 and r_1 being two random numbers from $[0, 1]$.

In order to compare different particles to determine the winners and the losers in a multi-objective context, there exists a need for an objective aggregation score. LMOCSO uses the the shift based density estimation (SDE) [42] method to achieve this. Using the SDE method, the fitness score of an arbitrary particle $\mathbf{p}$ belonging to

a set P is calculated as:

$$\text{Fitness } (\mathbf{p}) = \min_{\mathbf{q} \in P \setminus \{\mathbf{p}\}} \sqrt{\sum_{i=1}^{M} \left(\max\left\{0, f_i(\vec{q}) - f_i(\vec{p})\right\}\right)^2} \tag{2.24}$$

where f is the optimization problem, and $f_i(\mathbf{p})$ denotes the i-th objective value of $\mathbf{p}$. The SDE fitness, which is to be maximized, calculates each individual's fitness relative to other individuals belonging to the same set. The SDE fitness measure has been used in other multi-objective evolutionary algorithms before [46] [41]. When doing pairwise comparisons between $\mathbf{p}$ and some other particle $q \in P$, SDE shifts the i-th objective value of q (denoted by $f_i(q)$) to its counterpart in $\mathbf{p}$ (denoted by $f_i(\mathbf{p})$) if $f_i(q) < f_i(\mathbf{p})$ for density estimations.

The pseudo-code of LMOCSO is listed in Algorithm 9. The general format of LMOCSO is similar to that of CMOPSO's, the main difference being the use of the environmental selection of RVEA [8] which is discussed in the next section. The pseudo-code of LMOCSO's competition-based learning method is provided in Algorithm 10. This algorithm first selects two particles at random, removes them from the current set, and determines the winner and the loser based on SDE (Eq. (2.24)). Then, based on Eq. (2.22) and Eq. (2.23), new offspring are generated and put in a new set named $P^{'}$. The population for the next iteration will be made up of $P \cup P^{'}$ after undergoing RVEA's environmental selection.

The Environmental Selection of RVEA

As mentioned above, LMOCSO uses RVEA's environmental selection. This selection is based on uniformly distributed reference vectors in the objective space. At the beginning of the algorithm, N vectors are uniformly initialized in an n_m-dimensional vector space (line 5 in Algorithm 9) and put in a set named R. Then, during each selection, first each particle in $P \cup P^{'}$ is assigned to the closest reference vector to it (according to the angles between them). Each reference vector is therefore allocated a set of individuals, and chooses only one (assuming it is assigned to at least one) with the minimum angle-penalized distance (APD). For particle $\mathbf{p}$ and reference vector $\vec{r}$, the APD is formulated as:

$$APD(\mathbf{p}, \vec{r}) = \left(1 + M \cdot \left(\frac{t}{t_{\max}}\right)^{\alpha} \cdot \frac{\langle \vec{f}(\vec{p}), \vec{r} \rangle}{\min_{\vec{s} \in R, \vec{s} \neq \vec{r}} \langle \vec{s}, \vec{r} \rangle}\right) \cdot \|\vec{f}(\vec{p})\| \tag{2.25}$$

where $\vec{f}(\vec{p})$ is the objective vector of $\mathbf{p}$, $\langle \vec{s}, \vec{r} \rangle$ is the angle between $\vec{s}$ and $\vec{r}$, α is a penalty parameter, and t and $t_{\max}$ denote the current iteration and maximum number of iterations respectively. Finally, M is the number of objectives.

Algorithm 9 The general framework of LMOCSO [75]

```
1: procedure LMOCSO
2:     Input: N (population size)
3:     Output:P (final population)
4:     P ← RandomInitialization(N)
5:     R ← UniformRandomReferenceVector(N)
6:     while stopping condition not satisfied do
7:         P' ← CompetitionBasedLearning(P)
8:         P ← RVEA_EnvironmentalSelection(P ∪ P', R)
9:     end while
10:    return P;
11: end procedure
```

2.8.4 Large-scale Multi-objective Optimization using Problem Reformulation

He *et al.* [25] proposed the large-scale multi-objective optimization framework based on problem reformulation termed LSMOF. Similar to WOF, LSMOF's goal is to turn the large-scale problem into a low-dimensional one with completely new weight variables that indirectly optimize the original problem. LSMOF selects r reference solutions from the current population. For each reference solution, two direction vectors and two weight variables are defined. Let $\mathbf{s}_1$ be an n_x-dimensional decision vector transformed into a new bi-objective decision space. Furthermore, let $\mathbf{o}$ and $\mathbf{t}$ be the lower and upper bounds of the n_x-dimensional search space respectively. Direction vectors $\mathbf{v}_l$ (from $\mathbf{o}$ to $\mathbf{s}_1$) and $\mathbf{v}_u$ ($\mathbf{s}_1$ to $\mathbf{t}$) are formulated as:

$$\begin{aligned} \mathbf{v}_l &= \mathbf{s}_1 - \mathbf{o} \\ \mathbf{v}_u &= \mathbf{t} - \mathbf{s}_1 \end{aligned} \tag{2.26}$$

Now let $l_{max} = \|\mathbf{t} - \mathbf{o}\|$ be the maximum length of the decision space, with $\mathbf{p}_1$ and $\mathbf{p}_2$ denoting the respective intersections between $\mathbf{v}_l$ and $\mathbf{v}_u$ and the Pareto-optimal set.

Algorithm 10 LMOCSO's competition-based learning [75]

procedure CompetitionBasedLearning
  Input: P (current population)
  Output: $P^{'}$ (new population)
  $Fitness \leftarrow Calculate the fitness of each particle using Eq.$
  $P^{'} \leftarrow \emptyset$
  while $|P| > 1$ **do**
    $\{p, q\} \leftarrow$ Randomly select two particles from P;
    $P \leftarrow P - \{p, q\}$
    if $Fitness(p) < Fitness(q)$ **then**
      $x_l \leftarrow p$
      $x_w \leftarrow q$
    end if
    if $Fitness(q) < Fitness(p)$ **then**
      $x_l \leftarrow q$
      $x_w \leftarrow p$
    end if
    Update x_l by learning from x_w by Eq. (2.23)
    Mutate x_l and x_w by polynomial mutation
    $P^{'} \leftarrow P^{'} \cup \{x_l, x_w\}$
  end while
  return $P^{'}$;
end procedure

The values of $\mathbf{p}_1$ and $\mathbf{p}_2$ can be formulated as:

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{o} + \lambda_{11} \frac{\mathbf{v}_l}{\|\mathbf{v}_l\|} l_{\max} \\ \mathbf{p}_2 &= \mathbf{t} - \lambda_{12} \frac{\mathbf{v}_u}{\|\mathbf{v}_u\|} l_{\max}, \end{aligned} \tag{2.27}$$

where λ_{11}, $\lambda_{12} \in [0, 0.5]$ are two weight variables. Therefore, for r reference solutions, $2r$ subproblems are created. More formally, for each reference solution $j \in \{1, \ldots, r\}$, the following subproblems are generated:

$$\begin{aligned} z_{j1}\left(\lambda_{j1}\right) &= F\left(\mathbf{o} + \lambda_{j1} \frac{\mathbf{v}_l}{\|\mathbf{v}_l\|} l_{\max}\right) \\ z_{j2}\left(\lambda_{j2}\right) &= F\left(\mathbf{t} - \lambda_{j2} \frac{\mathbf{v}_u}{\|\mathbf{v}_u\|} l_{\max}\right), \end{aligned} \tag{2.28}$$

This leads to a new set of subproblems:

$$Z'(\mathbf{\Lambda}) = \left\{z_{11}\left(\lambda_{11}\right), z_{12}\left(\lambda_{12}\right), \ldots, z_{r1}\left(\lambda_{r1}\right), z_{r2}\left(\lambda_{r2}\right)\right\} \tag{2.29}$$

where $\mathbf{\Lambda}$ is the set of weight vectors:

$$\mathbf{\Lambda} = \{\lambda_{11}, \lambda_{12}, \ldots, \lambda_{r1}, \lambda_{r2}\} \tag{2.30}$$

Since the aforementioned $2r$ subproblems are all single-objective optimization problems, there exists a need for a single-objective optimization algorithm. The work in [25] used the differential evolution (DE) as the single-objective optimization algorithm and the HV measure of all obtained solutions from an arbitrary set of weight vectors as their collective fitness value. Unlike WOF, LSMOF does not use any grouping technique; instead, it assigns some weight variables to reference directions in hopes of reaching the true Pareto-optimal front.

2.8.5 Adaptive Offspring Generation for Evolutionary Large-Scale Multi-Objective Optimization

In a lot of multi-objective evolutionary algorithms (MOEAs) such as RVEA [8] and IBEA [94], offspring generation plays an important role. Generating offspring with the goal of improving the overall convergence and diversity is specifically of high importance in large-scale environments. He *et al.* [24] proposed an adaptive offspring generation framework, termed DGEA. DGEA is a general framework, into which a lot

of MOEAs that are based on offspring generation can be embedded. One interesting aspect of DGEA is its use of direction vectors for offspring reproduction. DGEA uses two kinds of direction vectors, the first kind that utilizes dominated solutions to aim for the Pareto optimal set (convergence improvement), whilst the second type takes advantage of non-dominated solutions to further spread the solutions (diversity improvement). The general framework of DGEA is listed in Algorithm 11. During each iteration, a preselection strategy is conducted to generate the parent population Q_0. Then, a reproduction mechanism is applied to Q_0 and the offspring from the previous iteration (Q). Finally, an algorithm-specific environmental selection is used to select N (population size) solutions from the current population P and the new offspring Q.

Preselection

DGEA's preselection strategy is listed in Algorithm 12. The preselection has two main parts that concentrate on convergence and diversity enhancements. First, the solutions go through a non-dominated sorting producing different non-dominated fronts F_i $1 \leq i \leq m$ where m is the index of the first non-dominated front that, if considered, would make the number of selected solutions exceed the number of reference vectors ($\sum_{i=1}^{m-1} \mid \mathrm{F}_i| \leq |V|$ and $\sum_{i=1}^{m} \mid \mathrm{F}_i| > |V|$). If the number of non-dominated solutions is greater than the number of direction vectors, the first $m-1$ non-dominated fronts are preselected and the m-th one is placed into P; otherwise, all non-dominated solutions are preselected. Then, the diversity enhancement phase is applied, which is based on RVEA's [8] environmental selection that uses uniformly distributed reference vectors (see Section 2.8.3). The ideal vector $\mathbf{z}$ is an objective vector that contains the minimum objective value for each objective in the current population. The vector between each solution's objective vector and the ideal vector is used for APD calculations (see Eq. (2.25)) as it could be a metric for convergence, while the use of APD aims to improve diversity among solutions. The main goal of this preselection strategy is to avoid ending up in a local optimum in large-scale environments by selecting well-converged solutions (non-dominated) and selecting well-distributed but less-converged solutions for reproduction.

Reproduction

The reproduction strategy of DGEA is listed in Algorithm 13. In this part, r direction vectors (from each solution to a group of selected promising solutions) are

constructed. First, the solutions are classified into two groups; the solutions that are non-dominated and the ones that are dominated by at least one solution. Then, $\mathbf{p}_s$ is randomly selected from the set of non-dominated solutions as the starting point. Next, the endpoints (a total of r solutions) are selected as in lines 8 to 11 in Algorithm 13. Because DGEA uses an environmental selection strategy to select the next population, when there are a lot of non-dominated solutions, the solutions are mostly well-distributed (good diversity) as this is usually ensured in environmental selection approaches. Therefore, if the number of dominated solutions is less than the number of direction vectors, all dominated solutions are selected as endpoints. Moreover, when the number of dominated solutions is greater than average, the non-dominated region is controlled by a few solutions which in turn results in poor diversity.

When the endpoints are determined, direction vectors are calculated (from the starting point to each end point). Then, $N_{sub} = \lfloor N/r \rfloor$ different solutions are generated towards each direction vector, making up N new offspring. As seen in Algorithm 13, in lines 15 to 18 the distances between the non-dominated solutions and the starting point along each direction vector is calculated, and the variance of these values (σ^2) is used in the normal distribution $\gamma = \mathcal{N}(0, \sigma^2)$, forming the following offspring generation equation:

$$\mathbf{x}_j = \gamma \cdot \mathbf{d}_j + \mathbf{p}_s \tag{2.31}$$

where $\mathbf{p}_s$ is the starting point, $\mathbf{d}_j$ is the direction vector, and $\mathbf{x}_j$ is the new solution.

Algorithm 11 The general framework of DGEA [24]

```
1: procedure DGEA
2:     Input: N (population size), r (number of direction vectors)
3:     Output:P (final population)
4:     P, Q ← RandomInitialization(N)
5:     V ← UniformRandomReferenceVector(N)
6:     while stopping condition not satisfied do
7:         Q_0, F_1 ∪ F_2 ∪ ... ← Preselection(P)
8:         Q ← Reproduction(Q_0, F_1 ∪ F_2 ∪ ..., r, N)
9:         P ← EnvironmentalSelection(P ∪ Q, N)
10:    end while
11:    return P;
12: end procedure
```

Algorithm 12 The preselection of DGEA [24]

procedure PRESELECTION
  Input: P (population), r (number of direction vectors), V (uniform reference vectors)
  Output:Q_0 (preselected population), F_i (solutions on the ith front).
  ▷ Convergence based selection
  $F_1, F_2, \ldots, F_m \leftarrow ND\_SORT(P)$
  $N_d \leftarrow |F_1|$
  if $N_d \geq r$ **then**
    $Q_0 \leftarrow F_1 \cup F_2 \cup \cdots \cup F_{m-1}$
    $P \leftarrow F_m$
  else
    $Q_0 \leftarrow F_1$ ▷ Non-dominated solutions
    $P \leftarrow F_2 \cup \cdots \cup F_m$ ▷ Other solutions (dominated ones)
  end if
  ▷ Now select based on diversity
  $\mathbf{z} \leftarrow$ The ideal point of P (the minimum of each objective)
  for $i \leftarrow 1 : |P|$ **do**
    $\mathbf{f_i'} \leftarrow \mathbf{f_i} - \mathbf{z}$
    for $j \leftarrow 1 : |V|$ **do**
      $cos\theta_{i,j} \leftarrow \frac{\mathbf{f_i'} \cdot \mathbf{v}_j}{||\mathbf{f_i'}||}$
    end for
    $k \leftarrow argmax_{j \in \{1,\ldots,|V|\}} cos\theta_{i,j}$
    $S_k \leftarrow \mathbf{p}_k$
  end for
  for $j \leftarrow 1 : |V|$ **do**
    for $i \leftarrow 1 : |P|$ **do**
      $d_{i,j} \leftarrow APD(\theta_{i,j}, \mathbf{f_i'})$
    end for
    $k \leftarrow argmin_{i \in \{1,\ldots,|S_j|\}} d_{i,j}$
    $Q_0 \leftarrow Q_0 \cup \{\mathbf{p}_k\}$
    Update the front members of Q_0
  end for
  return P;
end procedure

Algorithm 13 The reproduction of DGEA [24]

procedure REPRODUCTION
  Input: P (population), N (population size), r (number of direction vectors), V (uniform reference vectors)
  Output:Q (offspring population)
  $P_n \leftarrow F_1$ ▷ Non-dominated solutions
  $P_d \leftarrow F_2 \cup \ldots$ ▷ Other solutions (dominated solutions)
  Randomly select $\mathbf{p}_s$ from P_n
  $N_{sub} \leftarrow \lfloor N/r \rfloor$
  if $|P_d| < r$ **then**
    Select $r - |P_d|$ solutions at random from $P_n \setminus \mathbf{p}_s$ and put them in Q_0
    $Q \leftarrow P_d \cup Q_0$
  else
    $Q \leftarrow$ Randomly select r solutions from P_n
  end if
  for $i \leftarrow 1 : r$ **do**
    $\mathbf{d}_i \leftarrow \frac{\mathbf{p}_s - \mathbf{q}_i}{||\mathbf{p}_s - \mathbf{q}_i||}$
    for $k \leftarrow 1 : |P_n|$ **do**
      $\lambda_k \leftarrow (\mathbf{p}_{\mathbf{n}k} - \mathbf{p}_s).\mathbf{d}_i^T$
    end for
    $\sigma^2 \leftarrow$ the variance of $\{\lambda_1, \ldots, \lambda_{|P_n|}\}$
    for $j \leftarrow 1 : N_{sub}$ **do**
      $\gamma \leftarrow \mathcal{N}(0, \sigma^2)$
      $\mathbf{s}_j \leftarrow \gamma.\mathbf{d}_j + \mathbf{p}_s$
    end for
    $Q_i \leftarrow \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{N_{sub}}, \}$
  end for
  $Q \leftarrow Q_1 \cup Q_2 \cup \cdots \cup Q_r$
end procedure

2.8.6 Multi-Objective Orthogonal Opposition-Based Crow Search Algorithm

Crow Search Algorithm

Proposed by Askarzadeh [3], the crow search algorithm (CSA) is a novel evolutionary algorithm for single-objective optimization. CSA aims to simulate the social behaviour of crows. Crows are known to follow other birds to where they hide their food in order to steal it. Crows that steal foods from other birds or crows will be more careful in protecting their own food, using their past experience of thievery to predict other crows' plans [10]. Based on this interesting information, the CSA proposes the following position update equation:

$$x^{i,\,\mathrm{iter}\,+1} = \begin{cases} x^{i,\,\mathrm{iter}} + r_i \times fl^{\mathrm{i,iter}} \times \left(m^{j\,,\,\mathrm{iter}} - x^{i,\,\mathrm{iter}}\right) & r_j \geqslant AP^{j\,,\,\mathrm{iter}} \\ \text{a random position} & \text{otherwise} \end{cases} \tag{2.32}$$

where $m^{j\,,\,\mathrm{iter}}$ is crow j's hiding place at iteration $iter$, $x^{i.\,\mathrm{iter}}$ is the i-th crow's position at iteration $iter$, i is the index of the crow that is following, j is the index of the crow that is being followed by the i-th crow, $fl^{\mathrm{i,iter}}$ is the flight length of the i-th crow at iteration $iter$, $AP^{j\,,\,\mathrm{iter}}$ is the awareness probability of the j-th crow at iteration $iter$, and r_i and r_j are two numbers randomly chosen from $[0, 1]$. Put another way, the above equation describes a situation where the j-th crow is being followed by the i-th one. If crow j notices that it is being followed, it will change its position to a random one to fool crow i. Otherwise, it will keep heading to its hiding place, also revealing this position to crow i. Please note that from an implementation point of view, the hiding place of a crow could be interpreted as the personal best position found by the crow up to any point in the optimization process. Moreover, each crow randomly picks one of the crows and moves towards it, unlike in PSO where the global best is directly used for all particles.

Crow Search Algorithm for Large-Scale Multi-Objective Optimization

Looking to adapt the CSA for large-scale multi-objective optimization, Rizk-Allah *et al.* [61] proposed the multi-objective orthogonal opposition-based crow search algorithm (M2O-CSA). This algorithm uses the same position update of CSA, with a solution selected from the archive acting as the hiding place of each crow. M2O-CSA also uses a novel method, namely the orthogonal opposition strategy (OOS) (a combination of orthogonal arrays and opposition-based learning [76]), for improved

performance on large-scale problems.

Orthogonal Arrays and Orthogonal Crossover

Let S be a system with K factors (variables), where each factor can get one of the total Q levels (values). To find the best value for S and since trying out all Q^K combinations is not efficient, orthogonal arrays are some of the most popular methods for selecting a small and efficient subset of all possible combinations. For K factors, Q levels and M combinations (experiments), an orthogonal array is denoted by $L_M(Q^K)$ [81]. For example, $L_9(3^4)$ is shown below:

$$.L_9\left(3^4\right) = \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 3 & 3 & 2 & 1 \end{array}\right] \tag{2.33}$$

where each row represents an experiment or a combination, and factors are the assigned values. M2O-CSA also uses $L_9(3^4)$, but doing crossover between two parent solutions. During this crossover, two n_x-dimensional solutions are randomly generated. Then, from these two solutions, three different levels are generated and decision variables are divided into four groups. Finally, nine new solutions are generated according to $L_9(3^4)$, where each decision variable is a factor (variable).

2.8.7 Large-Scale Many-Objective Particle Swarm Optimizer Based on Alpha-Stable Mutation and Logistic Function

Cheng *et al.* [9] proposed the large-scale many-objective PSO based on Alpha-stable mutation (LMPSO). LMPSO draws inspiration from previous work in [22], where the Levy flight was successfully integrated into the PSO, enabling the particles to do long jumps and escape premature convergence. LMPSO uses the Alpha-stable distribution, which is a more general family that includes the Levy flight. The Alpha-distribution is used to mutate each individual around either its own position, its assigned global

best position, or the middle point between its position and its assigned global best (from the external archive), selecting between these three mutation operators based on predefined probabilities.

Dynamic Parameter Adjustment

LMPSO uses a dynamic parameter adjustment approach which aims to emphasize exploration at the beginning of search, and gradually shift the focus to exploitation by dynamically changing the values of the parameters related to PSO (c_1, c_2, and ω), and the ones related to the Alpha-stable distribution (α and β). In order to dynamically change the values of these variables, the work in [9] utilizes the distribution properties of Logistic function, dynamically updating the value of each parameter using the following equation:

$$f(x) = \frac{\max\left(L_{\text{Ini}}, L_{\text{Final}}\right) - \min\left(L_{\text{Ini}}, L_{\text{Final}}\right)}{1 + \exp\left(-k\left(x - x_0\right)\right)} + \min\left(L_{\text{Ini}}, L_{\text{Final}}\right) \tag{2.34}$$

where L_{Ini} and L_{Final} denote the initial and final values of parameter L respectively. Moreover, $x_0 = c \times maxIter$ and $k = d \times \frac{L_{\text{Final}} - L_{\text{Ini}}}{maxIter}$, where c and d are two constants and $maxIter$ denotes the maximum number of iterations. The values of L_{Ini} and L_{Final} are parameter-specific; for example, the values of c_1 and c_2 in [75] started with 2.5 and ended up with 0.75.

Dominance Resistance Solutions

LMPSO also addresses the issue of dominance resistance solutions (DRSs). The DRSs, which are typically observed in many-objective problems, are defined as non-dominated solutions that have very poor values for at least one objective, and optimal values for others [1] [29]. For example, in an external archive based on Pareto-dominance for a two-objective problem, objective vectors $(0.1, 2)$ and $(0, 100)$ both remain unaffected they are non-dominated with reference to each other. Even when the crowding distance is used as a secondary measure to remove the extra solutions, such solutions can stay in the archive as they are often isolated in the POF, thus having a preferable crowding distance score. In order to address this, LMPSO normalizes the objective values of the archive solutions in $[0, 1]$, sets the objectives values less than σ (the dominance resistance error) to zero, and finally remove the dominated solutions. The value 1e-6 was recommended for σ in [9].

2.8.8 A Center-Based Mutation for the Third Generalized Differential Evolution

The differential evolution (DE) [71] is an evolutionary algorithm for single-objective optimization. DE uses mutation and crossover operators to produce new individuals. For each gene j, the mutation operator is formulated as:

$$v_{j,i} = x_{j,i_1} + F \cdot (x_{j,i_2} - x_{j,i_3}) \tag{2.35}$$

where x_{j,i_1}, x_{j,i_2} and x_{j,i_3} are three distinct individuals randomly selected from the current population and F is the mutation factor. The resulting vector from the above equation goes into a crossover (which is typically a binary crossover) with the original individual based on a predefined crossover rate (C_r), producing a new target vector that replaces the original individual if it has a better fitness.

Similar to PSO, DE has also been adapted to solve multi-objective problems. Generalized differential evolution (GDE) [40], GDE2 [38] and GDE3 [39] [4] are some examples of multi-objective optimization algorithms based on DE. GDE was proposed to solve multi-objective optimization problems with K constraints, using constraint-domination concepts. Looking to extend GDE, GDE2 incorporated the crowding distance measure into GDE. In GDE3, each individual in the set of offspring is generated using three randomly selected solutions. Looking to improve GDE3's performance on large-scale problems, Hiba *et al.* [27] proposed a center-based mutation for the third generalized differential evolution (CGDE3). This novel center-based mutation approach, which replaces the traditional GDE3 mutation for a pre-defined number of iterations, randomly picks three individuals from the current population, and uses the mean of these individuals for the mean of a Gaussian distribution as formulated below:

$$x_{\text{center}} = \left(\frac{x_{i1} + x_{i2} + x_{i3}}{3} \right) \tag{2.36}$$

$$x_{\text{Ncenter}} = N\left(x_{\text{center}}, \sigma\right) \tag{2.37}$$

The standard deviation of this Gaussian distribution (σ) is calculated as $\frac{max_j - min_j}{6}$ where max_j and min_j refer to the maximum and the minimum values of the j-th decision variable (in the current population) respectively. Finally, the new mutation operator is formulated as:

$$v_i = x_{\text{Ncenter}} + F \cdot (x_{i4} - x_{i5}) \tag{2.38}$$

where x_{i4} and x_{i5} are also two randomly selected distinct individuals.

2.8.9 Coevolutionary Operations for Large Scale Multi-objective Optimization

Proposed by Antonio *et al.* [2], operational decomposition (OD) is a novel framework intended to improve the crossover operations in MOEAs while using no additional function evaluations. A motivation of OD is to reduce the computational cost, by stating the multiple subpopulations of the CC-based algorithms which consume additional functional evaluations as a drawback of such approaches. In OD, decision variables are randomly classified into S groups, thus forming S subcomponents. OD uses set X consisting of N full-dimensional individuals ($X = \{x^1, \ldots, x^N\}$). To produce offspring, and for each solution $x_i \in X$, T closest decision vectors are selected and put in a set named $B(i)$. Then, for each of the S subcomponents of x_i, two solutions are randomly selected from $B(i)$ and go through a crossover operation (the work in [2] used the simulated binary crossover (SBX) [15] for this purpose). The vector resulting from this crossover is the subvector corresponding to the appropriate subcomponent. Therefore, OD utilizes the CC framework not by using $\frac{n_x}{S}$-dimensional individuals, but by using full-dimensional individuals and generating offspring based on smaller subcomponents of them.

2.9 Other Advances in Evolutionary Algorithms

This section covers other advances in recent years in evolutionary computation for regular and large-scale single- and multi-objective optimization.

2.9.1 Contribution-Based Approaches

The cooperative coevolutionary (CC) framework, as discussed in previous sections, has shown competitive performance in solving large-scale problems. However, the algorithms that use this framework are typically costly in terms of the computation budget (which is usually the number of function evaluations). Moreover, a lot of CC-based approaches divide the computation budget between the subpopulations equally, whereas there might be subpopulations that contribute more to the search. Therefore, in recent years, contribution-based approaches have been developed to address this issue by defining contribution metrics, and allocating more computation

budget to subpopulations that have better scores. The main goal of such approaches is to dynamically allocate computational resources.

Contribution-Based Cooperative Coevolution

Proposed by Omidvar *et al.* [58], contribution-based CC selects the subpopulation with the most contribution to undergo evolution during each iteration. Both CBCC1 and CBCC2 accumulate the contribution scores from the beginning of the optimization process. In their original experiments, both CBCC1 and CBCC2 outperformed the original CC algorithms, especially on imbalanced problems where subpopulations had different contributions to the objective value. The main disadvantage of CBCC1 and CBCC2 is that they accumulate the contributions from the beginning of search. Since the changes in the objective value are usually more significant early on, CBCC's contribution score could be overwhelmed by initial changes in the fitness value without taking into account real-time objective improvements or adapting to them. In order to address these issues, CBCC3 [56] was proposed which forgoes the historical contribution information in favor of real-time changes.

CCFR

CCFR is another contribution-based CC [83]. CCFR calculates the contribution of each subpopulation using both historical data and real-time changes, as an average:

$$\Delta F_i = \frac{\Delta \hat{F}_i + |f(\hat{\mathbf{x}}_{\text{best}}) - f(\mathbf{x}_{\text{best}})|}{2} \tag{2.39}$$

For the i-th subpopulation (P_i), $\Delta \hat{F}_i$ is the last contribution of the subpopulation, and ΔF_i is initially equal to zero. Similarly, $\hat{\mathbf{x}}$ and $\mathbf{x}$ denote the best overall solution before and after P_i's coevolutionary cycle. Eq. (2.39) takes both real-time and historical objective improvement data into account, with the former having a greater effect on the contribution score. During each coevolutionary iteration, CCFR selects the subpopulation with the best contribution score to undergo coevolution. Given M subcomponents and a population size of N, CCFR spends $N \times M$ function evaluations early on to:

- Initialize the contribution score of each subpopulation, and

- calculate the best overall solution using:

$$\underset{\mathbf{x}\in Z}{\operatorname{argmin}} f(\mathbf{x}) = \left(\underset{\mathbf{x}_1\in P_1}{\operatorname{argmin}} f\left(\mathbf{x}_1, \ldots\right), \ldots, \underset{\mathbf{x}_M\in P_M}{\operatorname{argmin}} f\left(\ldots, \mathbf{x}_M\right) \right) \tag{2.40}$$

  where Z is a set containing all possible combinations of the individuals in all subpopulations.

CCFR2

Looking to improve CCFR's performance, Yang *et al.* [84] proposed CCFR2. In light of the fact that the decomposition strategy in the CC framework could divide the decision variables into unequal groups where some groups are assigned more variables, CCFR2 allocates more individuals to subpopulations with more variables, unlike CCGA [60], CCFR [83], CBCC [58] and many other CC-based approaches that assume an equal number of individuals for all subpopulations. More specifically, to subpopulation P_i with D_i assigned decision variables, CCFR2 allocates $D_i + d$ individuals where d is a positive integer. CCFR2 also modifies CCFR's contribution score as:

$$\Delta F_i = w \cdot \Delta \hat{F}_i + \frac{\left| f\left(\hat{\mathbf{x}}_{\text{best}}\right) - f\left(\mathbf{x}_{\text{best}}\right) \right|}{n_i} \tag{2.41}$$

where n_i is the number of function evaluations spent by P_i during the coevolutionary cycle and $w \in [0, 1]$. Eq. (2.41) takes the number of fitness evaluations into account, since it uses subpopulations of different size. This equation also introduces control parameter w, which controls the influence of the historical data on the contribution score.

2.9.2 Information Feedback Models

In most metaheuristic algorithms, a lot of search information from previous iterations is not considered when updating individuals. Recently, Wang and Tan [80] proposed the idea of using information feedback models (IFMs) in metaheuristic algorithms to make use of potential useful information from previous iterations. In their model, Wang and Tan proposed three different sets of feedback models, based on the number of previous individuals to consider. More formally, in their models $k \geq 1$ previous individuals can be considered, but they only used $k \in \{1, 2, 3\}$ to avoid over-complicated models. In a run of the single-objective PSO, let n_s be the swarm size, and x_i^t the i-th particle at iteration t with x_i and f_i^t denoting its position and fitness value respectively. Moreover, let y_i^{t+1} be the updated particle (by the vanilla PSO) with f_i^{t+1}

being its fitness value. The proposed models are the following:

- $R1$ and $F1$: These models update the i-th particle (x_i^{t+1}) using the j-th one as:

$$x_i^{t+1} = \alpha y_i^{t+1} + \beta x_j^t \tag{2.42}$$

Weight parameters α and β satisfy $\alpha + \beta = 1$ and are formulated as:

$$\alpha = \frac{f_j^t}{f_i^{t+1} + f_j^t}, \beta = \frac{f_i^{t+1}}{f_i^{t+1} + f_j^t} \tag{2.43}$$

The model in Eq. (2.42) is called $F1$ if $j = i$, and called $R1$ if j is a random integer from $\{1, \ldots, n_s\}$.

- $R2$ and $F2$: These models update the i-th particle (x_i^{t+1}) using the following:

$$x_i^{t+1} = \alpha y_i^{t+1} + \beta_1 x_{j_1}^t + \beta_2 x_{j_2}^{t-1} \tag{2.44}$$

Weight parameters α, β_1, and β_2 satisfy $\alpha + \beta_1 + \beta_2 = 1$ and are formulated as:

$$\begin{aligned} \alpha &= \frac{1}{2} \bullet \frac{f_{j_2}^{t-1} + f_{j_1}^t}{f_i^{t+1} + f_{t-1} + f_{j_1}^t} \\ \beta_1 &= \frac{1}{2} \bullet \frac{f_{j_2}^{t-1} + f_i^{t+1}}{f_i^{t+1} + f_{j_2}^{t-1} + f_{j_1}^t} \\ \beta_2 &= \frac{1}{2} \bullet \frac{f_i^{t+1} + f_{j_1}^t}{f_i^{t+1} + f_{j_2}^{t-1} + f_{j_1}^t} \end{aligned} \tag{2.45}$$

The model in Eq. (2.44) is called $F2$ if $j_1 = j_2 = i$, and called $R2$ if j_1 and j_2 are random integers from $\{1, \ldots, n_s\}$.

- $R3$ and $F3$: These models update the i-th particle (x_i^{t+1}) using the following:

$$x_i^{t+1} = \alpha y_i^{t+1} + \beta_1 x_{j_1}^t + \beta_2 x_{j_2}^{t-1} + \beta_3 x_{j_3}^{t-2} \tag{2.46}$$

Weight parameters α, β_1, β_2, and β_3 satisfy $\alpha + \beta_1 + \beta_2 + \beta_3 = 1$ and are

formulated as:

$$\begin{aligned}
\alpha &= \frac{1}{3} \bullet \frac{f_{j_1}^{t} + f_{j_2}^{t-1} + f_{j_3}^{t-2}}{f_{i}^{t+1} + f_{j_1}^{t} + f_{j_2}^{t-1} + f_{j_3}^{t-2}} \\
\beta_1 &= \frac{1}{3} \bullet \frac{f_{i}^{t+1} + f_{j_2}^{t-1} + f_{j_3}^{t-2}}{f_{i}^{t+1} + f_{j_1}^{t} + f_{j_2}^{t-1} + f_{j_3}^{t-2}} \\
\beta_2 &= \frac{1}{3} \bullet \frac{f_{i}^{t+1} + f_{j_1}^{t} + f_{j_3}^{t-2}}{f_{i}^{t+1} + f_{j_1}^{t} + f_{j_2}^{t-1} + f_{j_3}^{t-2}} \\
\beta_3 &= \frac{1}{3} \bullet \frac{f_{i}^{t+1} + f_{j_1}^{t} + f_{j_2}^{t-1}}{f_{i}^{t+1} + f_{j_1}^{t} + f_{j_2}^{t-1} + f_{j_3}^{t-2}}
\end{aligned} \tag{2.47}$$

The model in Eq. (2.46) is called $F3$ if $j_1 = j_2 = j_3 = i$, and called $R3$ if j_1, j_2, and j_3 are random integers from $\{1, \ldots, n_s\}$.

More recently, Gu and Wang [21] incorporated the IFMs into NSGA-III [17] (another state-of-the-art reference-point-based approach for many-objective optimization proposed by Deb and Jain [17]) for large-scale many-objective optimization. In their experiments, they observed that NSGAIII-$F1$ and NSGAIII-$R1$ were the best variants[1]. For large-scale many-objective optimization using IFMs and MOEA/D [87], Zhang *et al.* [90] proposed MOEA/D-IFM.

[1]Their experiments were conducted on NSGAIII, NSGAIII-$F1$, NSGAIII-$F2$, NGSAIII-$F3$, NSGAIII-$R1$, NSGAIII-$R2$, and NSGAIII-$R3$.

Chapter 3

A Scalability Study

Particle swarm optimization has been adapted to solve multi-objective optimization problems. However, these PSO-based multi-objective optimization algorithms typically face difficulties when the number of decision variables is increased and the problems turn into large-scale multi-objective problems (LSMOPs). This chapter presents a decision space scalability study of five PSO-based algorithms, namely optimized multi-objective particle swarm optimization [69], speed-constrained multi-objective particle swarm optimization [54], multi-objective particle swarm optimization with multiple search strategies [45], multi-guide particle swarm optimization [64] [63], and competitive mechanism-based multi-objective particle swarm optimization [89] for 24, 50, 100, 500, and 1000 decision variables to see how well each one of the algorithms scales as the number of decision variables is increased. This study is a part of a bigger one on developing scalable PSO-based algorithms for large-scale MOOPs. As previously used in [30], the 24-dimensional WFG problems (with 20 distance-related parameters) were used as the starting point of these experiments. The goal is to show how some of the state-of-the-art algorithms are prone to performance deterioration when the dimensionality of the problems is increased.

3.1 Experimental Setup

This section describes the empirical process followed to study the scalability of the selected MOPSO algorithms.

3.1.1 Benchmark Suites

This chapter uses the nine functions in the Walking Fish Group set [30]. The number of position-related parameters (k as explained in Appendix A) was set to 4, 10, 20, 100, and 200 for 24, 50, 100, 500, and 1000 dimensions respectively. In this test suite a series of transition and shape functions are used to create unique POF shapes. WFG1 has a convex front, WFG2 and WFG3 have disconnected convex and linear fronts respectively, while WFG4 to WFG9 all have concave fronts. More information about the WFG test suite can be found in [31] and Appendix A.

3.1.2 Performance Measures

The following performance measures were used in this work:

- **Inverted Generational Distance**: The inverted generational distance (IGD) [12] [68] was used to measure the closeness of the obtained fronts to the true fronts. The true fronts in the jMetal framework [53] were used as the reference fronts for the 2- and 3-objective WFG1 to WFG9.
- **Hypervolume**: The hypervolume (HV) measure [96], was used to measure the space covered by the obtained fronts. The nadir point was used as the reference vector.

3.1.3 Control Parameters

The total number of particles for all algorithms was set to 50, and all of the algorithms were allocated a maximum of 10^5 function evaluations. For MGPSO, these 50 particles were split between the subswarms in the same way as [64] and listed in Table 3.3[1]. For all algorithms except CMOPSO (which does not have an external archive), the archive size was set to 50. As suggested in [89], the value of γ was set to 10 for CMOPSO and similar to [45], the value of δ was set to 0.9 for MMOPSO. For all algorithms, if the value of one dimension of the position vector became greater than the upper bound or less than the lower bound, it was assigned the upper bound and the lower bound respectively; additionally, for OMOPSO and SMPSO, the corresponding dimension of the velocity vector was multiplied by -1 and 0.001 respectively (this was mentioned in [54]). For all algorithms with external archives, the crowding distance was used as

[1]The values in this table were empirically obtained by the author [64]

a measure to remove the overcrowded solutions in the case of the archive being full upon the arrival of a new Pareto-optimal solution. As previously used in [31], the 24-dimensional WFG problems were used here. Table 3.1 and Table 3.2 provide a summary of the used OMOPSO and SMPSO parameters respectively.

Table 3.1: OMOPSO parameters

Parameter	Value
Mutation Probability	$\frac{1}{number\ of\ dimensions}$
Uniform Mutation Perturbation	0.5
Nonuniform Mutation Perturbation	0.5
η	0.0075
Leaders Archive Size	50
Archive Deletion Metric	The Crowding Distance
Swarm Size	50

Table 3.2: SMPSO parameters

Parameter	Value
Mutation Probability	$\frac{1}{number\ of\ dimensions}$
Mutation Distribution Index	20
Mutation Operator	Polynomial Mutation
Archive Size	50
Archive Deletion Metric	The Crowding Distance
Swarm Size	50

3.1.4 Statistical Methods

The algorithms were compared using the Mann-Whitney U test [51] with a confidence level of 95%. For each pair of algorithms, if the difference was deemed statistically significant, the algorithm with the better mean over the 30 independent runs was given a win and the algorithm with the worse mean was given a loss. The difference between wins and losses ($wins - losses$) was then used for ranking the algorithms.

3.2 Results

Tables 3.4 and 3.5 provide the IGD rankings for WFG1 to WFG5 and WFG6 to WFG9 respectively, whereas the HV rankings are provided in Tables 3.6 and 3.7.

Table 3.3: MGPSO parameters

Problem	Objectives	$\|S_1\|$	$\|S_2\|$	$\|S_3\|$	T	ω	c_1	c_2	c_3
WFG1	2	45	5	-	3	0.275	1.65	1.80	1.75
WFG2	2	24	26	-	2	0.750	1.15	1.70	1.05
WFG3	2	31	19	-	2	0.600	1.60	1.85	0.95
WFG4	2	2	48	-	2	0.100	0.80	1.65	1.70
WFG5	2	50	0	-	2	0.600	0.80	1.60	1.85
WFG6	2	19	31	-	2	0.525	0.65	0.60	1.65
WFG7	2	29	21	-	2	0.450	1.20	1.85	1.55
WFG8	2	37	13	-	3	0.750	1.00	1.65	1.05
WFG9	2	13	37	-	2	0.275	1.00	0.50	1.70
WFG1	3	37	4	9	2	0.125	1.20	1.30	1.75
WFG2	3	24	25	1	2	0.275	1.25	1.40	1.70
WFG3	3	29	10	11	2	0.525	1.65	1.75	0.75
WFG4	3	29	21	0	2	0.275	1.75	0.50	1.05
WFG5	3	2	48	0	3	0.575	0.60	1.85	1.75
WFG6	3	5	30	15	3	0.300	0.90	0.90	1.90
WFG7	3	10	22	18	2	0.425	1.45	1.50	1.40
WFG8	3	4	23	23	3	0.425	0.95	1.75	1.85
WFG9	3	4	45	1	2	0.275	1.25	0.75	1.50

Fig. 3.1 depicts a summary of the overall IGD and HV rankings.

3.2.1 24 Dimensions

MGPSO had the best IGD performance on the 2-objective WFG2, WFG3, WFG6, and WFG7. MGPSO also had the best overall aggregated IGD scores (Table 3.5) for the 2- and 3-objective functions, followed by SMPSO and CMOPSO respectively. In terms of HV (Tables 3.6 and 3.7), MMOPSO had the best performance on the 2-objective functions followed by MGPSO; however, MGPSO managed to beat MMOPSO to become the best-performer on the 3-objective functions. SMPSO had the best IGD and HV performance on the 2-objective WFG9, obtaining the best possible score (+4). Moreover, MMOPSO obtained the best possible IGD and HV scores for the 2- and 3-objective WFG1. Despite MGPSO's good overall performance

Table 3.4: IGD rankings for WFG1 to WFG5

		Algorithm (2-Objective)						Algorithm (3-Objective)					
Problem	Dimension	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*
WFG1	24	Difference	−2	−4	4	2	0	Difference	−4	−2	4	0	2
		Rank	4	5	**1**	2	3	Rank	5	4	**1**	3	2
	50	Difference	−2	−4	4	1	1	Difference	−3	−3	4	1	1
		Rank	3	4	**1**	2	2	Rank	3	3	**1**	2	2
	100	Difference	−3	1	4	1	−3	Difference	−1	−1	4	−4	2
		Rank	3	2	**1**	2	3	Rank	3	3	**1**	4	2
	500	Difference	0	4	2	−2	−4	Difference	−2	1	4	−4	1
		Rank	3	**1**	2	4	5	Rank	3	2	**1**	4	2
	1000	Difference	−1	2	4	−1	−4	Difference	−2	1	4	−4	1
		Rank	3	2	**1**	3	4	Rank	3	2	**1**	4	2
WFG2	24	Difference	2	−1	−1	4	−4	Difference	−1	−1	−4	4	2
		Rank	2	3	3	**1**	4	Rank	3	3	4	**1**	2
	50	Difference	2	−1	0	3	−4	Difference	3	−1	−3	−2	3
		Rank	2	4	3	**1**	5	Rank	**1**	2	4	3	**1**
	100	Difference	2	−2	2	2	−4	Difference	4	1	1	−4	−2
		Rank	**1**	2	**1**	**1**	3	Rank	**1**	2	2	4	3
	500	Difference	3	−2	2	−1	−2	Difference	3	−1	3	−4	−1
		Rank	**1**	4	2	3	4	Rank	**1**	2	**1**	3	2
	1000	Difference	2	2	−2	−1	−1	Difference	2	2	2	−3	−3
		Rank	**1**	**1**	3	2	2	Rank	**1**	**1**	**1**	2	2
WFG3	24	Difference	0	0	−4	4	0	Difference	−1	−4	2	4	−1
		Rank	2	2	3	**1**	2	Rank	3	4	2	**1**	3
	50	Difference	−1	−1	2	−4	4	Difference	0	−2	2	4	−4
		Rank	3	3	2	4	**1**	Rank	3	4	2	**1**	5
	100	Difference	0	−2	3	−4	3	Difference	−1	3	−1	3	−4
		Rank	2	3	**1**	4	**1**	Rank	2	**1**	2	**1**	3
	500	Difference	4	−2	1	−4	1	Difference	1	1	4	−2	−4
		Rank	**1**	3	2	4	2	Rank	2	2	**1**	3	4
	1000	Difference	2	−4	−1	−1	4	Difference	2	−2	4	0	−4
		Rank	2	4	3	3	**1**	Rank	2	4	**1**	3	5
WFG4	24	Difference	1	1	4	−4	−2	Difference	2	−2	−2	4	−2
		Rank	2	2	**1**	4	3	Rank	2	3	3	**1**	3
	50	Difference	0	2	4	−4	−2	Difference	2	2	2	−3	−3
		Rank	3	2	**1**	5	4	Rank	**1**	**1**	**1**	2	2
	100	Difference	0	3	3	−4	−2	Difference	0	4	2	−4	−2
		Rank	2	**1**	**1**	4	3	Rank	3	**1**	2	5	4
	500	Difference	0	2	4	−4	−2	Difference	0	4	2	−4	−2
		Rank	3	2	**1**	5	4	Rank	3	**1**	2	5	4
	1000	Difference	0	2	4	−4	−2	Difference	0	4	2	−4	−2
		Rank	3	2	**1**	5	4	Rank	3	**1**	2	5	4
WFG5	24	Difference	1	4	−2	1	−4	Difference	−2	−2	−2	4	2
		Rank	2	**1**	3	2	4	Rank	3	3	3	**1**	2
	50	Difference	3	3	0	−4	−2	Difference	0	−4	4	1	−1
		Rank	**1**	**1**	2	4	3	Rank	3	5	**1**	2	4
	100	Difference	2	−1	4	−4	−1	Difference	0	−3	4	−3	2
		Rank	2	3	**1**	4	3	Rank	3	4	**1**	4	2
	500	Difference	0	−2	4	−4	2	Difference	0	−2	4	−4	2
		Rank	3	4	**1**	5	2	Rank	3	4	**1**	5	2
	1000	Difference	0	−2	4	−4	2	Difference	0	0	4	−4	0
		Rank	3	4	**1**	5	2	Rank	2	2	**1**	3	2

on the 24-dimensional problems, it had the worst IGD and HV performance on the 2-objective WFG4 and the 2-objective WFG8.

3.2.2 50 Dimensions

MGPSO's IGD and HV performances dropped noticeably compared with 24 dimensions. MGPSO ranked last and third overall in IGD for the 2- and 3-objective functions respectively, having had the best overall IGD scores for the 24-dimensional problems. The transition from 24 to 50 dimensions particularly turned MGPSO into the worst IGD and HV performer on the 2-objective WFG6 and WFG7, having had the best performances on the same functions for 24 dimensions. On the contrary, MMOPSO rose to the top as the problems were scaled up to become the best IGD

Table 3.5: IGD rankings for WFG6 to WFG9

		Algorithm (2-Objective)						Algorithm (3-Objective)					
Problem	Dimension	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*
WFG6	24	Difference	−4	2	−2	3	1	Difference	−4	0	−2	3	3
		Rank	5	2	4	**1**	3	Rank	4	2	3	**1**	**1**
	50	Difference	1	4	−2	−4	1	Difference	1	4	1	−3	−3
		Rank	2	**1**	3	4	2	Rank	2	**1**	2	3	3
	100	Difference	2	4	0	−4	−2	Difference	1	4	1	−4	−2
		Rank	2	**1**	3	5	4	Rank	2	**1**	2	4	3
	500	Difference	4	2	0	−4	−2	Difference	2	4	−1	−4	−1
		Rank	**1**	2	3	5	4	Rank	2	**1**	3	4	3
	1000	Difference	2	4	0	−4	−2	Difference	3	3	0	−3	−3
		Rank	2	**1**	3	5	4	Rank	**1**	**1**	2	3	3
WFG7	24	Difference	2	0	−4	4	−2	Difference	0	−3	−3	3	3
		Rank	2	3	5	**1**	4	Rank	2	3	3	**1**	**1**
	50	Difference	4	−3	2	−3	0	Difference	1	−2	3	2	−4
		Rank	**1**	4	2	4	3	Rank	3	4	**1**	2	5
	100	Difference	2	−2	4	−4	0	Difference	−1	2	4	−1	−4
		Rank	2	4	**1**	5	3	Rank	3	2	**1**	3	4
	500	Difference	−1	4	0	−4	1	Difference	2	2	2	−4	−2
		Rank	4	**1**	3	5	2	Rank	**1**	**1**	**1**	3	2
	1000	Difference	−4	1	−2	1	4	Difference	0	3	3	−3	−3
		Rank	4	2	3	2	**1**	Rank	2	**1**	**1**	3	3
WFG8	24	Difference	4	0	0	−4	0	Difference	0	−4	0	0	4
		Rank	**1**	2	2	3	2	Rank	2	3	2	2	**1**
	50	Difference	−2	2	2	−4	2	Difference	2	−4	4	−1	−1
		Rank	2	**1**	**1**	3	**1**	Rank	2	4	**1**	3	3
	100	Difference	0	−2	2	−4	4	Difference	2	2	2	−3	−3
		Rank	3	4	2	5	**1**	Rank	**1**	**1**	**1**	2	2
	500	Difference	0	−4	3	−2	3	Difference	3	3	0	−4	−2
		Rank	2	4	**1**	3	**1**	Rank	**1**	**1**	2	4	3
	1000	Difference	−1	−4	−1	4	2	Difference	4	2	0	−4	−2
		Rank	3	4	3	**1**	2	Rank	**1**	2	3	5	4
WFG9	24	Difference	−3	4	−2	0	1	Difference	−2	0	−4	2	4
		Rank	5	**1**	4	3	2	Rank	4	3	5	2	**1**
	50	Difference	0	4	−2	−4	2	Difference	−2	−1	0	−1	4
		Rank	3	**1**	4	5	2	Rank	4	3	2	3	**1**
	100	Difference	0	4	2	−4	−2	Difference	−1	3	3	−4	−1
		Rank	3	**1**	2	5	4	Rank	2	**1**	**1**	3	2
	500	Difference	1	4	1	−4	−2	Difference	1	4	1	−2	−4
		Rank	2	**1**	2	4	3	Rank	2	**1**	2	3	4
	1000	Difference	2	4	0	−2	−4	Difference	2	4	0	−2	−4
		Rank	2	**1**	3	4	5	Rank	2	**1**	3	4	5
Overall (WFG1-9)	24	Difference	1	6	−7	10	−10	Difference	−12	−18	−11	24	17
		Rank	3	2	4	**1**	5	Rank	4	5	3	**1**	2
	50	Difference	5	6	10	−23	2	Difference	4	−11	17	−2	−8
		Rank	3	2	**1**	5	4	Rank	2	5	**1**	3	4
	100	Difference	5	3	24	−25	−7	Difference	3	15	20	−24	−14
		Rank	2	3	**1**	5	4	Rank	3	2	**1**	5	4
	500	Difference	11	6	17	−29	−5	Difference	10	16	19	−32	−13
		Rank	2	3	**1**	5	4	Rank	3	2	**1**	5	4
	1000	Difference	2	5	6	−12	−1	Difference	11	17	19	−27	−20
		Rank	3	2	**1**	5	4	Rank	3	2	**1**	5	4

performer on the 2- and 3-objective functions, having finished fourth and third in the 24-dimensional 2- and 3-objective problems respectively. CMOPSO's IGD performance on the 3-objective problems also dropped compared with 24 dimensions. MMOPSO was also the best HV performer on the 2- and 3-objective functions, taking over MGPSO's spot for the latter. Moreover, SMPSO continued its IGD and HV dominance for the 2-objective WFG9 and despite CMPSO's relatively poor IGD performance, it still managed to rank first in IGD for the 2-objective WFG8, together with SMPSO and MMOPSO.

Table 3.6: HV rankings for WFG1 to WFG5

		Algorithm (2-Objective)						Algorithm (3-Objective)					
Problem	Dimension	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*
WFG1	24	Difference	−2	−4	4	2	0	Difference	−4	−2	4	0	2
		Rank	4	5	**1**	2	3	Rank	5	4	**1**	3	2
	50	Difference	−2	−4	4	1	1	Difference	−4	−2	4	0	2
		Rank	3	4	**1**	2	2	Rank	5	4	**1**	3	2
	100	Difference	−2	1	4	1	−4	Difference	−2	1	4	−4	1
		Rank	3	2	**1**	2	4	Rank	3	2	**1**	4	2
	500	Difference	0	2	4	−2	−4	Difference	0	2	4	−4	−2
		Rank	3	2	**1**	4	5	Rank	3	2	**1**	5	4
	1000	Difference	−1	2	4	−1	−4	Difference	0	2	4	−4	−2
		Rank	3	2	**1**	3	4	Rank	3	2	**1**	5	4
WFG2	24	Difference	2	0	−2	4	−4	Difference	2	−4	−1	−1	4
		Rank	2	3	4	**1**	5	Rank	2	4	3	3	**1**
	50	Difference	2	−2	1	3	−4	Difference	4	0	1	−4	−1
		Rank	2	4	3	**1**	5	Rank	**1**	3	2	5	4
	100	Difference	2	−2	2	2	−4	Difference	3	1	2	−4	−2
		Rank	**1**	2	**1**	**1**	3	Rank	**1**	3	2	5	4
	500	Difference	2	2	2	−3	−3	Difference	2	2	2	−3	−3
		Rank	**1**	**1**	**1**	2	2	Rank	**1**	**1**	**1**	2	2
	1000	Difference	1	4	−2	−2	−1	Difference	2	4	0	−3	−3
		Rank	2	**1**	4	4	3	Rank	2	**1**	3	4	4
WFG3	24	Difference	−2	−3	2	4	−1	Difference	0	−2	2	4	−4
		Rank	4	5	2	**1**	3	Rank	3	4	2	**1**	5
	50	Difference	−1	−1	2	−4	4	Difference	0	−2	3	3	−4
		Rank	3	3	2	4	**1**	Rank	2	3	**1**	**1**	4
	100	Difference	1	−2	4	−4	1	Difference	−2	3	2	1	−4
		Rank	2	3	**1**	4	2	Rank	4	**1**	2	3	5
	500	Difference	3	−4	0	−2	3	Difference	0	−2	3	3	−4
		Rank	**1**	4	2	3	**1**	Rank	2	3	**1**	**1**	4
	1000	Difference	1	−4	−2	4	1	Difference	2	−2	0	4	−4
		Rank	2	4	3	**1**	2	Rank	2	4	3	**1**	5
WFG4	24	Difference	0	−1	4	−4	1	Difference	−2	1	4	1	−4
		Rank	3	4	**1**	5	2	Rank	3	2	**1**	2	4
	50	Difference	0	2	4	−4	−2	Difference	0	3	3	−4	−2
		Rank	3	2	**1**	5	4	Rank	2	**1**	**1**	4	3
	100	Difference	0	2	4	−4	−2	Difference	0	4	2	−4	−2
		Rank	3	2	**1**	5	4	Rank	3	**1**	2	5	4
	500	Difference	0	2	4	−4	−2	Difference	0	4	2	−4	−2
		Rank	3	2	**1**	5	4	Rank	3	**1**	2	5	4
	1000	Difference	0	2	4	−4	−2	Difference	1	4	1	−4	−2
		Rank	3	2	**1**	5	4	Rank	2	**1**	2	4	3
WFG5	24	Difference	0	2	4	−4	−2	Difference	−1	−1	4	2	−4
		Rank	3	2	**1**	5	4	Rank	3	3	**1**	2	4
	50	Difference	2	−1	4	−4	−1	Difference	2	−1	4	−1	−4
		Rank	2	3	**1**	4	3	Rank	2	3	**1**	3	4
	100	Difference	2	−2	4	−4	0	Difference	2	−2	4	−4	0
		Rank	2	4	**1**	5	3	Rank	2	4	**1**	5	3
	500	Difference	2	0	4	−4	−2	Difference	2	−1	4	−4	−1
		Rank	2	3	**1**	5	4	Rank	2	3	**1**	4	3
	1000	Difference	2	0	4	−4	−2	Difference	1	1	4	−4	−2
		Rank	2	3	**1**	5	4	Rank	2	2	**1**	4	3

3.2.3 100 Dimensions

MMOPSO was once again the best overall (Table 3.4) IGD performer on the 2- and 3-objective functions, achieving a noticeably better overall score than its competitors for two objectives. MMOPSO took OMOPSO and SMPSO's spot as the best IGD performer on the 2-objective WFG5. Despite CMOPSO being the best IGD performer on the 2-objective WFG8, MGPSO and CMOPSO were the worst overall IGD performers on the 2- and 3-objective functions, having obtained notably worse overall scores than their counterparts. At 100 dimensions, MMOPSO managed to achieve the best IGD performance on the 2-objective WFG7 and WFG5 for the first time since 24 dimensions. Despite poor overall scores, CMOPSO and MGPSO had the best IGD performances on the 2- and 3-objective WFG3, respectively. A similar

Table 3.7: HV rankings for WFG6 to WFG9

		Algorithm (2-Objective)						Algorithm (3-Objective)					
Problem	Dimension	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*	Result	*OMOPSO*	*SMPSO*	*MMOPSO*	*MGPSO*	*CMOPSO*
WFG6	24	Difference	−3	2	−3	2	2	Difference	−4	2	−2	4	0
		Rank	2	**1**	2	**1**	**1**	Rank	5	2	4	**1**	3
	50	Difference	3	3	−1	−4	−1	Difference	0	4	2	−3	−3
		Rank	**1**	**1**	2	3	2	Rank	3	**1**	2	4	4
	100	Difference	3	3	0	−4	−2	Difference	2	4	0	−3	−3
		Rank	**1**	**1**	2	4	3	Rank	2	**1**	3	4	4
	500	Difference	2	4	0	−2	−4	Difference	2	4	−2	0	−4
		Rank	2	**1**	3	4	5	Rank	2	**1**	4	3	5
	1000	Difference	2	4	0	−2	−4	Difference	2	4	−2	0	−4
		Rank	2	**1**	3	4	5	Rank	2	**1**	4	3	5
WFG7	24	Difference	2	−2	0	4	−4	Difference	0	−4	2	4	−2
		Rank	2	4	3	**1**	5	Rank	3	5	2	**1**	4
	50	Difference	2	−2	4	−4	0	Difference	1	−2	4	1	−4
		Rank	2	4	**1**	5	3	Rank	2	3	**1**	2	4
	100	Difference	2	−2	4	−4	0	Difference	0	2	4	−2	−4
		Rank	2	4	**1**	5	3	Rank	3	2	**1**	4	5
	500	Difference	−2	3	−3	−1	3	Difference	3	3	0	−3	−3
		Rank	3	**1**	4	2	**1**	Rank	**1**	**1**	2	3	3
	1000	Difference	−3	2	−3	4	0	Difference	2	4	0	−2	−4
		Rank	4	2	4	**1**	3	Rank	2	**1**	3	4	5
WFG8	24	Difference	4	0	0	−4	0	Difference	0	−4	4	2	−2
		Rank	**1**	2	2	3	2	Rank	3	5	**1**	2	4
	50	Difference	−1	1	3	−4	1	Difference	1	−2	4	1	−4
		Rank	3	2	**1**	4	2	Rank	2	3	**1**	2	4
	100	Difference	0	−2	4	−4	2	Difference	1	1	4	−2	−4
		Rank	3	4	**1**	5	2	Rank	2	2	**1**	3	4
	500	Difference	−2	−4	2	0	4	Difference	3	3	0	−2	−4
		Rank	4	5	2	3	**1**	Rank	**1**	**1**	2	3	4
	1000	Difference	−1	−4	−1	4	2	Difference	4	2	0	−2	−4
		Rank	3	4	3	**1**	2	Rank	**1**	2	3	4	5
WFG9	24	Difference	−3	4	−3	0	2	Difference	−4	2	−1	2	1
		Rank	4	**1**	4	3	2	Rank	4	1	3	**1**	2
	50	Difference	−1	4	2	−4	−1	Difference	−1	0	4	−4	1
		Rank	3	**1**	2	4	3	Rank	4	3	**1**	5	2
	100	Difference	0	3	3	−4	−2	Difference	0	2	4	−4	−2
		Rank	2	**1**	**1**	4	3	Rank	3	2	**1**	5	4
	500	Difference	2	4	−1	−1	−4	Difference	2	4	0	−2	−4
		Rank	2	**1**	3	3	4	Rank	2	**1**	3	4	5
	1000	Difference	2	4	−2	0	−4	Difference	2	4	0	−2	−4
		Rank	2	**1**	4	3	5	Rank	2	**1**	3	4	5
Overall (WFG1-9)	24	Difference	−2	−2	6	4	−6	Difference	−13	−12	16	18	−9
		Rank	3	3	**1**	2	4	Rank	5	4	2	**1**	3
	50	Difference	4	0	23	−24	−3	Difference	3	−2	29	−11	−19
		Rank	2	3	**1**	5	4	Rank	2	3	**1**	4	5
	100	Difference	8	−1	29	−25	−11	Difference	4	16	26	−26	−20
		Rank	2	3	**1**	5	4	Rank	3	2	**1**	5	4
	500	Difference	7	9	12	−19	−9	Difference	14	19	13	−19	−27
		Rank	3	2	**1**	5	4	Rank	2	**1**	3	4	5
	1000	Difference	3	10	2	−1	−14	Difference	16	23	7	−17	−29
		Rank	2	**1**	3	4	5	Rank	2	**1**	3	4	5

pattern to IGD was observed for HV, where MMOPSO finished first and MGPSO and CMOPSO were the worst performers. SMPSO managed to continue its HV dominance for the 2-objective WFG9, finishing first together with MMOPSO. Interestingly, MMOPSO managed to outperform SMPSO on the 3-objective WFG9.

3.2.4 500 Dimensions

With reference to IGD, MMOPSO once again had the best overall scores (Table 3.5) for the 2- and 3-objective functions. However, this time the margin was closer compared with 50 dimensions, with OMOPSO and SMPSO as the second best algorithms for the 2- and 3-objective functions respectively. MGPSO had the worst overall scores (Table 3.5) for the 2- and 3-objective functions by noticeable margins, followed by

CMOPSO as the second worst. In fact, MGPSO had the worst IGD performance on 13 out of the 18 500-dimensional functions. On the 2-objective WFG8, CMOPSO once again had the best performance together with MMOPSO. With reference to HV, although MMOPSO had the best overall score (Table 3.7) for the 2-objective functions, its spot was overtaken by SMPSO and OMOPSO as the best and the second best performing algorithms on the 3-objective functions respectively. Particularly, for the 3-objective functions, the transition from 100 to 500 dimensions made MMOPSO lose its number one spot (HV) for WFG7, WFG8 and WFG9 to SMPSO. MGPSO and CMOPSO once again had the worst overall HV scores for the 2- and 3-objective functions respectively. CMOPSO, after showing positive signs of IGD scalability on the 2-objective WFG8 (see Table 3.5), had the best HV performance on the 2-objective 500-dimensional WFG8, taking zero losses and winning against all other algorithms.

3.2.5 1000 Dimensions

With reference to IGD, MMOPSO had the best overall scores for the 2- and 3-objective functions (Table 3.5) followed by SMPSO, although the margin was much closer for 1000 dimensions compared with 500 or 100 dimensions. For the 2-objective WFG7, CMOPSO gained the best possible score (+4), becoming the best IGD performer on this function. Despite MGPSO's poor overall scalability, it had the best IGD performance on the 2-objective WFG8. With reference to HV, SMPSO had the best overall scores (Table 3.7) for the 2- and 3-objective functions, followed by OMOPSO and MMOPSO. SMPSO continued its good scalability for the 2- and 3-objective WFG9, WFG2, WFG6 and the 3-objective WFG4. Moreover, MMOPSO was once again the algorithm with the best HV scalability on the 2- and 3-objective WFG5 and the 2-objective WFG4. Similar to the IGD measure (Table 3.5), MGPSO had the best HV performance on the 1000-dimensional 2-objective WFG8 despite its overall poor performance. Additionally, MGPSO had the best HV performance on the 2- and 3-objective WFG3 and the 2-objective WFG7.

3.2.6 Summary

A summary of the overall IGD and HV rankings of the selected algorithms for different dimensions based on the aggregated scores is depicted in Fig. 3.1. With reference to IGD, MMOPSO had the best overall performance for dimensions greater than 50 on the 2- and 3-objective functions. With reference to HV, SMPSO had the best overall performance on the 500-dimensional 3-objective functions and the

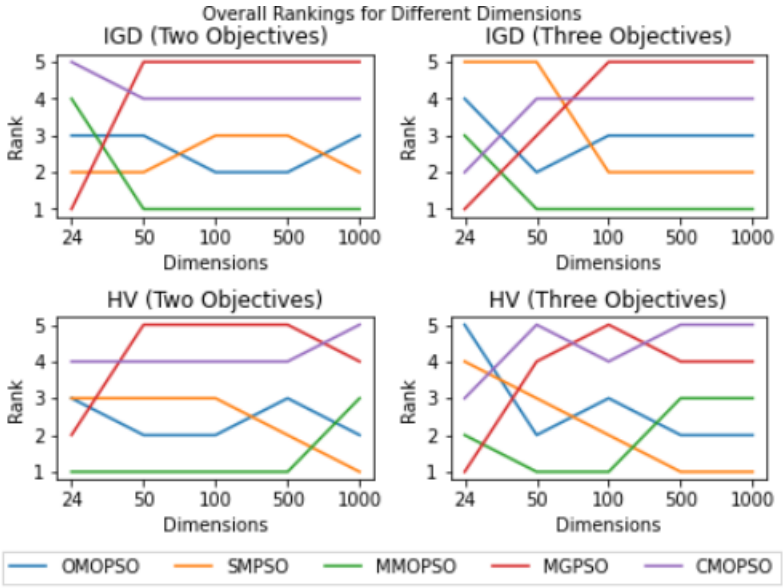

Figure 3.1: Overall rankings of the algorithms as a function of the number of dimensions with reference to IGD (upper left and upper right for two and three objectives respectively) and HV (lower left and lower right for two and three objectives respectively).

1000-dimensional functions. MMOPSO had the best overall HV scalability for up to 500 dimensions (two objectives) and 100 dimensions (three objectives); then, it was outperformed by SMPSO and OMOPSO as the best and second best performing algorithms respectively. Except for the 500-dimensional 2-objective functions where OMOPSO had the better overall IGD performance, SMPSO outperformed OMOPSO in both HV and IGD for dimensions greater than 500. As seen in Fig. 3.1, MGPSO and CMOPSO were the worst-performing algorithms for dimensions greater than 100. MGPSO had the best IGD performance on the 24-dimensional functions, the best and the second best HV performances on the 24-dimensional 3 and 2-objective functions respectively. However, it showed poor scalability, constantly obtaining the worst or the second worst (behind CMOPSO) overall IGD and HV scores from 50 to 1000 dimensions.

Regarding individual problems, MMOPSO was a regular top IGD and HV performer on the 2- and 3-objective WFG1, the 2-objective WFG4 and the 2- and 3-objective WFG5, as well as earning some number one rankings in WFG7, WFG2 and WFG9. SMPSO had the best IGD and HV scalability for the 2- and 3-objective WFG6 and WFG9. In spite of its good overall scalability, MMOPSO failed to register a single number one ranking in IGD or HV for the 2- and 3-objective WFG6. WFG6

was MMOPSO's most significant weak point, where it was constantly outperformed by SMPSO and sometimes OMOPSO and even MGPSO. OMOPSO in particular showed scalable IGD and HV performances on the 2- and 3-objective WFG2 and the 3-objective WFG8, constantly ranking as the best performing algorithm. In terms of the overall aggregated HV scores, MMOPSO was outperformed by OMOPSO and SMPSO on the 500-dimensional 3-objective and 1000-dimensional functions. This was mostly a result of MMOPSO's HV performance drop on the 3-objective WFG7 to WFG9 (from 100 to 1000 dimensions) and the 2- and 3-objective WFG2 (from 500 to 1000 dimensions). OMOPSO showed good signs of IGD and HV scalability on the 2- and 3-objective WFG2 and the 3-objective WFG8, but it did not dominate any functions as significantly as MMOPSO and SMPSO did with WFG5 and WFG9 respectively. MGPSO showed poor scalability as the dimensionality of the problems was increased. MGPSO particularly struggled with the 2- and 3-objective WFG4 to WFG6 and the 3-objective WFG1. There were some positive signs of MGPSO scalability, such as the best IGD and HV performances on the 1000-dimensional 2-objective WFG8, the best HV performance on the 2-objective WFG7 and the best HV performance on the 1000-dimensional 2- and 3-objective WFG3. However, MGPSO's overall performance on the high-dimensional problems was not on par with its performance on the low-dimensional ones. Similar to MGPSO, CMOPSO only showed minor signs of scalability. Particularly, CMOPSO had the best scalability on the 2-objective WFG8, achieving the best IGD scores for 50, 100 and 500 dimensions and the best HV score for 500 dimensions.

3.3 Conclusion & Future Work

This chapter presented a decision space scalability study of five PSO-based algorithms for multi-objective optimization, namely optimized multi-objective particle swarm optimization (OMOPSO), speed-constrained multi-objective particle swarm optimization (SMPSO), multi-objective particle swarm optimization with multiple search strategies (MMOPSO), multi-guide particle swarm optimization (MGPSO), and competitive mechanism-based multi-objective particle swarm optimization (CMOPSO) for 24, 50, 100, 500 and 1000 dimensions (decision variables). These algorithms were tested on 18 problems (the 2- and 3-objective WFG1 to WFG9) from the Walking Fish Group (WFG) problems. Inverted generational distance (IGD) and hypervolume (HV) were used as the measures to compare the algorithms. The results showed

that, despite a few exceptions involving WFG3, WFG7, and WFG8, MGPSO and CMOPSO showed the worst scalability despite the fact that MGPSO showed competitive performance on the low-dimensional problems. MMOPSO showed the best overall IGD scalability (up to 1000 dimensions) and the best overall HV scalability (for up to 500 dimensions), showing dominant performances on WFG1, WFG5 and the 2-objective WFG4. SMPSO and OMOPSO both showed scalability on different functions, with SMPSO having the more significant highlights (dominating WFG6, WFG9 and the 3-objective WFG4). Additionally, SMPSO had the best overall HV performance on the 1000-dimensional and the 500-dimensional 3-objective functions. In the following chapters, a new MGPSO-based algorithm is proposed to address the shortcomings of MGPSO in dealing with large-scale problems as discussed in this study.

Chapter 4

A New Scalable MGPSO-Based Approach

This chapter provides the details of a proposed improved MGPSO algorithm that incorporates ideas from cooperative PSOs, termed cooperative co-evolutionary multi-guide particle swarm optimization (CCMGPSO).

4.1 Introduction

The scalability study in Chapter 3 showed that the performance of MGPSO quickly drops with the increase in the number of decision variables. Therefore, this chapter proposes a new scalable multi-objective optimization algorithm based on MGPSO, termed cooperative co-evolutionary multi-guide particle swarm optimization (CCMGPSO).

A motivation behind CCMGPSO is to efficiently tackle large-scale multi-objective optimization problems, while preserving computational budget. Since a lot of approaches based on the CC framework use k dimension groups (and $\frac{n_x}{k}$-dimensional individuals) for optimizing the objective value, an important task when incorporating cooperative approaches into multi-objective optimization is determining an efficient way of cooperatively optimizing the objective values. Using an independent CPSO per objective could increase the total number of individuals and consequently the computational cost of the algorithm. Therefore, the optimization of the problem would become impractical under such circumstances. For example, in a cooperative MOO approach where each objective is optimized separately and the decision variables are divided into 200 groups with 200 individuals, there would be a need for 400 and 600 individuals for two and three objectives respectively. The proposed CCMG-

PSO, which is discussed in greater detail in the following sections, aims to share a single CPSO between different objectives in cycles of equal length.

4.2 The Proposed Approach

The proposed CCMGPSO adds a single CPSO on top of the MGPSO, meaning that the CCMGPSO consists of n_x- and $\frac{n_x}{k}$-dimensional individuals, inspired by the CPSO-H_k [77]. The pseudo-code for CCMGPSO is provided in Algorithm 14. At the beginning of the algorithm, CCMGPSO randomly initializes n_c context vectors in the decision space. For each context vector, the objective vector is also kept track of to save computational budget (the set of context vector objective values is denoted by $\widehat{C}$ in Algorithm 14). Every run of the n_x-dimensional subswarms is followed by a single run of the CPSO (the $\frac{n_x}{k}$-dimensional subswarms). These $\frac{n_x}{k}$-dimensional individuals optimize different objectives depending on the iteration number. This is controlled through a control parameter named iterations per objective (γ) (Line 21 in Algorithm 14). For example, for a bi-objective optimization problem with $\gamma = 10$, in the first 10 iterations every run of the n_x-dimensional subswarms is followed by a run of the $\frac{n_x}{k}$-dimensional subswarms on the first objective. For the next 10 iterations, every run of the n_x-dimensional subswarms is followed by a run of the $\frac{n_x}{k}$-dimensional subswarms on the second objective. When all objectives have had one cooperative cycle (γ iterations) and after $n_m \times \gamma$ iterations, the decision variables are regrouped and the particles are randomly reinitialized according to the new dimension indices (lines 14 to 16 in Algorithm 14).

It is worth mentioning that when the CPSO is switching[1] objectives, the personal best values of the $\frac{n_x}{k}$-dimensional particles are reset (set to $+\infty$) (line 18 in Algorithm 14). Moreover, before the CPSO starts optimizing a specific objective, it is randomly assigned a context vector from the pool of context vectors (line 19 in Algorithm 14). The CPSO then holds onto this context vectors for the next γ iterations, and it will update this context vector based on improvements in the assigned objective (Algorithm 16).

[1]Here, "switching" means when a specific objective has had its allowed amount of time (γ iterations) and the next objective is chosen.

4.2.1 Velocity and Position Update Equations

The CCMGPSO also utilizes the original MGPSO velocity and position update equations. For the particles representing a smaller subset of dimensions, the global best is extracted from the assigned context vector (C_l in Algorithm 14 and Algorithm 16) based on the corresponding decision variables. The archive guide for these particles is also selected from the archive using tournament selection and trimmed into the correct decision variables. An example of the archive guide implementation for the $\frac{n_x}{k}$-dimensional particles is depicted in Figure 4.1 through an arbitrary 9-dimensional problem whose decision variables are classified into three different groups.

Many PSO-based approaches for multi-objective optimization (such as SMPSO [54] and OMOPSO [69]) use random values for parameters such as c_1, c_2, and ω. Similarly, and contrary to MGPSO, in CCMGPSO the values of c_1, c_2, c_3 are randomly selected from $[1.5, 2]$ and the value of ω is sampled from $[0.1, 0.5]$. This randomization is done for every particle at every iteration. This was done to deal with the difficulties associated with tuning the CCMGPSO parameters on every test instance. The aforementioned ranges for c_1, c_2 and ω have been successfully used in other MOPSO-based approaches, such as OMOPSO [69] and MMOPSO [45]. Therefore, for c_3 in CCMGPSO the same range of possible random values was used.

Note that in CCMGPSO the particles' position vectors are randomly re-initialized after each decision variable regrouping (lines 14 to 16 in Algorithm 14). The reasoning behind this is that in some large-scale problems (such as DTLZ6), the values of some decision variables tend to get stuck at specific values. This means that if the particles representing specific decision variables fail to update the corresponding dimensions in the context vectors, they will gradually be driven towards the same values because the context vectors are also used as the global best guides for the $\frac{n_x}{k}$-dimensional particles. Moreover, because a lot of solutions are added to the archive on the basis of small modifications to the context vectors (line 19 in Algorithm 16), in these situations the archive could get replete with solutions that have the same values for the aforementioned decision variables. Therefore, there may be cases where the current position, the personal best position, the global best position, and the archive guide position vectors all have the exact same or very similar values in some dimensions, making the velocity smaller and smaller over time causing premature convergence in those specific decision variables. Hence, the random re-initialization of the position vectors generally showed effectiveness in helping the particles avoid these situations and was used in CCMGPSO.

Algorithm 14 The pseudo-code of CCMGPSO

procedure CCMGPSO
  Input: n_x (number of decision variables), n_m (number of objectives), n_A (archive size), k (the number of decision variable groups), n_c (the number of context vectors), γ (the number of CPSO iterations per objective), f (the optimization problem)
  Output: non-dominated front (F).
  Randomly initialize n_c context vectors in the decision space in a set called C;
  $P \leftarrow$ Initialize n_m subswarms each having n_x decision variables;
  Initialize a crowding distance-based archive of size n_A;
  $l \leftarrow$ a randomly selected context vector index from the set C;
  For each context vector $C_{l'}$, $1 \leq l' \leq n_c$, initialize an objective vector $\widehat{C}_{l'}$ of size n_m, set all objective values of $\widehat{C}_{l'}$ to $+\infty$
  for $iteration = 0 : maxIterations - 1$ **do**
    for each objective $i = 1 : n_m$ **do**
      Randomly select a particle from $P[i]$, replace it with C_l;
      Optimize $P[i]$ using MGPSO;
    end for
    if $iteration\%(n_m \times \gamma) = 0$ **then**
      $S \leftarrow$ Randomly group the decision variables into k groups, initialize subswarms with $\frac{n_x}{k}$ decision variables;
    end if
    if $iteration\%\gamma = 0$ **then**
      For all particles in the subswarms of S, set the $pBest$ value to $+\infty$;
      $l \leftarrow$ a randomly selected context vector index from the set C;
    end if
    $o \leftarrow (\lfloor \frac{iteration}{\gamma} \rfloor \% n_m) + 1$; ▷ o is the objective number that the CPSO is currently optimizing
    Optimize o using C_l and the subswarms in S, update C_l if needed;
  end for
end procedure

Algorithm 15 The modified context vector replacement of the CCMGPSO

procedure B
  Input: C (set of context vector decision vectors), l (the current context vector index),j (the subswarm index), z (an arbitrary $\frac{n_x}{k}$-dimensional vector belonging to the j-th subswarm;
  Return $(C_{l_1}, C_{l_2}, \ldots, C_{l_{j-1}}, z, C_{l_{j+1}}, \ldots, C_{l_k})$
end procedure

Algorithm 16 The cooperative part of the CCMGPSO

1: **procedure** COOPERATIVE
2: **Input**: C (the set of context vector decision vectors), $\widehat{C}$ (the set of context vector objective vectors), l (the current context vector index), o (the objective that is currently being optimized by the CPSO), S (the set of the $\frac{n_x}{k}$-dimensional subswarms), f (the optimization problem)
3: **for** each sub-swarm $j = 1, \dots k$ **do**
4: **for** each particle $i = 1, \dots, n_s$ **do**
5: $X \leftarrow f(\mathbf{B}(C, l, j, S_j.x_i))$
6: **if** $X_o < S_j.pBest_i$ **then** ▷ X_o is the o-th objective value of the objective vector X
7: $S_j.\mathbf{y}_i \leftarrow P_j.x_i$
8: $S_j.pBest_i \leftarrow X_o$
9: **end if**
10: **if** $X_o < \widehat{C}_{l_o}$ **then**
11: $\widehat{C}_l \leftarrow X$
12: $C_l \leftarrow \mathbf{B}(C, l, j, S_j.x_i)$
13: **else if** $X_o = \widehat{C}_{l_o}$ **then**
14: **if** $\widehat{C}_l$ does not dominate X **then**
15: $\widehat{C}_l \leftarrow X$
16: $C_l \leftarrow \mathbf{B}(C, l, j, S_j.x_i)$
17: **end if**
18: **end if**
19: $AddToArchive(\mathbf{B}(C, l, j, S_j.x_i), X)$
20: **end for**
21: **end for**
22: **end procedure**

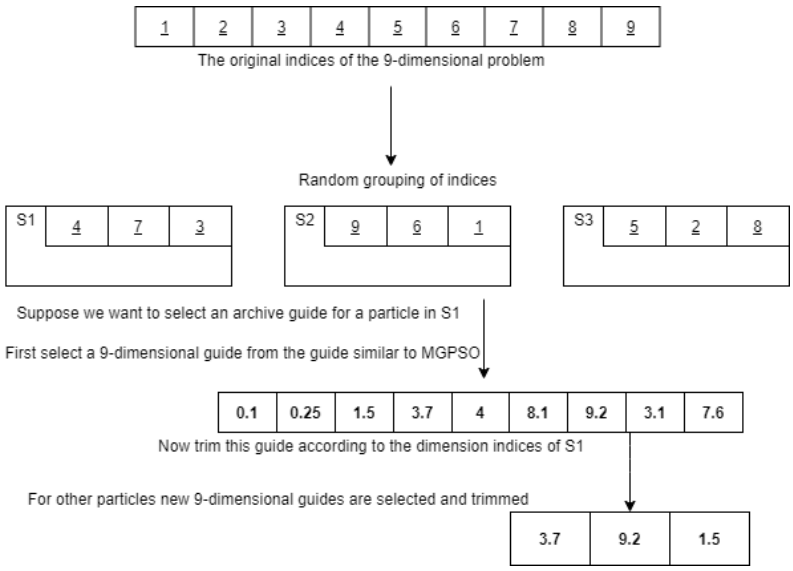

Figure 4.1: The archive guide implementation of CCMGPSO for $\frac{n_x}{k}$-dimensional particles explained through an arbitrary 9-dimensional problem whose decision variables are divided into three subpopulations.

4.3 Supplementary Experiments on CCMGPSO

In order to empirically verify some of the design choices made for CCMGPSO, and to further discuss some parts of this proposed approach, a small set of experiments was conducted. These supplementary experiments are fully discussed in Appendix B.

Chapter 5

Experimental Results

This chapter presents a comparative study involving the proposed CCMGPSO and six other multi-objective optimization algorithms from the literature on three well-known benchmark suites, namely WFG1 to WFG9, ZDT1 to ZDT6, and DTLZ1 to DTLZ7 that were used for this study.

5.1 Experimental Setup

The experimental setup of this study is provided in this section.

5.1.1 Implementation Details

The CCMGPSO was compared with MGPSO [64] [63], MMOPSO [45] (the algorithm with the best scalability from the previous scalability study found in Chapter 3), WOF-NSGAII [91], LMOCSO [75], S3-CMA-ES [6], and D-IBEA [24] (four approaches proposed for large-scale multi-objective optimization). For all algorithms except WOF-NSGAII and S3-CMA-ES, we rely on our own Java implementations. For WOF-NSGAII, the source code was obtained from the authors' website [1] and for S3-CMA-ES, the implementation in the PlatEMO framework [74] (as made available online by the authors) was used. The PlatEMO framework was also used for verifying the Java implementations of LMOCSO and D-IBEA against the ones provided by their respective authors. All plots were drawn using Matplolib [32] in Python. For algorithms implemented in Java, the benchmark suites of the jMetal [53] framework were used. For all algorithms, if the value of one dimension of the position vector

[1] https://www.ci.ovgu.de/Research/Codes.html

became greater than the upper bound or less than the lower bound, it was assigned the upper bound and the lower bound respectively.

5.1.2 Parameters

All parameters in all algorithms were set to their recommended values as per their respective authors. For S3-CMA-ES [6], the sample size for variable classification was set to 50, and the subpopulation size was set to 5. IBEA was embedded into the DGEA framework, (referred to as D-IBEA) as it had previously shown competitive results in [24] and the number of direction vectors in D-IBEA was set to 10. The value of δ in MMOPSO was set to 0.9 as recommended in [45]. In LMOCSO [75], the penalty parameter α was set to 2. For WOF-NSGAII, t_1 (the number of function evaluations for the original problem) and t_2 (the number of function evaluations for the transformed problem) were set to 1000 and 500 respectively, and the p-value transformation with $p = 0.2$ was used as the transformation function. Moreover, in WOF-NSGAII, the number of selected solutions (q) and the number of groups (γ) were set to 3 and 4 respectively.

For MGPSO on the WFG and ZDT problems, the same parameters as in [64] and [63] were used. For the DTLZ problems, MGPSO showed better performance when the tournament size was equal to 2 or 3; therefore, and for the ease of parameter tuning, the tournament size of MGPSO was set to 3 on the DTLZ problems. Other MGPSO parameters (c_1, c_2, c_3, and ω) were tuned on the 2000-dimensional DTLZ problems (the highest number of DTLZ dimensions in these experiments). For this, 5000 random combinations of these parameters were picked for each test instance and the best one was picked based on the IGD measure (the best mean over three independent runs). These 5000 combinations for c_1, c_2, c_3, and ω were picked from the same sets as previously defined in [64]:

- $c_1, c_2, c_3 \in \{0.50, 0.55, 0.6, \ldots, 1.9\}$, and
- $w \in \{0.05, 0.075, 0.1, \ldots, 0.95\}$.

The resulting parameters are listed in Table 5.1.

The number of individuals was set to 300 in all algorithms. For MGPSO, these 300 particles were split evenly between subswarms, $\{150, 150\}$ and $\{100, 100, 100\}$ for two and three objectives respectively. For CCMGPSO, these 300 particles were split between the subswarms in the following way:

- 10 n_x-dimensional particles per objective,

- $k = 280\ \frac{n_x}{280}$-dimensional particles for two objectives and $k = 270\ \frac{n_x}{270}$-dimensional particles for three objectives.

When applicable, the size of the external archive was also set to 300 in algorithms. The maximum number of function evaluations was used as the computational budget, the values of the maximum function evaluations used for each test instance are listed in Tables 5.2, 5.3, 5.5, 5.6, and 5.8 and were obtained using empirical observations.

Regarding the CCMGPSO parameters, the tournament size (for archive guide selection) was set to 2 on all test instances. The number of context vectors was set to n_m on DTLZ and ZDT problems and 10 on WFG problems. The value of γ was set to 10 for ZDT and DTLZ problems and 20 for WFG problems, this is discussed in more detail in the following sections.

5.1.3 Benchmark Suites

Three benchmark suites (a total of 21 functions) were used in this study, WFG, ZDT, and DTLZ. With reference to the WFG problems, 500- and 1000-dimensional 2- and 3-objective problems (as used in the previous scalability study) were used; moreover, an additional dimension level (750 dimensions) was added to the test instances. The number of position parameters (k) in WFG problems was set to 100, 150, and 200 for 500, 750, and 1000 dimensions respectively. With reference to the DTLZ and ZDT problems, the 1000-, 1500-, and 2000-dimensional problems were used. For DTLZ problems, two and three objectives were used; however, for ZDT problems only two objectives were used as the ZDT problems are not scalable objective-wise.

5.1.4 Performance Measures

The inverted generational distance (IGD) [12] [68] was used as the performance measure. This measure calculates the distance between the obtained front and a reference front, this work used the reference fronts provided in the jMetal framework [53] for this purpose. Please note that the hypervolume measure (HV) was excluded from this study due to the difficulties of selecting an appropriate reference point. As previously mentioned in [9], using the HV measure when the number of decision variables is relatively big might not be beneficial since on problems of such kind MOEAs usually fail to converge to the true front. This is more specifically the case on the large-scale instances of some DTLZ problems, such as DTLZ1, DTLZ3, and DTLZ6. Therefore, only IGD is used in this study for performance evaluation.

5.1.5 Statistical Significance

All algorithms were run 30 times on each test instance. The algorithms were compared using the Mann-Whitney U test [51] with a confidence level of 95%. For each pair of algorithms, if the difference was deemed statistically significant, the algorithm with the better mean over the 30 independent runs was given a win and the algorithm with the worse mean was given a loss. The difference between wins and losses ($wins - losses$) was then used for ranking the algorithms, as shown in Tables 5.2 to 5.9.

5.1.6 The Sensitivity Analysis of CCMGPSO Parameters

CCMGPSO has three main parameters, the tournament size (T), the number of context vectors (n_c), and the number of CPSO iterations per objective (γ). On all problems, competitive performance was observed when at least n_m context vectors were used. On ZDT and DTLZ problems, better performance was observed when exactly n_m context vectors were used. On most WFG problems, more than n_m context vectors (5 or 10) were required for competitive performance. An example depicting this for the multi-modal WFG2 is provided in Figure 5.1. As seen in the plot, CCMGPSO with 10 context vectors ended up with and spent most of the optimization process with better IGD values. It may be worth noting that, since the CCMGPSO assigns context vectors for γ iterations, using too many context vectors could hurt the overall convergence as some context vectors could not get enough iterations under a limited computational budget. Therefore, n_c was set to 10 on all WFG test instances.

Regarding the γ parameter, $\gamma = 10$ produced competitive performance on all DTLZ and ZDT problems. On WFG problems, $\gamma = 10$ or $\gamma = 20$ resulted in good performance; however, slightly better performance was observed with $\gamma = 20$ on some functions. Two examples of this for the 2-objective 2000-dimensional DTLZ6 and the 2-objective 1000-dimensional WFG6 are depicted in Figure 5.2. As seen in the plot, on both functions, $\gamma = 1$ iteration per objective was not enough for satisfactory convergence, only improving the IGD for the first few iterations over a completely random initial population. On DTLZ6, slightly better results were observed with $\gamma = 10$. As for the WFG6 problem in Figure 5.2, $\gamma = 10$ had better IGD performance than $\gamma = 20$ until the 900-th iteration possibly due to having shorter and thus more CPSO cycles per objective; however, in the long run, CCMGPSO with $\gamma = 20$ caught up with it and ended up with better IGD values. Similar to n_c, one must be careful with using γ values that are too big as such values could result in objectives and context vectors not getting enough CPSO cycles under a limited

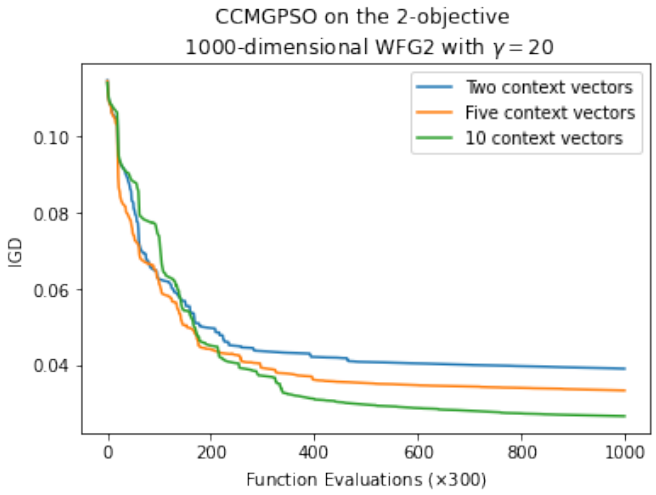

Figure 5.1: The sensitivity of the n_c parameter demonstrated on the 2-objective 1000-dimensional WFG2.

computational budget. Hence, the value of γ was set to 20 on all WFG problems.

5.2 Results of The Experimental Study

The experimental results of this study are listed in Tables 5.2 to 5.9 and discussed in the following subsections.

5.2.1 The WFG Test Suite

The results related to the WFG test suite are provided in Tables 5.2, 5.3, and 5.4. Tables 5.2 and 5.3 list the results for WFG1 to WFG4 and WFG5 to WFG9 respectively, whereas Table 5.4 provides overall results on the WFG suite based on aggregated scores.

Generally speaking, CCMGPSO had the best performance on most WFG problems, including WFG1, WFG3, WFG4, WFG5, and WFG7. On the other hand, S3-CMA-ES had the worst possible performance on all functions. This is due to the fact that this algorithm uses a lot of function evaluations prior to the optimization process for decision variable analysis, and thus could be left with very few or no function evaluations when the variables are classified. Under a limited computational budget, algorithms such as WOF-NSGAII and CCMGPSO that (perhaps indirectly) take variable interactions into account while still working towards a better

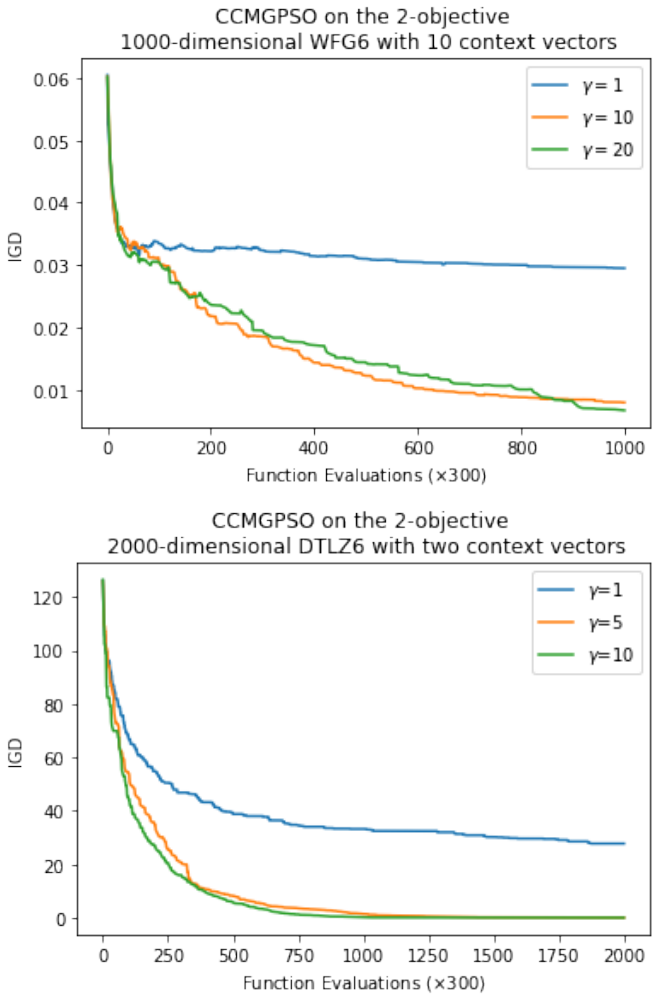

Figure 5.2: The sensitivity of the γ parameter demonstrated on the 2-objective 1000-dimensional WFG6 and the 2-objective 2000-dimensional DTLZ6.

Table 5.1: The MGPSO parameters tuned for the DTLZ problems

		Parameter			
Problem	Objective	c_1	c_2	c_3	ω
DTLZ1	2	0.80	1.35	1.90	0.80
	3	1.30	1.55	1.90	0.75
DTLZ2	2	0.95	0.55	0.80	0.25
	3	1.30	0.70	0.50	0.25
DTLZ3	2	0.65	1.80	1.50	0.55
	3	1.80	1.65	1.75	0.70
DTLZ4	2	1.75	0.50	0.55	0.15
	3	1.30	0.50	0.65	0.15
DTLZ5	2	1.35	0.75	0.50	0.15
	3	1.45	0.65	0.50	0.15
DTLZ6	2	1.55	1.70	1.90	0.80
	3	1.90	1.60	1.75	0.80
DTLZ7	2	1.55	1.50	1.95	0.65
	3	1.75	1.85	0.60	0.80

POF could outperform the ones that use separate function evaluations for decision variable analysis.

On WFG1, CCMGPSO had the best performance on all 2- and 3-objective test instances; however, despite not being statistically outperformed, CCMGPSO achieved a worse mean than that of WOF-NSGAII's on the 3-objective 1000-dimensional WFG1. After CCMGPSO, WOF-NSGAII, MMOPSO and LMOCSO were the best-performing algorithms. MGPSO and D-IBEA both showed poor scalability on WFG1, constantly being outperformed by LMOCSO, WOF-NSGAII, and CCMGPSO. MMOPSO again showed good scalability on WFG1 similar to the previous scalability study. Previous studies [49] had also shown the efficiency of algorithms using the penalty-based boundary intersection method (PBI) (such as MOEA/D [87] or MMOPSO [45]) in solving biased multi-objective optimization problems with mixed POF geometries (such as WFG1).

On the non-separable multi-modal WFG2, CCMGPSO had the best performance for two objectives; however, it was outperformed by WOF-NSGAII and MMOPSO for three objectives. This could be indicative of a potential weakness of the context vector-based approaches for multi-objective optimization. Because a limited number of context vectors are used to guide the individuals, there is a probability that all of these context vectors could end up in a local optimum that is far from the global one. On WFG2, this performance drop happened when the objective space got bigger (from

two to three objectives). LMOCSO showed poor scalability on WFG2, constantly being outperformed by other algorithms including D-IBEA and MGPSO. D-IBEA outperformed MGPSO on the 2-objective test instances of WFG2, however; similar to CCMGPSO, it suffered from a performance drop from two to three objectives.

On WFG3 and the multi-modal WFG4, CCMGPSO was the best-performing algorithm on all test instances except the 500-dimensional 2-objective WFG3. Similar to WFG2, D-IBEA had a loss in relative IGD performance from two to three objectives and LMOCSO had poor dimension-wise scalability. LMOCSO outperformed D-IBEA and MGPSO on the 2- and 3-objective WFG4. It may be worth mentioning that using big values for k (number of dimension groups) and dividing the large-scale problems into smaller subproblems resulted in improved performances on the separable and multi-modal WFG4. An example demonstrating this on the 2-objective 1000-dimensional WFG4 is provided in Figure 5.3. With $k = 1$ subswarm having 280 particles, CCMGPSO obtained a POF similar to that of the MGPSO's, which ranked fifth in Table 5.2 with reference to IGD. However, with the increase in the value of k, the obtained POFs visibly got better.

With few exceptions, CCMGPSO had the best performance on most test instances of WFG5 to WFG7. D-IBEA and MGPSO showed poor scalability on WFG5, LMOCSO showed poor scalability on WFG5 for two objectives but outperformed D-IBEA and MGPSO for three objectives. On the 2-objective 1000-dimensional WFG6 (a non-separable function), MMOPSO, WOF-NSGAII, and CCMGPSO had the best (and statistically similar) performances, although WOF-NSGAII obtained the best mean over 30 runs. On WFG6, D-IBEA was the best algorithm after CCMGPSO, WOF-NSGAII, and MMOPSO, constantly outperforming MGPSO, LMOCSO, and S3-CMA-ES.

On WFG7, CCMGPSO was the best-performing algorithm on all test instances, obtaining the maximum possible score (+6) in all of them. D-IBEA and LMOCSO failed to show competitive performance on WFG7, as they were often outperformed by all algorithms except S3-CMA-ES. On the non-separable WFG8, CCMGPSO and WOF-NSGAII had the best performances. CCMGPSO outperformed WOF-NSGAII on the 500-dimensional 2- and 3-objective WFG8, but was outperformed by it on the 750- and 1000-dimensional 2-objective WFG8. Besides S3-CMA-ES, MGPSO and LMOCSO were the worst-performing algorithms on WFG8. Interestingly enough, MMOPSO was outperformed by D-IBEA on the 750- and 1000-dimensional 2-objective WFG8.

Finally, the worst CCMGPSO performance on an individual problem was observed

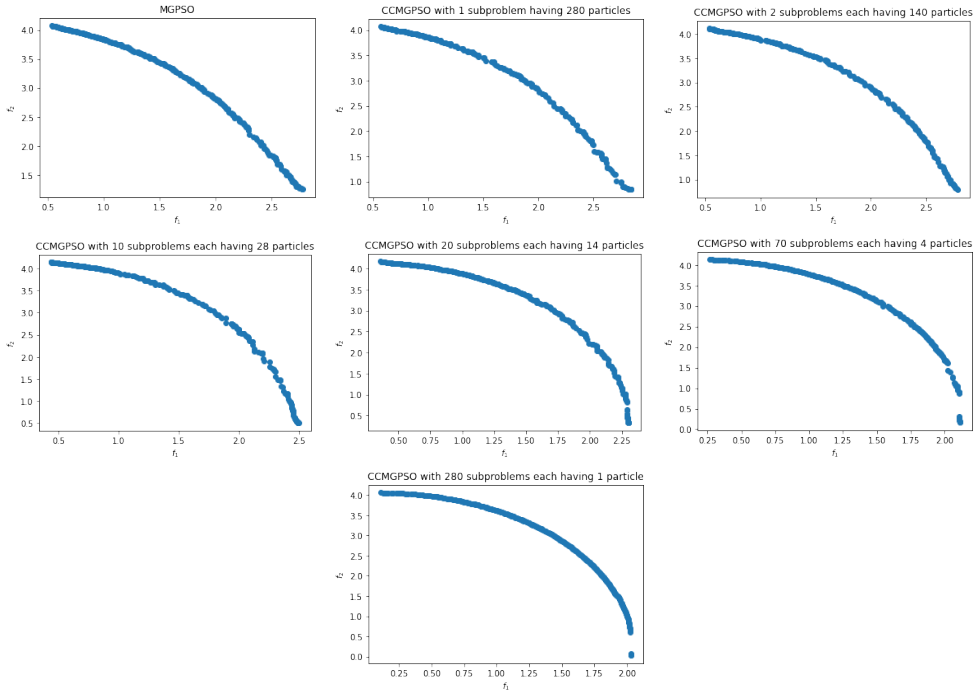

Figure 5.3: The efficiency of dividing a large-scale multi-objective optimization problem into smaller subproblems demonstrated on the 2-objective 1000-dimensional WFG4. The figures are all Pareto-optimal fronts.

on WFG9, which with different properties such as multi-modality, non-separability and a deceptive fitness landscape[2] is considered to be a rather difficult problem. CCMGPSO ranked fourth overall on the 750- and 1000-dimensional 2-objective WFG9, after being outperformed by WOF-NSGAII, MMOPSO, and D-IBEA which were the best-performing algorithms in the aforementioned order. WOF-NSGAII was significantly the best-performing algorithm on WFG9, ranking first on all test instances and only failing to outperform MMOPSO on the 500-dimensional 3-objective WFG9 while still obtaining the better mean. Similar to CCMGPSO, MGPSO, and S3-CMA-ES, LMOCSO also showed poor scalability on almost all test instances of WFG9.
A visual comparison of these algorithms on the 1000-dimensional 3-objective WFG3 is depicted in Figure 5.4. As seen in Figure 5.4, CCMGPSO showed both better convergence and diversity on WFG3, followed by WOF-NSGAII. MMOPSO found a relatively well-converged POF; however, the diversity was not on par with CCMGPSO or WOF-NSGAII.

[2]As defined in [30], in a deceptive objective function the majority of the search space leans towards a deceptive optimum instead of the true one.

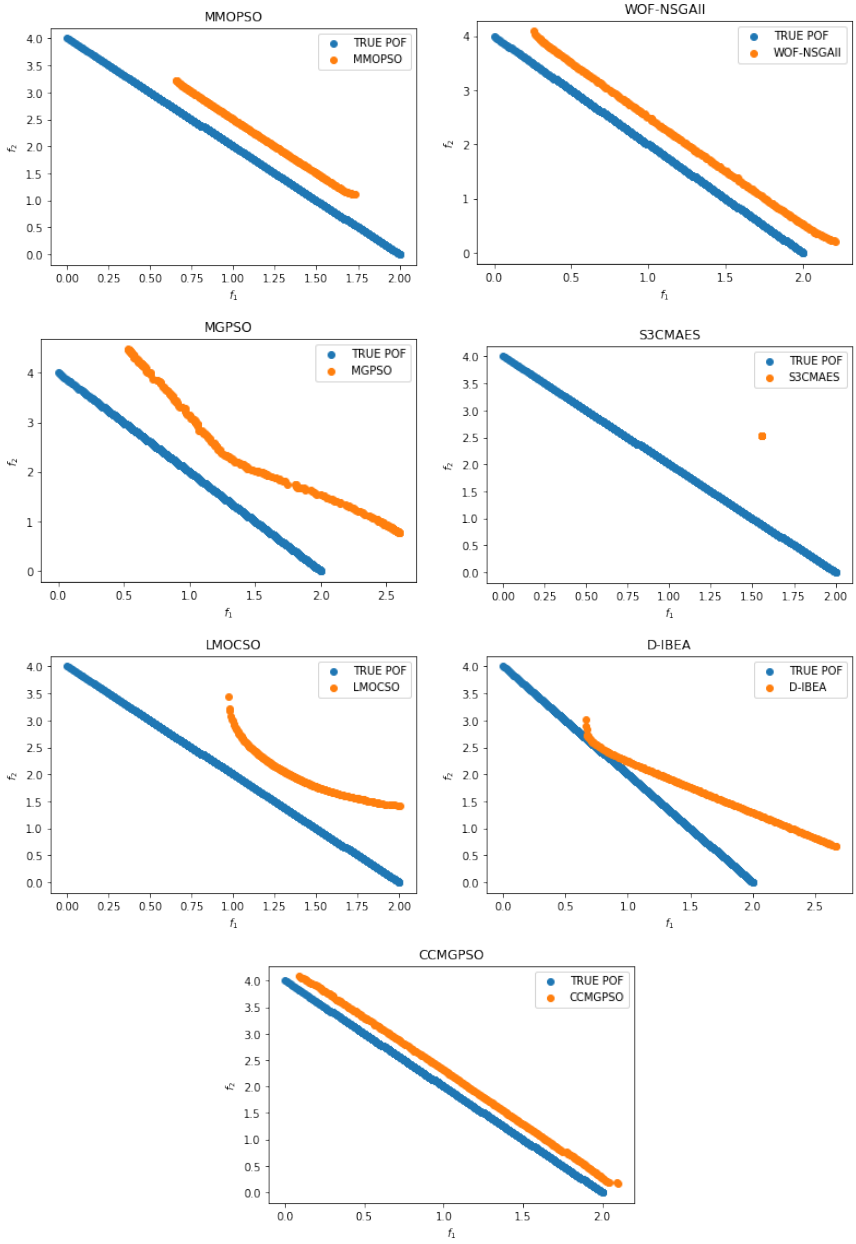

Figure 5.4: A visual comparison of the obtained POFs by the algorithms on the 1000-dimensional 2-objective WFG3.

Table 5.2: IGD rankings for WFG1 to WFG4. The highest-ranking algorithm is highlighted in grey background. If two or more algorithms were ranked first, all of them were marked using "≈" and the one with the best mean IGD score over 30 independent runs was highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

					Algorithm						
Problem[1]	Objectives	Dimensions	FEs	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
WFG1 (**S**, **U**)	2	500	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		750	3.00e+5	Difference	2	4	-2	-6	-1	-3	6
				Rank	3	2	5	7	4	6	1
		1000	3.00e+5	Difference	2	4	-2	-6	-2	-2	6
				Rank	3	2	4	5	4	4	1
	3	500	3.00e+5	Difference	2	4	-4	-6	0	-2	6
				Rank	3	2	6	7	4	5	1
		750	3.00e+5	Difference	2	4	-4	-6	0	-2	6
				Rank	3	2	6	7	4	5	1
		1000	3.00e+5	Difference	2	5≈	-4	-6	0	-2	5≈
				Rank	2	1≈	5	6	3	4	1≈
WFG2 (**NS**, **U-M**)	2	500	3.00e+5	Difference	3	0	-2	-6	1	-2	6
				Rank	2	4	5	6	3	5	1
		750	3.00e+5	Difference	1	4	-1	-6	-4	0	6
				Rank	3	2	5	7	6	4	1
		1000	3.00e+5	Difference	1	3	-2	-6	-4	2	6
				Rank	4	2	5	7	6	3	1
	3	500	3.00e+5	Difference	4	6	-1	-6	-3	-2	2
				Rank	2	1	4	7	6	5	3
		750	3.00e+5	Difference	4	6	0	-6	-2	-4	2
				Rank	2	1	4	7	5	6	3
		1000	3.00e+5	Difference	3	6	0	-6	-2	-4	3
				Rank	2	1	3	6	4	5	2
WFG3 (**NS**, **U**)	2	500	3.00e+5	Difference	2	6	-4	-6	-2	0	4
				Rank	3	1	6	7	5	4	2
		750	3.00e+5	Difference	2	4	-3	-6	-3	0	6
				Rank	3	2	5	6	5	4	1
		1000	3.00e+5	Difference	2	4	-2	-6	-4	0	6
				Rank	3	2	5	7	6	4	1
	3	500	3.00e+5	Difference	3	3	-1	-6	-1	-4	6
				Rank	2	2	3	5	3	4	1
		750	3.00e+5	Difference	2	4	0	-6	-3	-3	6
				Rank	3	2	4	6	5	5	1
		1000	3.00e+5	Difference	2	4	0	-6	-3	-3	6
				Rank	3	2	4	6	5	5	1
WFG4 (**S**, **M**)	2	500	3.00e+5	Difference	4	2	-2	-6	0	-4	6
				Rank	2	3	5	7	4	6	1
		750	3.00e+5	Difference	4	2	-2	-6	0	-4	6
				Rank	2	3	5	7	4	6	1
		1000	3.00e+5	Difference	4	2	-2	-6	0	-4	6
				Rank	2	3	5	7	4	6	1
	3	500	3.00e+5	Difference	2	4	-1	-6	-1	-4	6
				Rank	3	2	4	6	4	5	1
		750	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		1000	3.00e+5	Difference	2	4	-1	-6	-1	-4	6
				Rank	3	2	4	6	4	5	1

(1)S: Separable, **NS**: Non-separable, **U**: Uni-modal, **M**: Multi-modal, **U-M**: Both uni-modal and multi-modal (on different objectives), and **D**: Deceptive.

Table 5.3: IGD rankings for WFG5 to WFG9. The highest-ranking algorithm is highlighted in grey background. If two or more algorithms were ranked first, all of them were marked using "≈" and the one with the best mean IGD score over 30 independent runs was highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

					Algorithm						
Problem(1)	Objectives	Dimensions	FEs	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
WFG5 (**S**, **D**)	2	500	3.00e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
		750	3.00e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
		1000	3.00e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
	3	500	3.00e+5	Difference	2	6	-2	-6	0	-4	4
				Rank	3	1	5	7	4	6	2
		750	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		1000	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
WFG6 (**NS**, **U**)	2	500	3.00e+5	Difference	2	2	-4	-6	-2	2	6
				Rank	2	2	4	5	3	2	1
		750	3.00e+5	Difference	3	4	-4	-6	-2	0	5
				Rank	3	2	6	7	5	4	1
		1000	3.00e+5	Difference	4≈	4≈	-4	-6	-2	0	4≈
				Rank	1≈	1≈	4	5	3	2	1≈
	3	500	3.00e+5	Difference	0	6	-2	-6	-4	3	3
				Rank	3	1	4	6	5	2	2
		750	3.00e+5	Difference	-1	4	-4	-6	-1	3	5
				Rank	4	2	5	6	4	3	1
		1000	3.00e+5	Difference	0	5≈	-2	-6	-4	2	5≈
				Rank	3	1≈	4	6	5	2	1≈
WFG7 (**S**, **U**)	2	500	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		750	3.00e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
		1000	3.00e+5	Difference	2	4	0	-6	-4	-2	6
				Rank	3	2	4	7	6	5	1
	3	500	3.00e+5	Difference	2	4	-1	-6	-3	-2	6
				Rank	3	2	4	7	6	5	1
		750	3.00e+5	Difference	2	4	0	-6	-4	-2	6
				Rank	3	2	4	7	6	5	1
		1000	3.00e+5	Difference	2	4	-1	-6	-4	-1	6
				Rank	3	2	4	6	5	4	1
WFG8 (**NS**, **U**)	2	500	3.00e+5	Difference	2	4	-2	-6	-4	0	6
				Rank	3	2	5	7	6	4	1
		750	3.00e+5	Difference	0	6	-2	-6	-4	2	4
				Rank	4	1	5	7	6	3	2
		1000	3.00e+5	Difference	0	6	-2	-6	-4	2	4
				Rank	4	1	5	7	6	3	2
	3	500	3.00e+5	Difference	2	4	-4	-6	-2	0	6
				Rank	3	2	6	7	5	4	1
		750	3.00e+5	Difference	2	5≈	-2	-6	-4	0	5≈
				Rank	2	1≈	4	6	5	3	1≈
		1000	3.00e+5	Difference	2	5≈	-2	-6	-4	0	5≈
				Rank	2	1≈	4	6	5	3	1≈
WFG9 (**NS**, **M**, **D**)	2	500	3.00e+5	Difference	2	6	-4	-6	-2	2	2
				Rank	2	1	4	5	3	2	2
		750	3.00e+5	Difference	3	6	-2	-6	-4	2	1
				Rank	2	1	5	7	6	3	4
		1000	3.00e+5	Difference	3	6	-2	-6	-4	2	1
				Rank	2	1	5	7	6	3	4
	3	500	3.00e+5	Difference	5≈	5≈	-3	-6	-3	0	2
				Rank	1≈	1≈	4	5	4	3	2
		750	3.00e+5	Difference	4	6	-3	-6	-3	0	2
				Rank	2	1	5	6	5	4	3
		1000	3.00e+5	Difference	4	6	-2	-6	-4	0	2
				Rank	2	1	5	7	6	4	3

(1)S: Separable, **NS**: Non-separable, **U**: Uni-modal, **M**: Multi-modal, **U-M**: Both uni-modal and multi-modal (on different objectives), and **D**: Deceptive.

Table 5.4: The overall performance of each algorithm on the WFG test suite, based on the aggregated scores. The best-performing algorithms were highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

				Algorithm						
Overall	Objectives	Dimensions	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
WFG1 to WFG9	2	500	Difference	21	32	-22	-54	-11	-14	48
			Rank	3	2	6	7	4	5	1
		750	Difference	19	38	-16	-54	-22	-11	46
			Rank	3	2	5	7	6	4	1
		1000	Difference	20	37	-16	-54	-26	-6	45
			Rank	3	2	5	7	6	4	1
	3	500	Difference	22	42	-19	-54	-17	-15	41
			Rank	3	1	6	7	5	4	2
		750	Difference	19	41	-17	-54	-17	-16	44
			Rank	3	2	5	6	5	4	1
		1000	Difference	19	43	-14	-54	-22	-16	44
			Rank	3	2	4	7	6	5	1

5.2.2 Summary of Results on The WFG Test Suite

The overall IGD rankings of the algorithms (based on the aggregated scores) on the WFG test suite are listed in Table 5.4. As seen in Table 5.4, CCMGPSO had the best overall score on all test instances expect the 500-dimensional 3-objective problems, where WOF-NSGAII had the best overall score. CCMGPSO was followed by WOF-NSGAII and MMOPSO as the second- and third-best performing algorithms respectively. S3-CMA-ES was always the worst-performing algorithm due to the reasons discussed in the previous section, mostly as a result of spending too many function evaluations on variable analysis prior to the optimization process and leaving little or no computational budget for the main search. As seen in Tables 5.2, 5.3, and 5.4, LMOCSO, MGPSO, and D-IBEA all showed poor scalability on the WFG suite. LMOCSO's velocity update mechanism based on the competitive swarm optimizer (CSO) that was proposed to improve diversity, showed unsatisfactory convergence on the large-scale instances of the WFG problems. However, this approach showed minor signs of scalability on the multi-modal WFG4, with LMOCSO outperforming all algorithms except the first three (CCMGPSO, WOF-NSGAII, and MMOPSO) on most test instances. D-IBEA managed to outperform MGPSO and LMOCSO (on WFG3, WFG6, and WFG9), MMOPSO on the 750-dimensional 3-objective WFG6, and CCMGPSO on some instances of WFG9; however, it showed really poor performance on other problems such as WFG4, WFG5, and WFG7, resulting in an overall mediocre performance.

Regarding CCMGPSO, it had the best overall scores for all experiments except one as discussed above and listed in Table 5.4. With reference to individual problems and

the different types of them, CCMGPSO showed extremely competitive performance on separable and uni-modal problems, achieving the best performance on all experiments involving them (WFG1 and WFG7). The experiments in which CCMGPSO was outperformed all involved problems that were multi-modal (such as being outperformed by WOF-NSGAII on some instances of WFG2, and all instances of WFG9), deceptive (such as being outperformed by WOF-NSGAII on one instance of WFG5, and by MMOPSO and D-IBEA on some instances of WFG9) or non-separable (such as being outperformed on some instances of WFG6, WFG8, and WFG9). Perhaps unsurprisingly, CCMGPSO's worst performance on an individual optimization problem was observed on WFG9, a problem which has all of the aforementioned properties simultaneously (multi-modal, non-separable and deceptive). When an optimization problem is multi-modal or deceptive, using a relatively small set of context vectors to guide the search could potentially hurt the diversity of particles if all or some of these context vectors end up in a local optimum which is far away from the global one. In the case of non-separable problems, the use of many dimension groups (280 and 270 for 2- and 3-objective functions respectively in these experiments) into which decision variables are classified could hurt the overall performance as all decision variables in problems of such kind interact with each other and have to be optimized together. Another interesting observation was on WFG4 (separable and multi-modal), where CCMGPSO had the best performance in all experiments.

5.2.3 The DTLZ Test Suite

The results related to the DTLZ test suite are provided in Tables 5.5, 5.6, and 5.7. Tables 5.5 and 5.6 list the results for DTLZ1 to DTLZ4 and DTLZ5 to DTLZ7 respectively, whereas Table 5.7 provides overall results on the DTLZ suite based on aggregated scores.

On DTLZ1, CCMGPSO had the best performance, outperforming its counterparts in most experiments. On the 2000-dimensional 3-objective DTLZ1, CCMGPSO, WOF-NSGAII, LMOCSO, and D-IBEA had statistically similar results, despite CCMGPSO obtaining the best mean over 30 independent runs. MGPSO showed poor performance on the multi-modal DTLZ1, constantly being outperformed by all other algorithms except S3-CMA-ES.

Similar to DTLZ1, CCMGPSO also had the best performance on DTLZ2, obtaining the best possible score (+6) in all experiments. D-IBEA and MGPSO both showed poor scalability on DTLZ2, whereas WOF-NSGAII and MMOPSO were the second-

and third-best performing algorithms after CCMGPSO on DTLZ2 respectively. On the multi-modal DTLZ3, D-IBEA and CCMGPSO showed relatively better scalability compared with other algorithms, although CCMGPSO ranked third behind D-IBEA and LMOCSO (also in that order) on the 2000-dimensional 2-objective DTLZ3. Similar to DTLZ1 (another multi-modal problem), MGPSO showed poor scalability on DTLZ3. Despite competitive performance on almost all problems, WOF-NSGAII was often outperformed on the test instances of DTLZ3; for example, WOF-NSGAII ranked fourth on the 1000-dimensional 3-objective DTLZ3 behind CCMGPSO and D-IBEA (tied at first), LMOCSO, and MMOPSO.

With reference to DTLZ4, CCMGPSO was outperformed by WOF-NSGAII for two objectives, but had the best performance for three objectives. On the 2-objective DTLZ4, CCMGPSO was outperformed mainly due to some bad runs, despite having some good runs as well. As depicted in Figure B.1, a bad run of CCMGPSO on the 2-objective DTLZ4 typically involved a POF that only covered a very specific (and small) portion of the true POF. The reason behind this could be the existence of *bias* in DTLZ4 [31]. As previously defined in [31], in an optimization problem with bias an evenly distributed set of decision vectors does not map onto an evenly distributed set of objective vectors resulting in a highly non-uniform POF in DTLZ4 [9]. In order to visually illustrate this, 5000 vectors were randomly distributed in the decision spaces of the 2000-dimensional 2-objective DTLZ2, DTLZ4, DTLZ5, and DTLZ6. These 5000 vectors were then evaluated and plotted as depicted in Figure 5.6. As seen in Figure 5.6, DTLZ4 had a more non-uniform set of objective vectors compared with DTLZ2, DTLZ5, and DTLZ6, where one specific area of the plot (close to $f_2 = 0$) was more crowded with vectors. As seen in Figure B.2, in bad runs of CCMGPSO on the 2-objective DTLZ4 only one Pareto-optimal solution was found in the same region (close to $f_2 = 0$). This could be a result of all context vectors ending up in one specific area of the objective space where f_2 is close to zero; in such cases, CCMGPSO will keep optimizing f_1 until it is equal or very close to 1, producing only one Pareto-optimal point. The same phenomenon was also observed on the 3-objective DTLZ4 (as shown in Figure 5.7); however, it did not happen as often and therefore CCMGPSO was able to statistically outperform all other algorithms on the 3-objective DTLZ4.

On DTLZ5, CCMGPSO was the best-performing algorithm in all experiments, whereas D-IBEA and MGPSO both showed poor scalability. CCMGPSO and WOF-NSGAII were the best-performing algorithms on DTLZ6 and DTLZ7. On the 2-objective DTLZ6, CCMGPSO and WOF-NSGAII had statistically similar perfor-

Table 5.5: IGD rankings for DTLZ1 to DTLZ4. The highest-ranking algorithm is highlighted in grey background. If two or more algorithms were ranked first, all of them were marked using "≈" and the one with the best mean IGD score over 30 independent runs was highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

					Algorithm						
Problem[(1)]	Objectives	Dimensions	FEs	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
DTLZ1 (**S**, **M**)	2	1000	3.00e+5	Difference	3	3	-4	-6	0	-2	6
				Rank	2	2	5	6	3	4	1
		1500	4.50e+5	Difference	2	2	-4	-6	2	-2	6
				Rank	2	2	4	5	2	3	1
		2000	6.00e+5	Difference	1	4	-4	-6	1	-2	6
				Rank	3	2	5	6	3	4	1
	3	1000	3.00e+5	Difference	-1	-1	-4	-6	3	3	6
				Rank	3	3	4	5	2	2	1
		1500	5.10e+5	Difference	1	1	-4	-6	1	1	6
				Rank	2	2	3	4	2	2	1
		2000	6.00e+5	Difference	-2	3≈	-4	-6	3≈	3≈	3≈
				Rank	2	1≈	3	4	1≈	1≈	1≈
DTLZ2 (**S**, **U**)	2	1000	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		1500	4.50e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		2000	6.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
	3	1000	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		1500	5.10e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		2000	6.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
DTLZ3 (**S**, **M**)	2	1000	3.00e+5	Difference	0	0	-4	-6	3	1	6
				Rank	4	4	5	6	2	3	1
		1500	4.50e+5	Difference	0	0	-4	-6	4≈	2	4≈
				Rank	3	3	4	5	1≈	2	1≈
		2000	6.00e+5	Difference	-1	3	-4	-6	3	4	1
				Rank	4	2	5	6	2	1	3
	3	1000	3.00e+5	Difference	0	-2	-4	-6	2	5≈	5≈
				Rank	3	4	5	6	2	1≈	1≈
		1500	5.10e+5	Difference	-1	-1	-4	-6	2	5≈	5≈
				Rank	3	3	4	5	2	1≈	1≈
		2000	6.00e+5	Difference	-1	-1	-4	-6	3	6	3
				Rank	3	3	4	5	2	1	2
DTLZ4 (**S**, **U**)	2	1000	3.00e+5	Difference	2	6	0	-6	-3	-3	4
				Rank	3	1	4	6	5	5	2
		1500	4.50e+5	Difference	2	6	0	-6	-3	-3	4
				Rank	3	1	4	6	5	5	2
		2000	6.00e+5	Difference	2	6	0	-6	-3	-3	4
				Rank	3	1	4	6	5	5	2
	3	1000	3.00e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
		1500	5.10e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
		2000	6.00e+5	Difference	1	4	1	-6	-2	-4	6
				Rank	3	2	3	6	4	5	1

(1)S: Separable, **NS**: Non-separable, **U**: Uni-modal, **M**: Multi-modal, **U-M**: Both uni-modal and multi-modal (on different objectives), and **D**: Deceptive.

Table 5.6: IGD rankings for DTLZ5 to DTLZ7. The highest-ranking algorithm is highlighted in grey background. If two or more algorithms were ranked first, all of them were marked using "≈" and the one with the best mean IGD score over 30 independent runs was highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

					Algorithm						
Problem[(1)]	Objectives	Dimensions	FEs	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
DTLZ5 (**S**, **U**)	2	1000	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		1500	4.50e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		2000	6.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
	3	1000	3.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		1500	5.10e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
		2000	6.00e+5	Difference	2	4	-2	-6	0	-4	6
				Rank	3	2	5	7	4	6	1
DTLZ6 (**S**, **U**)	2	1000	3.00e+5	Difference	0	5≈	-2	-6	2	-4	5≈
				Rank	3	1≈	4	6	2	5	1≈
		1500	4.50e+5	Difference	0	5≈	-3	-6	2	-3	5≈
				Rank	3	1≈	4	5	2	4	1≈
		2000	6.00e+5	Difference	0	5≈	-3	-6	2	-3	5≈
				Rank	3	1≈	4	5	2	4	1≈
	3	1000	3.00e+5	Difference	0	4	-2	-6	2	-4	6
				Rank	4	2	5	7	3	6	1
		1500	5.10e+5	Difference	0	4	-2	-6	2	-4	6
				Rank	4	2	5	7	3	6	1
		2000	6.00e+5	Difference	0	6	-3	-6	2	-3	4
				Rank	4	1	5	6	3	5	2
DTLZ7 (**S**, **U-M**)	2	1000	3.00e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
		1500	4.50e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
		2000	6.00e+5	Difference	2	4	0	-6	-2	-4	6
				Rank	3	2	4	7	5	6	1
	3	1000	3.00e+5	Difference	2	6	0	-6	-2	-4	4
				Rank	3	1	4	7	5	6	2
		1500	5.10e+5	Difference	2	5≈	0	-6	-2	-4	5≈
				Rank	2	1≈	3	6	4	5	1≈
		2000	6.00e+5	Difference	2	5≈	0	-6	-2	-4	5≈
				Rank	2	1≈	3	6	4	5	1≈

(1)S: Separable, **NS**: Non-separable, **U**: Uni-modal, **M**: Multi-modal, **U-M**: Both uni-modal and multi-modal (on different objectives), and **D**: Deceptive.

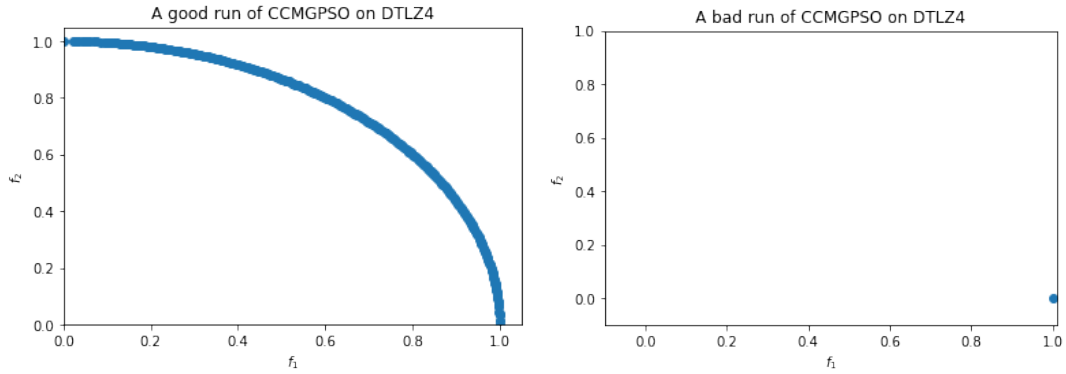

Figure 5.5: Examples of a good and a bad run of CCMGPSO on the 2000-dimensional 2-objective DTLZ4

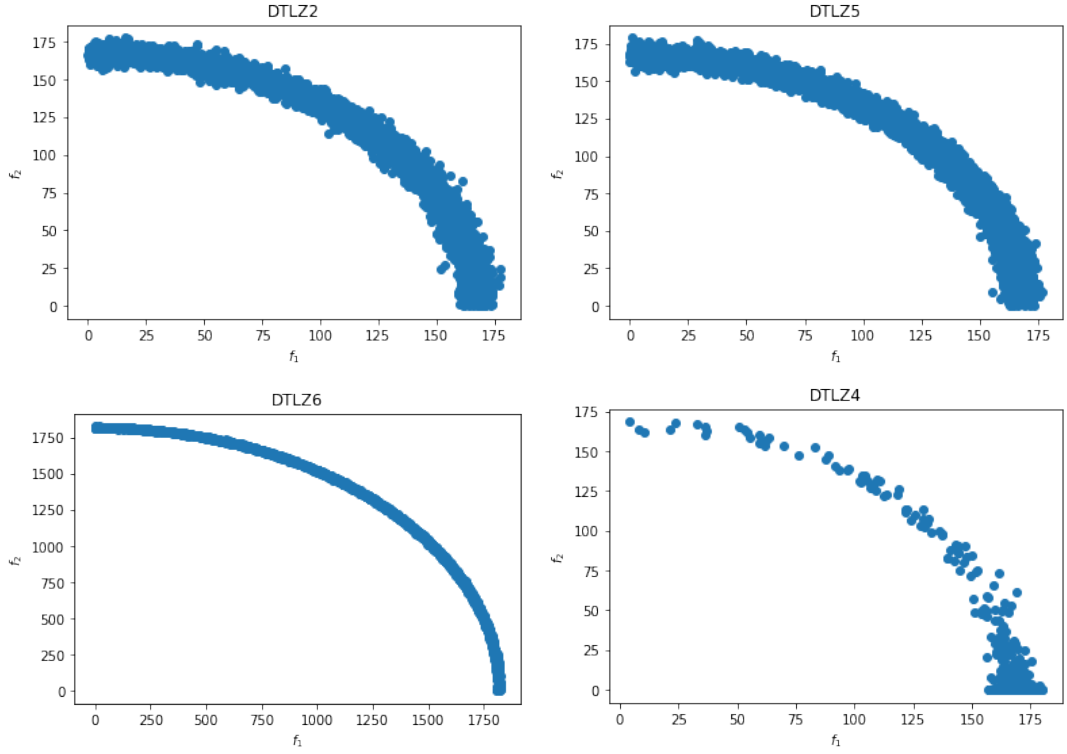

Figure 5.6: The effect of bias demonstrated by sampling 5000 random vectors in the decision spaces of the 2-objective 2000-dimensional DTLZ2, DTLZ5, DTLZ6, and DTLZ4

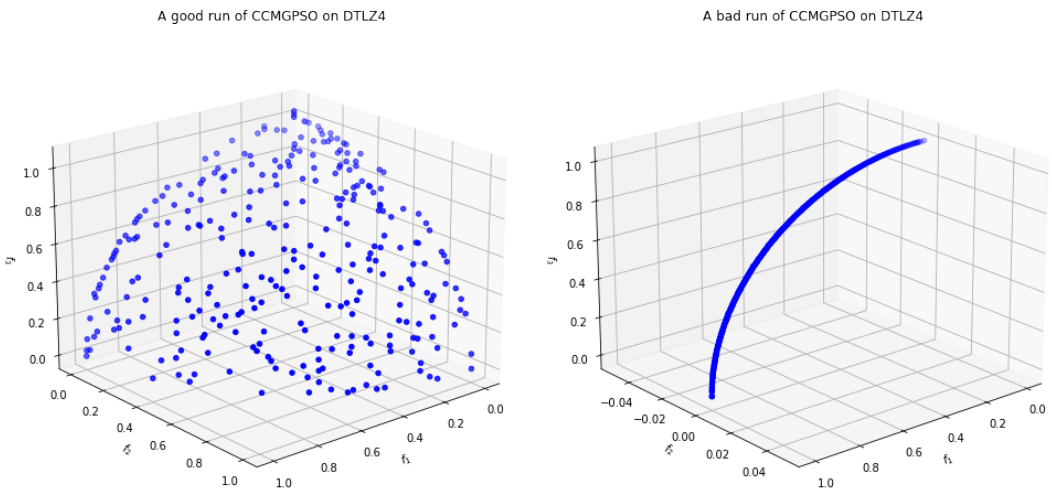

Figure 5.7: Examples of a good and a bad run of CCMGPSO on the 2000-dimensional 3-objective DTLZ4

mances, with the latter obtaining the best mean over 30 independent runs. On the 3-objective DTLZ6, CCMGPSO statistically outperformed WOF-NSGAII for 1000 and 1500 dimensions but was outperformed for 2000 dimensions. Similar to DTLZ5, MGPSO and D-IBEA showed poor performances on DTLZ6 and were constantly outperformed by CCMGPSO, WOF-NSGAII and LMOCSO. LMOCSO also managed to outperform MMOPSO on the 3-objective DTLZ6.

Finally, on DTLZ7, CCMGPSO and WOF-NSGAII again were the best performers. CCMGPSO outperformed all algorithms for two objectives but had a statistically similar performance to WOF-NSGAII for three objectives. MGPSO showed relatively good scalability on DTLZ7, outperforming both LMOCSO and D-IBEA. A visual comparison of these algorithms on the 2000-dimensional 3-objective DTLZ7 is depicted in Figure 5.8. As seen in Table 5.6 and Figure 5.8, CCMGPSO and WOF-NSGAII were the best performers on this problem, with the best convergence and diversity in their obtained POFs. Moreover, it can be seen that the solutions in LMOCSO's obtained POF were more equally-spaced than those of CCMGPSO's or MMOPSO's, with LMOCSO having fewer solutions in its obtained POF. The reason behind this is the use of a reference vector-based selection strategy in LMOCSO; where n_s (swarm size) uniformly-distributed reference vectors in the objective space are used to select the next population, such that for each reference vector up to one solution is selected. This means that, if some solutions are clustered in area of the POF, most of them are discarded. Since LMOCSO does not use an external archive,

Table 5.7: The overall performance of each algorithm on the DTLZ test suite, based on the aggregated scores. The best-performing algorithms were highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

				Algorithm						
Overall	Objectives	Dimensions	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
DTLZ1 to DTLZ7	2	1000	Difference	11	26	-14	-42	0	-20	39
			Rank	3	2	5	7	4	6	1
		1500	Difference	10	25	-15	-42	3	-18	37
			Rank	3	2	5	7	4	6	1
		2000	Difference	8	30	-15	-42	1	-16	34
			Rank	3	2	5	7	4	6	1
	3	1000	Difference	7	19	-14	-42	3	-12	39
			Rank	3	2	6	7	4	5	1
		1500	Difference	8	21	-14	-42	1	-14	40
			Rank	3	2	5	6	4	5	1
		2000	Difference	4	25	-14	-42	4	-10	33
			Rank	3	2	5	6	3	4	1

discarding solutions based on diversity can potentially hurt the overall performance when the solutions are not well converged yet.

5.2.4 Summary of Results on The DTLZ Test Suite

The overall results on the DTLZ suite (based on aggregated scores) are provided in Table 5.7. As seen in Table 5.7, CCMGPSO achieved the best overall scores on DTLZ1 to DTLZ7 in all six experiments, followed by WOF-NSGAII and MMOPSO at second and third respectively. LMOCSO mostly ranked fourth; however, on the 2000-dimensional 3-objective DTLZ problems, it was tied with MMOPSO at third, possibly due to its superiority on DTLZ6.

MGPSO and D-IBEA both failed to show competitive performance on most DTLZ problems. D-IBEA showed scalable performance on fully multi-modal problems (DTLZ1 and DTLZ3), whereas MGPSO managed to outperform LMOCSO and D-IBEA on DTLZ7. D-IBEA showing promising results on multi-modal could imply the efficiency of its proposed offspring generation approach based on direction vectors in maintaining both convergence and diversity in large-scale environments. On the other hand, MGPSO performed poorly on DTLZ1 and DTLZ3, one reason of which could be a false sense of the global best position as discussed in the previous chapter (see Section B.1 in Appendix B). For example, on DTLZ1, at the very early stages of optimization a decision vector whose corresponding objective vector has the minimum value for one objective but poor values for the others could be set to the global best of that specific objective. If such a decision vector is found for all objectives early on, the overall performance of MGPSO can be damaged to the point that even

the archive guide can become obsolete, as these global best positions are never updated. On the other hand, CCMGPSO constantly shares different global best vectors (context vectors) between objectives, and constantly resets the personal best vectors, resulting in the best performance on DTLZ1 as well as competitive scalability on DTLZ3. This is mainly effective because a context vector that already has a good value for one objective, can be optimized from the perspective of another objective, avoiding the stagnant global best problem to some degree.

5.2.5 The ZDT Test Suite

The IGD rankings for the ZDT test suite are provided in Table 5.8, whereas Table 5.9 lists the overall rankings based on aggregated scores.

One interesting observation was that MGPSO showed promising scalability on ZDT1 to ZDT3, outperforming both LMOCSO and D-IBEA. Moreover, CCMGPSO had the best performance on all experiments involving ZDT1 to ZDT3, only failing to statistically outperform WOF-NSGAII on the 1000-objective ZDT3 despite having the best mean over 30 runs.

CCMGPSO experienced a major performance drop on the multi-modal ZDT4, ranking third on the 1500- and 2000-dimensional ZDT4, having had the best performance for 1000 dimensions. LMOCSO was the best-performing algorithm on ZDT4, followed by WOF-NSGAII as the second-best performing one. Similar to the multi-modal DTLZ1 and DTLZ3, MGPSO did not show competitive performance on ZDT4 and it was outperformed by all algorithms except D-IBEA.

On the multi-modal ZDT6, CCMGPSO was the best-performing algorithm for 1000 and 1500 dimensions, but it was outperformed by WOF-NSGAII for 2000 dimensions. With reference to ZDT6, MGPSO was tied with LMOCSO for 1000 dimensions; however, it was outperformed by it for 1500 and 2000 dimensions.

5.2.6 Summary of Results on The ZDT Test Suite

The overall results on the ZDT suite (based on aggregated scores) are provided in Table 5.9.

Similar to the DTLZ suite, CCMGPSO had the best overall (aggregated) scores in all experiments on the ZDT suite as well, with WOF-NSGAII and MMOPSO finishing second and third respectively. Despite being outperformed by MGPSO on ZDT1 to ZDT3, LMOCSO obtained better overall scores than MGPSO as a result

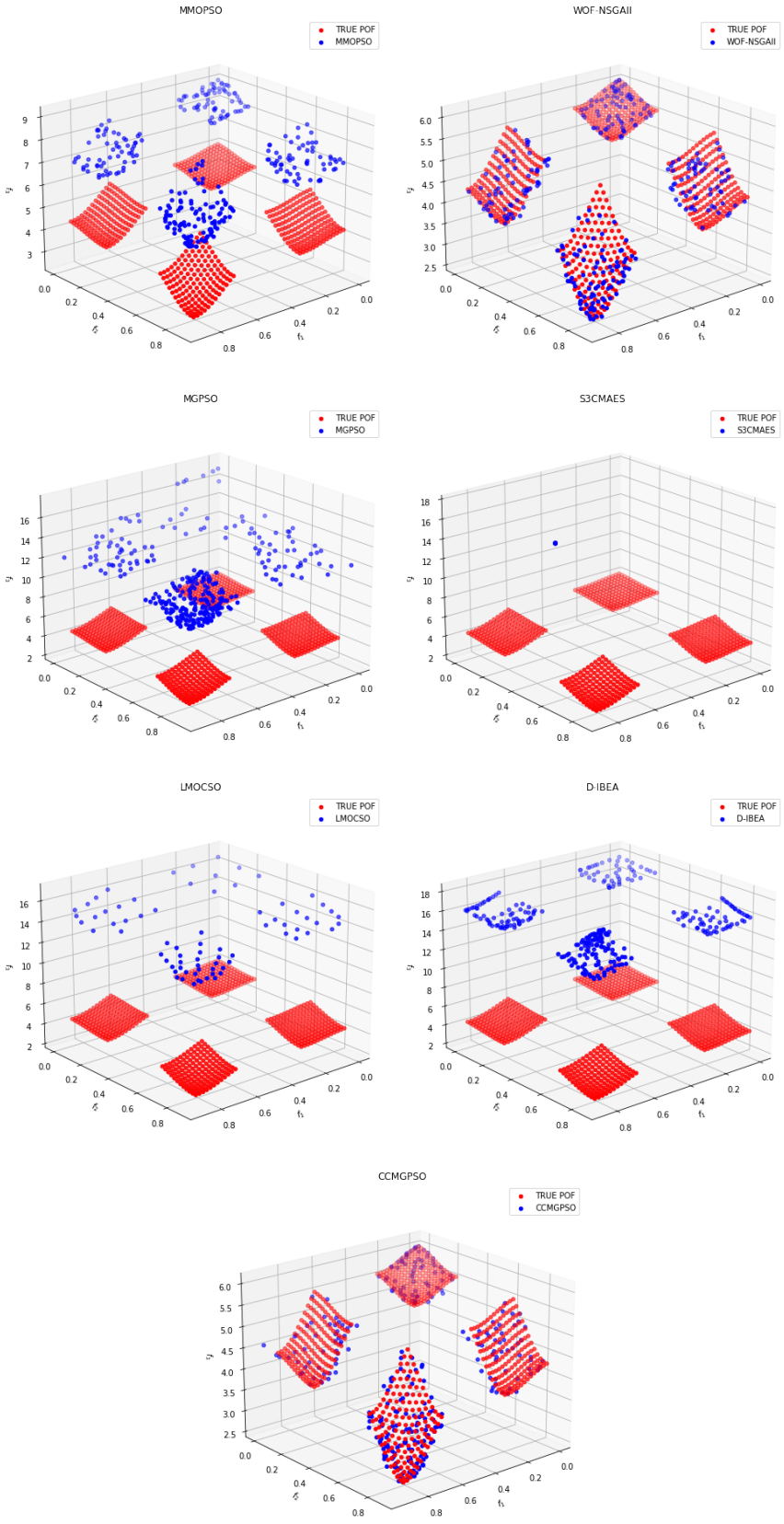

Figure 5.8: A visual comparison of the obtained POFs by the algorithms on the 2000-dimensional 3-objective DTLZ7.

Table 5.8: IGD rankings for ZDT1 to ZDT6. The highest-ranking algorithm is highlighted in grey background. If two or more algorithms were ranked first, all of them were marked using "≈" and the one with the best mean IGD score over 30 independent runs was highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

				Algorithm						
Problem	Dimensions	FEs	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
ZDT1 (**S**, **U**)	1000	3.00e+5	Difference	2	4	0	-6	-2	-4	6
			Rank	3	2	4	7	5	6	1
	1500	4.50e+5	Difference	2	4	0	-6	-2	-4	6
			Rank	3	2	4	7	5	6	1
	2000	4.50e+5	Difference	1	4	1	-6	-2	-4	6
			Rank	3	2	3	6	4	5	1
ZDT2 (**S**, **U**)	1000	3.00e+5	Difference	-1	4	2	-6	-1	-4	6
			Rank	4	2	3	6	4	5	1
	1500	4.50e+5	Difference	0	4	2	-6	-2	-4	6
			Rank	4	2	3	7	5	6	1
	2000	4.50e+5	Difference	0	4	2	-6	-2	-4	6
			Rank	4	2	3	7	5	6	1
ZDT3 (**S**, **U-M**)	1000	3.00e+5	Difference	2	5≈	0	-6	-2	-4	5≈
			Rank	2	1≈	3	6	4	5	1≈
	1500	4.50e+5	Difference	2	4	0	-6	-2	-4	6
			Rank	3	2	4	7	5	6	1
	2000	4.50e+5	Difference	2	4	0	-6	-2	-4	6
			Rank	3	2	4	7	5	6	1
ZDT4 (**S**, **U-M**)	1000	3.00e+5	Difference	-2	3	-4	-6	3	0	6
			Rank	4	2	5	6	2	3	1
	1500	4.50e+5	Difference	-2	4	-4	-6	6	0	2
			Rank	5	2	6	7	1	4	3
	2000	4.50e+5	Difference	-2	4	-4	-6	6	0	2
			Rank	5	2	6	7	1	4	3
ZDT6 (**S**, **M**)	1000	3.00e+5	Difference	2	4	-1	-6	-1	-4	6
			Rank	3	2	4	6	4	5	1
	1500	4.50e+5	Difference	2	4	-2	-6	0	-4	6
			Rank	3	2	5	7	4	6	1
	2000	4.50e+5	Difference	2	6	-2	-6	0	-4	4
			Rank	3	1	5	7	4	6	2

(1)S: Separable, **NS**: Non-separable, **U**: Uni-modal, **M**: Multi-modal, **U-M**: Both uni-modal and multi-modal (on different objectives), and **D**: Deceptive.

Table 5.9: The overall performance of each algorithm on the ZDT test suite, based on the aggregated scores. The best-performing algorithms were highlighted in grey background. The ranking scheme is explained in Section 5.1.5.

			Algorithm						
Overall	Dimensions	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
ZDT1 TO ZDT6	1000	Difference	3	20	-3	-30	-3	-16	29
		Rank	3	2	4	6	4	5	1
	1500	Difference	4	20	-4	-30	0	-16	26
		Rank	3	2	5	7	4	6	1
	2000	Difference	3	22	-3	-30	0	-16	24
		Rank	3	2	5	7	4	6	1

of its superiority on ZDT6 and the big difference in performance between them on ZDT4.

Just like the DTLZ suite, D-IBEA did not perform well on the ZDT suite. In fact, D-IBEA's performance on the 1000-dimensional ZDT4 (where it finished third behind CCMGPSO and LMOCSO) was its best performance on an individual ZDT problem. MGPSO managed to outperform both D-IBEA and LMOCSO on ZDT1 to ZDT3, but due to its poor performance on ZDT4 and ZDT6, it accumulated a worse overall score than LMOCSO.

Chapter 6

Conclusion & Future Work

This thesis explored the vast area of large-scale multi-objective optimization, and the difficulties associated with it. First, a literature review of some of the past and recent proposed approaches for large-scale single- and multi-objective optimization was provided. This included a brief history of particle swarm optimization (PSO) and its applications to multi-objective optimization and large-scale single-objective optimization. Then, some evolutionary approaches for large-scale multi-objective optimization were discussed. These approaches used different concepts for scalable performance. Some used competitions between individuals, some used novel reproduction approaches, while others proposed problem transformation methodologies.

Next, a scalability study of five well-known PSO-based algorithms for multi-objective optimization was conducted. More specifically, optimized multi-objective particle swarm optimization (OMOPSO), speed-constrained multi-objective particle swarm optimization (SMPSO), multi-objective particle swarm optimization with multiple search strategies (MMOPSO), multi-guide particle swarm optimization (MGPSO), and competitive mechanism-based multi-objective particle swarm optimization (CMOPSO) were compared with each other on the Walking Fish Group (WFG) test suite for 24, 50, 100, 500, 1000 dimensions, and two and three objectives. MMOPSO and SMPSO showed the best scalability in this study, with the latter specifically performing well on multi-modal problems. MGPSO and CMOSPO showed the worst scalability, despite MGPSO being off to a good start on low-dimensional problems.

In order to address the shortcoming of MGPSO, a scalable MGPSO-based approach for large-scale multi-objective optimization, termed cooperative co-evolutionary multi-guide particle swarm optimization (CCMGPSO), was proposed. CCMGPSO drew inspiration from AM-CCPSO [72] in using multiple context vectors, and sharing them between different objectives in cycles of equal length. A small set of experiments

was conducted to study the effects of each part of the proposed CCMGPSO. For example, it was shown that using multiple context vectors and sharing them between objectives could specifically be of use in multi-modal problems.

Finally, an empirical study was done in order to compare CCMGPSO with six other algorithms from the literature, namely MGPSO [64] [63], MMOPSO [45], WOF-NSGAII [91], LMOCSO [75], S3-CMA-ES [6], and D-IBEA [24] on the WFG, ZDT, and DTLZ benchmark suites. CCMGPSO scaled exceptionally well on separable and uni-modal problems. However, it was outperformed on some instances including problems that were either multi-modal, deceptive, or non-separable. In fact, CCMGPSO's worst performance on an individual problem was observed on WFG9, a problem possessing all of the aforementioned properties simultaneously. Moreover, CCMGPSO was also outperformed on the 2-objective DTLZ4 due to the existence of bias, where in some runs all context vectors got stuck in an area of the POF towards which DTLZ4 was biased. However, with reference to overall scores, CCMGPSO was only outperformed once on the 3-objective 500-dimensional WFG problems. Therefore, it can be concluded that the proposed CCMGPSO is **highly competitive**.

In the final set of experiments, some interesting observations were also made regarding other algorithms. For example, on some instances of the DTLZ test suite, LMOCSO obtained fewer, more equally-spaced, and less-converged Pareto-optimal solutions compared with the better-performing algorithms. The hypothesis was made as to whether using a reference vector-based selection strategy in the absence of an external archive could potentially hurt the convergence in LMOCSO. Moreover, although D-IBEA's proposed offspring generation approach that was designed to evade local optima showed promising signs on some multi-modal and deceptive functions such as WFG2, WFG8, and DTLZ3, it did not obtain satisfactory convergence on most of the large-scale instances of uni-modal problems.

With reference to future work, different considerations can be made. First, more algorithms could be compared with CCMGPSO on more benchmark suites. Moreover, CCMGPSO's susceptibility to performance loss on multi-modal, non-separable, deceptive, or biased problems can be further studied in different ways such as using other benchmark suites or using the current ones with more decision variables. More studies can also be conducted on the order in which CCMGPSO optimizes different objectives. For example, in some problems such as ZDT3 or DTLZ7 the last objective is multi-modal with the rest being uni-modal. Future work could study the potential effects of optimizing different objectives in different orders on problems of such kind. Finally, CCMGPSO can be modified with other approaches for objective swapping.

For example, instead of allocating co-evolutionary cycles of equal length to all objectives, contribution-based approaches could be considered where an objective that has contributed more to the search would undergo co-evolution during each iteration.

Moreover, the experiments of Chapter 5 can be further expanded with more performance measures. As previously mentioned, the hypervolume measure (HV) was excluded from the experiments of Chapter 5 due to the difficulties of selecting an appropriate reference vector in the presence of a high variance in the obtained POFs by different algorithms. As potential future work, other different measures, based on which different algorithms are assessed from varying perspectives, can be included.

Finally, this thesis only considered two- and three-objective problems. Future studies could involve the analysis of the proposed CCMGPSO for large-scale many-objective problems that have four or more objectives. It is worth noting that on such problems, the performance of the proposed CCMGPSO could potentially drop due to the high consumption of the computational budget. Since in CCMGPSO the objectives of a problem are optimized in a successive manner (starting from the first objective), when the number of objectives is not limited to two or three, CCMGPSO has the potential to consume all its computational budget at the early stages of the optimization process. Therefore, for many-objective optimization other alternatives can be considered for the order in which the objectives of a problem are optimized.

Bibliography

[1] Salem F Adra and Peter J Fleming. Diversity management in evolutionary many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 15(2):183–195, 2010.

[2] Luis Miguel Antonio, Carlos A Coello Coello, Mario A Ramírez Morales, Silvia González Brambila, Josué Figueroa González, and Guadalupe Castillo Tapia. Coevolutionary operations for large scale multi-objective optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[3] Alireza Askarzadeh. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures*, 169:1–12, 2016.

[4] Azam Asilian Bidgoli, Sedigheh Mahdavi, Shahryar Rahnamayan, and Hessein Ebrahimpour-Komleh. Gde4: The generalized differential evolution with ordered mutation. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 101–113. Springer, 2019.

[5] Bin Cao, Jianwei Zhao, Yu Gu, Yingbiao Ling, and Xiaoliang Ma. Applying graph-based differential grouping for multiobjective large-scale optimization. *Swarm and Evolutionary Computation*, 53:100626, 2020.

[6] Huangke Chen, Ran Cheng, Jinming Wen, Haifeng Li, and Jian Weng. Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. *Information Sciences*, 509:457–469, 2020.

[7] Ran Cheng and Yaochu Jin. A competitive swarm optimizer for large scale optimization. *IEEE transactions on cybernetics*, 45(2):191–204, 2014.

[8] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.

[9] Shixin Cheng, Hao Zhan, Huiqin Yao, Huayu Fan, and Yan Liu. Large-scale many-objective particle swarm optimizer with fast convergence based on alpha-stable mutation and logistic function. *Applied Soft Computing*, 99:106947, 2021.

[10] Nicola Clayton and Nathan Emery. Corvid cognition. *Current biology*, 15(3):R80–R81, 2005.

[11] Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1):58–73, 2002.

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

[13] Yann Collette and Patrick Siarry. *Multiobjective optimization: principles and case studies*. Springer Science & Business Media, 2004.

[14] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary computation*, 7(3):205–230, 1999.

[15] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[16] Kalyanmoy Deb and Rituparna Datta. Hybrid evolutionary multi-objective optimization and analysis of machining operations. *Engineering Optimization*, 44(6):685–706, 2012.

[17] Kalyanmoy Deb and Himanshu 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*, 18(4):577–601, 2013.

[18] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[19] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 825–830. IEEE, 2002.

[20] Kyle Erwin and Andries Engelbrecht. Control parameter sensitivity analysis of the multi-guide particle swarm optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 22–29, 2019.

[21] Zi-Min Gu and Gai-Ge Wang. Improving nsga-iii algorithms with information feedback models for large-scale many-objective optimization. *Future Generation Computer Systems*, 107:49–69, 2020.

[22] Hüseyin Haklı and Harun Uğuz. A novel particle swarm optimization algorithm with levy flight. *Applied Soft Computing*, 23:333–345, 2014.

[23] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[24] Cheng He, Ran Cheng, and Danial Yazdani. Adaptive offspring generation for evolutionary large-scale multiobjective optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[25] Cheng He, Lianghao Li, Ye Tian, Xingyi Zhang, Ran Cheng, Yaochu Jin, and Xin Yao. Accelerating large-scale multiobjective optimization via problem reformulation. *IEEE Transactions on Evolutionary Computation*, 23(6):949–961, 2019.

[26] Mardé Helbig. *Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation*. PhD thesis, University of Pretoria, 2012.

[27] Hanan Hiba, Azam Asilian Bidgoli, Amin Ibrahim, and Shahryar Rahnamayan. Cgde3: An efficient center-based algorithm for solving large-scale multi-objective optimization problems. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 350–358. IEEE, 2019.

[28] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, pages 82–87. Ieee, 1994.

[29] Wang Hu, Gary G Yen, and Guangchun Luo. Many-objective particle swarm optimization using two-stage strategy and parallel cell coordinate system. *IEEE transactions on cybernetics*, 47(6):1446–1459, 2016.

[30] Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston. A scalable multi-objective test problem toolkit. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 280–295. Springer, 2005.

[31] Simon Huband, Philip Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[32] John D Hunter. Matplotlib: A 2d graphics environment. *IEEE Annals of the History of Computing*, 9(03):90–95, 2007.

[33] Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, 19(2):264–283, 2014.

[34] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. How to specify a reference point in hypervolume calculation for fair performance comparison. *Evolutionary computation*, 26(3):411–440, 2018.

[35] Hisao Ishibuchi, Yu Setoguchi, Hiroyuki Masuda, and Yusuke Nojima. Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169–190, 2016.

[36] Sampreeti Jena. *Multi-Objective Optimization of the Design Parameters of a Shell and Tube Type Heat Exchanger Based on Economic and Size Consideration.* PhD thesis, 2013.

[37] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[38] Saku Kukkonen and Jouni Lampinen. An extension of generalized differential evolution for multi-objective optimization with constraints. In *International Conference on Parallel Problem Solving from Nature*, pages 752–761. Springer, 2004.

[39] Saku Kukkonen and Jouni Lampinen. Gde3: The third evolution step of generalized differential evolution. In *2005 IEEE congress on evolutionary computation*, volume 1, pages 443–450. IEEE, 2005.

[40] Jouni Lampinen et al. De's selection rule for multiobjective optimization. *Lappeenranta University of Technology, Department of Information Technology, Tech. Rep*, pages 03–04, 2001.

[41] Bingdong Li, Ke Tang, Jinlong Li, and Xin Yao. Stochastic ranking algorithm for many-objective optimization based on multiple indicators. *IEEE Transactions on Evolutionary Computation*, 20(6):924–938, 2016.

[42] Miqing Li, Shengxiang Yang, and Xiaohui Liu. Shift-based density estimation for pareto-based algorithms in many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):348–365, 2013.

[43] Xiaodong Li and Xin Yao. Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In *2009 IEEE Congress on Evolutionary Computation*, pages 1546–1553. IEEE, 2009.

[44] Xiaodong Li and Xin Yao. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 16(2):210–224, 2011.

[45] Qiuzhen Lin, Jianqiang Li, Zhihua Du, Jianyong Chen, and Zhong Ming. A novel multi-objective particle swarm optimization with multiple search strategies. *European Journal of Operational Research*, 247(3):732–744, 2015.

[46] Qiuzhen Lin, Songbai Liu, Qingling Zhu, Chaoyu Tang, Ruizhen Song, Jianyong Chen, Carlos A Coello Coello, Ka-Chun Wong, and Jun Zhang. Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems. *IEEE Transactions on Evolutionary Computation*, 22(1):32–46, 2016.

[47] Xiaoliang Ma, Fang Liu, Yutao Qi, Xiaodong Wang, Lingling Li, Licheng Jiao, Minglei Yin, and Maoguo Gong. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Transactions on Evolutionary Computation*, 20(2):275–298, 2015.

[48] Justin Maltese, Beatrice Ombuki-Berman, and Andries Engelbrecht. Co-operative vector-evaluated particle swarm optimization for multi-objective optimization. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1294–1301. IEEE, 2015.

[49] Justin Maltese, Beatrice M Ombuki-Berman, and Andries P Engelbrecht. A scalability study of many-objective optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 22(1):79–96, 2016.

[50] Masuduzzaman and GP Rangaiah. Multi-objective optimization applications in chemical engineering. *Multi-Objective Optimization: Techniques and Applications in Chemical Engineering (With CD-ROM)*, pages 27–59, 2009.

[51] Patrick E McKnight and Julius Najab. Mann-whitney u test. *The Corsini encyclopedia of psychology*, pages 1–1, 2010.

[52] Sadip Midya, Asmita Roy, Koushik Majumder, and Santanu Phadikar. Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach. *Journal of Network and Computer Applications*, 103:58–84, 2018.

[53] Antonio J Nebro, Juan J Durillo, and Matthieu Vergne. Redesigning the jmetal multi-objective optimization framework. In *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation*, pages 1093–1100, 2015.

[54] Antonio J Nebro, Juan José Durillo, Jose Garcia-Nieto, CA Coello Coello, Francisco Luna, and Enrique Alba. Smpso: A new pso-based metaheuristic for multi-objective optimization. In *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pages 66–73. IEEE, 2009.

[55] Elre T Oldewage, Andries P Engelbrecht, and Christopher W Cleghorn. The merits of velocity clamping particle swarm optimisation in high dimensional spaces. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2017.

[56] Mohammad Nabi Omidvar, Borhan Kazimipour, Xiaodong Li, and Xin Yao. Cbcc3—a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3541–3548. IEEE, 2016.

[57] Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, and Xin Yao. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on evolutionary computation*, 18(3):378–393, 2013.

[58] Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1115–1122, 2011.

[59] Mohammad Nabi Omidvar, Ming Yang, Yi Mei, Xiaodong Li, and Xin Yao. Dg2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, 21(6):929–942, 2017.

[60] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer, 1994.

[61] Rizk M Rizk-Allah, Aboul Ella Hassanien, and Adam Slowik. Multi-objective orthogonal opposition-based crow search algorithm for large-scale multi-objective optimization. *Neural Computing and Applications*, pages 1–32, 2020.

[62] Enrique H Ruspini and Igor S Zwir. Automated qualitative description of measurements. In *IMTC/99. Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference (Cat. No. 99CH36309)*, volume 2, pages 1086–1091. IEEE, 1999.

[63] Christiaan Scheepers, Andries P Engelbrecht, and Christopher W Cleghorn. Multi-guide particle swarm optimization for multi-objective optimization: empirical and stability analysis. *Swarm Intelligence*, 13(3):245–276, 2019.

[64] Christiaan Scheepers et al. *Multi-guided particle swarm optimization: A multi-objective particle swarm optimizer*. PhD thesis, University of Pretoria, 2017.

[65] Haitham Seada and Kalyanmoy Deb. A unified evolutionary optimization procedure for single, multiple, and many objectives. *IEEE Transactions on Evolutionary Computation*, 20(3):358–369, 2015.

[66] Matias Sessarego, KR Dixon, DE Rival, and DH Wood. A hybrid multi-objective evolutionary algorithm for wind-turbine blade optimization. *Engineering Optimization*, 47(8):1043–1062, 2015.

[67] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, pages 69–73. IEEE, 1998.

[68] Margarita R Sierra and Carlos A Coello Coello. A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms. *Technical Report of CINVESTAV-IPN*, 2004.

[69] Margarita Reyes Sierra and Carlos A Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and-dominance. In *International conference on evolutionary multi-criterion optimization*, pages 505–519. Springer, 2005.

[70] Cian Steenkamp and Andries P. Engelbrecht. A scalability study of the multi-guide particle swarm optimization algorithm to many-objectives. *Swarm and Evolutionary Computation*, 2021.

[71] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[72] Ruo-Li Tang, Zhou Wu, and Yan-Jun Fang. Adaptive multi-context cooperatively coevolving particle swarm optimization for large-scale problems. *Soft Computing*, 21(16):4735–4754, 2017.

[73] Ma Guadalupe Castillo Tapia and Carlos A Coello Coello. Applications of multi-objective evolutionary algorithms in economics and finance: A survey. In *2007 IEEE Congress on Evolutionary Computation*, pages 532–539. IEEE, 2007.

[74] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017.

[75] Ye Tian, Xiutao Zheng, Xingyi Zhang, and Yaochu Jin. Efficient large-scale multiobjective optimization based on a competitive swarm optimizer. *IEEE Transactions on Cybernetics*, 50(8):3696–3708, 2019.

[76] Hamid R Tizhoosh. Opposition-based learning: a new scheme for machine intelligence. In *International conference on computational intelligence for modelling,*

control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06), volume 1, pages 695–701. IEEE, 2005.

[77] Frans Van den Bergh and Andries Petrus Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE transactions on evolutionary computation*, 8(3):225–239, 2004.

[78] David A Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING, 1999.

[79] Tobias Wagner, Nicola Beume, and Boris Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *International conference on evolutionary multi-criterion optimization*, pages 742–756. Springer, 2007.

[80] Gai-Ge Wang and Ying Tan. Improving metaheuristic algorithms with information feedback models. *IEEE transactions on cybernetics*, 49(2):542–555, 2017.

[81] Yong Wang, Zixing Cai, and Qingfu Zhang. Enhancing the search ability of differential evolution through orthogonal crossover. *Information Sciences*, 185(1):153–177, 2012.

[82] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[83] Ming Yang, Mohammad Nabi Omidvar, Changhe Li, Xiaodong Li, Zhihua Cai, Borhan Kazimipour, and Xin Yao. Efficient resource allocation in cooperative coevolution for large-scale global optimization. *IEEE Transactions on Evolutionary Computation*, 21(4):493–505, 2016.

[84] Ming Yang, Aimin Zhou, Changhe Li, Jing Guan, and Xuesong Yan. Ccfr2: A more efficient cooperative co-evolutionary framework for large-scale global optimization. *Information Sciences*, 512:64–79, 2020.

[85] Zhenyu Yang, Ke Tang, and Xin Yao. Differential evolution for high-dimensional function optimization. In *2007 IEEE congress on evolutionary computation*, pages 3523–3530. IEEE, 2007.

[86] Zhenyu Yang, Ke Tang, and Xin Yao. Large scale evolutionary optimization using cooperative coevolution. *Information sciences*, 178(15):2985–2999, 2008.

[87] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[88] Xingyi Zhang, Ye Tian, Ran Cheng, and Yaochu Jin. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):97–112, 2016.

[89] Xingyi Zhang, Xiutao Zheng, Ran Cheng, Jianfeng Qiu, and Yaochu Jin. A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. *Information Sciences*, 427:63–76, 2018.

[90] Yin Zhang, Gai-Ge Wang, Keqin Li, Wei-Chang Yeh, Muwei Jian, and Junyu Dong. Enhancing moea/d with information feedback models for large-scale many-objective optimization. *Information Sciences*, 522:1–16, 2020.

[91] Heiner Zille, Hisao Ishibuchi, Sanaz Mostaghim, and Yusuke Nojima. A framework for large-scale multiobjective optimization based on problem transformation. *IEEE Transactions on Evolutionary Computation*, 22(2):260–275, 2017.

[92] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[93] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[94] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International conference on parallel problem solving from nature*, pages 832–842. Springer, 2004.

[95] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[96] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[97] Fernando Alonso Zotes and Matilde Santos Peñas. Particle swarm optimisation of interplanetary trajectories from earth to jupiter and saturn. *Engineering Applications of Artificial Intelligence*, 25(1):189–199, 2012.

Appendix A

Benchmark Suites

This section fully discusses three well-known benchmark suites, namely the Zitzler-Deb-Thiele (ZDT) test suite [92], the Deb-Thiele-Laumanns-Zitzler (DTLZ) [19] test suite, and the Walking Fish Group (WFG) test suite [30] [31].

A.1 Position Variables and Distance Variables

Before discussing the test suites, it is necessary to cover the concept of position and distance variables. The existence of these variables in multi-objective problems is what makes them different from their single-objective counterparts. In fact, one reason that decision variable grouping algorithms for large-scale single-objective optimization (such as differential grouping (DG)-based approaches [57] [59]) might not work well on multi-objective problems is this property of the MOO problems. For example, the multi-objective evolutionary algorithm based on decision variable analyses (MOEA/DVA) [47] proposed a decision variable grouping scheme for large-scale multi-objective optimization. This proposed algorithm classifies the variables into three categories, namely position variables, distance variables and mixed variables. Based on their interactions, distance variables are further classified into subgroups.

Simply put, position variables are related to diversity, distance variables control convergence, while mixed variables affect both [5]. More specifically, a decision variable is:

- **position-related** if perturbing the variable generates solutions that are non-dominated with reference to each other,
- **distance-related** if the solutions generated by perturbing it are dominated one by one,

- **mixed** otherwise.

In a multi-objective optimization problem, the number of position-related parameters should be divisible by $n_m - 1$, where n_m is the number of objectives. In order to further illustrate this, the work in [5] used the following problem as an example:

$$\begin{cases} f_1(\mathbf{x}) = x_1 - \cos(2.2\pi x_2) + x_2(x_3 + x_4) \\ f_2(\mathbf{x}) = 1 - x_1 + \sin(2.2\pi x_2) + x_2(x_3 + x_4) \\ \text{s.t. } x_i \in [0,1], i = 1,2,3,4 \end{cases} \tag{A.1}$$

Since perturbing only x_1 generates solutions that are non-dominated with respect to each other, x_1 is a position-related variable. Perturbing x_3 or x_4 generates solutions that are dominated one by one, so these are distance-related variables. Finally, since perturbing x_2 affects both diversity and convergence, x_2 is a mixed variable.

A.2 Zitzler-Deb-Thiele Test Suite

Zitzler, Deb, and Thiele proposed six multi-objective problems in a test suite termed the Zitzler-Deb-Thiele (ZDT) test suite (ZDT1 to ZDT6) [92]. Since ZDT5 is a binary-encoded problem, it was excluded from this thesis and its experiments. All ZDT problems are of the following general form:

$$\begin{aligned} &\text{minimize} \quad f_1(\mathbf{x}) \\ &\text{minimize} \quad f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}), g(\mathbf{x})) \end{aligned} \tag{A.2}$$

The different ZDT functions differ in their definitions of $f_1(\mathbf{x})$, $h(\mathbf{x})$, and $g(\mathbf{x})$ as listed below:

A.2.1 ZDT1

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 \\ g(\mathbf{x}) &= 1 + \frac{9}{n_x - 1}\sum_{i=2}^{n_x} x_i \\ h(f_1, g) &= 1 - \sqrt{\frac{f_1}{g}}, x_i \in [0,1] \end{aligned} \tag{A.3}$$

A.2.2 ZDT2

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 \\ g(\mathbf{x}) &= 1 + 9\sum_{i=2}^{n_x} \frac{x_i}{n_x - 1} \\ h\left(f_1, g\right) &= 1 - \left(\frac{f_1}{g}\right)^2, x_i \in [0,1] \end{aligned} \tag{A.4}$$

A.2.3 ZDT3

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 \\ g(\mathbf{x}) &= 1 + 9\sum_{i=2}^{n_x} \frac{x_i}{n_x - 1} \\ h\left(f_1, g\right) &= 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right)\sin\left(10\pi f_1\right), x_i \in [0,1] \end{aligned} \tag{A.5}$$

A.2.4 ZDT4

$$\begin{aligned} f_1(\mathbf{x}) &= x_1 \\ g(\mathbf{x}) &= 1 + 10(n_x - 1) + \sum_{i=2}^{n_x} \left(x_i^2 - 10\cos\left(4\pi x_i\right)\right) \\ h\left(f_1, g\right) &= 1 - \sqrt{\frac{f_1}{g}}, x_1 \in [0,1], x_2, \ldots, x_{n_x} \in [-5,5] \end{aligned} \tag{A.6}$$

With 10^9 local bests, ZDT4 is a multi-modal and rather difficult problem to solve.

A.2.5 ZDT6

$$\begin{aligned} f_1(\mathbf{x}) &= 1 - e^{-4x_1}\sin^6\left(6\pi x_1\right) \\ g(\mathbf{x}) &= 1 + 9\left(\frac{\sum_{i=2}^{n_x} x_i}{n_x - 1}\right)^{\frac{1}{4}} \\ h\left(f_1, g\right) &= 1 - \left(\frac{f_1}{g}\right)^2, x_i \in [0,1] \end{aligned} \tag{A.7}$$

All ZDT problems have only one position variable. As mentioned in [31], the biggest disadvantages of the ZDT suite are having only two objectives, and only having one deceptive problem which is a binary-encoded one (ZDT5).

A.3 Deb-Thiele-Laumanns-Zitzler Test Suite

Looking to address the ZDT suite's weaknesses, Deb *et al.* [19] proposed the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite which is scalable to any number of objectives, as well as decision variables. The DTLZ suite has nine functions, DTLZ1 to DTLZ9, where DTLZ8 and DTLZ9 have side constraints and therefore have been excluded from this thesis. All decision variables in all DTLZ functions have a domain of $[0, 1]$.

A.3.1 DTLZ1

$$\begin{aligned} f_1 &= (1+g)0.5 \prod_{i=1}^{n_m-1} y_i \\ f_{m=2:n_m-1} &= (1+g)0.5 \left(\prod_{i=1}^{n_m-m} y_i \right) (1 - y_{n_m-m+1}) \\ f_{n_m} &= (1+g)0.5 \left(1 - y_1\right) \\ g &= 100 \left[k + \sum_{i=1}^{k} \left((z_i - 0.5)^2 - \cos\left(20\pi \left(z_i - 0.5\right)\right) \right) \right] \end{aligned} \tag{A.8}$$

A.3.2 DTLZ2

$$\begin{aligned} f_1 &= (1+g) \prod_{i=1}^{n_m-1} \cos\left(y_i \pi/2\right) \\ f_{m=2:n_m-1} &= (1+g) \left(\prod_{i=1}^{n_m-m} \cos\left(y_i \pi/2\right) \right) \sin\left(y_{n_m-m+1} \pi/2\right) \\ f_{n_m} &= (1+g) \sin\left(y_1 \pi/2\right) \\ g &= \sum_{i=1}^{k} \left(z_i - 0.5\right)^2 \end{aligned} \tag{A.9}$$

A.3.3 DTLZ3

Similar to DTLZ2, the only difference being the use of DTLZ1's g function.

A.3.4 DTLZ4

Similar to DTLZ2, the only difference being the use of y_i^α, $\alpha > 0$ instead of y_i^α.

A.3.5 DTLZ5

Similar to DTLZ2, the only difference being the replacement of $y_2, \ldots, y_{n_m-1} \in \mathbf{y}$ by $\frac{1+2gy_i}{2(1+g)}$.

A.3.6 DTLZ6

Similar to DTLZ5, the only difference being the replacement of g by $g = \sum_{i=1}^{k} z_i^{0.1}$.

A.3.7 DTLZ7

$$\begin{aligned} f_{m=1:n_m-1} &= y_m \\ f_{n_m} &= (1+g)\left(M - \sum_{i=1}^{n_m-1}\left[\frac{f_i}{1+g}\left(1+\sin\left(3\pi f_i\right)\right)\right]\right) \\ g &= 1 + 9\sum_{i=1}^{k} z_i/k \end{aligned} \tag{A.10}$$

where $k = n_x - (n_m - 1)$ is the number of **distance-related** variables.

A.4 Walking Fish Group Test Suite

Despite the DTLZ problems being scalable both objective- and dimension-wise, the number of position variables in them is fixed $(n_m - 1)$. Therefore, Huband *et al.* [30] proposed the walking fish group (WFG) test suite. This suite, which consists of nine functions (WFG1 to WFG9), offers a greater level of customizability, allowing the users to change the number of position variables and consequently the difficulty of the problems. The WFG problems are all of the following general format:

$$\begin{aligned} \text{given } \mathbf{z} &= \{z_1, \ldots, z_k, z_{k+1}, \ldots, z_{n_x}\} \\ \text{minimize } f_m(\mathbf{x}) &= Dx_M + S_m h_m\left(x_1, \ldots, x_{n_m-1}\right) \forall m \in [1, n_m] \\ \text{where } \mathbf{x} &= \{x_1, \ldots, x_{n_m}\} \\ &= \left\{\max\left(t^p_{n_m}, A_1\right)\left(t^p_1 - 0.5\right) + 0.5, \ldots,\right. \\ &\quad \left.\max\left(t^p_{n_m}, A_{n_m-1}\right)\left(t^p_{n_m-1} - 0.5\right) + 0.5, t^p_{n_m}\right\} \\ \mathbf{t}^p &= \left\{t^p_1, \ldots, t^p_{n_m}\right\} \leftarrow \mathbf{t}^{p-1} \leftarrow \ldots \leftrightarrow \mathbf{t}^1 \leftarrow \mathbf{z}_{[0,1]} \\ \mathbf{z}_{[0,1]} &= \left\{z_{1,[0,1]}, \ldots, z_{n_x,[0,1]}\right\} \\ &= \left\{\frac{z_1}{z_{1,\max}}, \ldots, \frac{z_{n_x}}{z_{n_x,\max}}\right\} \end{aligned} \tag{A.11}$$

where:

- n_m is the number of objectives,
- $\mathbf{x}$ is a set of parameters, where $\{x_1, \ldots, x_{n_m-1}\}$ and x_{n_m} are position and distance parameters respectively,
- k is the number of position parameters, l is the number of distance parameters, such that $k + l = n_x \geq n_m$,
- Consequently, $\mathbf{z}$ is a set of $k + l = n_x$ working parameters;
- $D > 0$ is a scaling constant, $A_{1:n_m-1} \in \{0, 1\}$ and $S_{1:n_m} > 0$ are degeneracy and shape constants respectively. The number of the dimensions of the POF is decreased by one for each $A_i = 0$,
- $h_{1:n_m}$ are shape functions,
- and $\mathbf{t}^{1:p}$ is a set of transition vectors. The symbol "$\leftarrow$" represents the creation of each transition vector from another one using transformation functions.

Appendix B

Supplementary Experiments on CCMGPSO

This section provides the details of some supplementary CCMGPSO experiments that were conducted to further study the different parts of this proposed approach.

B.1 Multiple Context Vectors: The Seesaw Effect

One problem with the original MGPSO implementation is the use of individual objective values to update the personal best and the global best position vectors. In single-objective minimization problems where the ultimate goal is to minimize one objective, this is usually effective. However, in multi-objective optimization, this may not always be advantageous. For example, in the 2-objective WFG4 the objective vector $(0, 4)$ is a boundary point of the true front. However, if a particle in the first subswarm (that is optimizing the first objective) finds the objective vector $(0, 7)$, it will never update its personal best position afterwards. Moreover, the global best position in that subswarm will also remain unchanged until the end of the optimization process because 0 is the best possible value for the first objective. Despite the archive guide being proposed to address this issue, on multi-modal problems it might still not be enough for competitive performance. In this section, the motivation behind multiple context vectors alongside objective switching is discussed through some experimental data. The main inspiration for using multiple context vectors was drawn from AM-CCPSO [72], where multiple context vectors were used for large-scale single-objective optimization. Moreover, Maltese *et al.* [48] proposed CVEPSO by incorporating the vector-evaluated particle swarm optimization (VEPSO) [26] into the cooperative framework. Although the work in [48] used separate context vectors

per objective, the context vectors were shared between different subswarms through a knowledge transfer strategy (KTS), enabling the different objectives of a problem to share context vectors.

B.1.1 Experimental Setup and Results

In this section, a small study is conducted to analyze the effects of using multiple context vectors alongside objective swapping. For this study, the 1500-dimensional DTLZ problems were used. Two experiments were conducted on each test instance, the first experiment used n_m context vectors and each objective was tied[1] to one context vector. The second experiment used n_m context vectors where each objective had access to all context vectors by choosing between them randomly. Each experiment was run 20 times on each test instance, the number of maximum function evaluations was set to 4.5×10^5. The total number of particles and the archive size were both set to 300 in all experiments, and these 300 particles were split between the subswarms in the following way:

- 10 n_x-dimensional particles per objective,
- $k = 280\ \frac{n_x}{280}$-dimensional particles for two objectives and $k = 270\ \frac{n_x}{270}$-dimensional particles for three objectives.

For each experiment, the number of iterations where the corresponding objective value of the context vector was equal to zero was recorded as a percentage of the total number of iterations. The goal was to show how long a simple cooperative MGPSO (with a fixed context vector per objective) would go without ever updating the context vectors, because the objective values in the DTLZ problems are all non-negative.

Results

The relevant results for 1500-dimensional 2- and 3-objective DTLZ functions are listed in Tables B.1 and B.2 respectively. Without any exceptions and on all test instances, CCMGPSO with dynamic context vectors (CCMGPSO **(2)**) had fewer iterations where the corresponding value of the context vector was equal to zero compared with CCMGPSO with fixed context vectors (CCMGPSO **(1)**). In the multi-modal DTLZ1

[1]Here, "tied" means when for each objective, a specific context vector is used. Even when the CPSO swaps objectives, the new objective will not randmoly choose its context vector; instead, it uses the one assigned to it at the very beginning.

this was more noticeable, CCMGPSO **(1)** on average spent 98.34% and 99.23% of the entire duration of the search without updating the context vectors for two and three objectives respectively. It should be noted that for CCMGPSO **(1)**, lines 13-17 in Algorithm 16 were also excluded to study the effect of not updating the context vectors. Performance wise, having big numbers in Tables B.1 and B.2 could have several impacts. On multi-modal problems, CCMGPSO **(1)** could get trapped in a local optimum, where the objective values are way worse than those of the true POF. On uni-modal problems, this impact could range from not being significant at all to achieving good objective values but clustered around specific areas of the POF, rather than obtaining a well-distributed one.

A few examples depicting the effects of using CCMGPSO **(1)** on the quality of the obtained fronts for DTLZ1 (multi-modal), DTLZ2 (uni-modal) and DTLZ7 (uni-modal on all objectives except the last one which is multi-modal) are provided in Figures B.1, B.2, and B.3 respectively. As seen in Figure B.1, the objective values obtained by CCMGPSO **(1)** were noticeably worse than those of CCMGPSO **(2)**, and a lot of the solutions were clustered in hyperplanes where individual objective values were close or equal to zero. Judging by Figure B.1 and the numbers in Table B.1 and Table B.2, one could surmise that, on DTLZ1, CCMGPSO **(1)** found n_m objective vectors where the value of at least one objective was equal to zero (with bad values for other objectives), set them as context vectors, and never updated them until the end of the optimization process. For DTLZ2 (Figure B.2) which is a uni-modal problem, CCMGPSO **(1)** obtained good objective values, but with a visibly inferior distribution of solutions. As seen in Figure B.2, a large set of the obtained solutions was clustered around the hyperplane where f_1 has a value close to zero. In the 3-objective DTLZ7, the third objective is multi-modal, and as seen in Figure B.3, CCMGPSO **(1)** had visibly worse distribution on the third objective compared with its counterpart.

Now the real reason behind smaller numbers for CCMGPSO **(2)** in Tables B.1 and B.2 is explained. Let $\hat{f}$ be an arbitrary bi-objective minimization problem, where the minimum possible value for each objective is zero. Moreover, suppose after γ CPSO iterations on the first objective, the objective vector of the first context vector ($\widehat{C_1}$) is equal to (0,2). Moreover, suppose in the next γ iterations $\widehat{C_1}$ is assigned to the second objective and C_1 is updated based on an improvement in the second objective, making $\widehat{C_1}$ equal to $(1, 1.5)$. Because $\widehat{C_{11}}$ is no longer equal to zero, more opportunities arise the next time C_1 is assigned to the first objective. As previously explained and seen in Tables B.1 and B.2, and Figure B.1, this could specifically be of use

Table B.1: The effect of multiple context vectors with objective swapping on the 2-objective 1500-dimensional DTLZ problems, the listed values are the mean (over 20 independent runs) percentage of the total number of iterations where the corresponding objective of the assigned context vector was equal to zero

	Experiment	
Problem (Two Objectives)	CCMGPSO **(1)**	CCMGPSO **(2)**
DTLZ1	98.34%	36.26%
DTLZ2	49.62%	16.13%
DTLZ3	49.33%	16.34%
DTLZ4	49.93%	11.93%
DTLZ5	49.86%	20.33%
DTLZ6	49.13%	11.93%
DTLZ7	49.06%	19.26%

(1) CCMGPSO n_m context vectors, where each objective only had access to one context vector,
(2)CCMGPSO n_m context vectors, where each objective had access to all context vectors by randomly choosing between them.

on multi-modal problems. In order to further address this issue, a dominance-based update mechanism is also incorporated into CCMGPSO (lines 13-17 in Algorithm 16). Inspired by the personal best update approach in OMOPSO [69], if the corresponding value of the new vector is equal to that of the context vector's, the context vector is replaced by the new solution if its objective vector does not dominate that of the new solution's. The possible effects of using this approach are studied in the next section.

B.2 Spending Computational Budget on Accurate Values: A Worthwhile Investment?

In a lot of PSO-based approaches for cooperative co-evolution, the personal best value is calculated at every iteration, to ensure the accuracy of results. To explain this further, the relevant lines from the pseudo-code of CPSO-S (Algorithm 1) are given in Algorithm 17. A closer look at Algorithm 17 reveals that CPSO-S uses four function evaluations for each particle per iteration, because function f is invoked four different times. From an implementation point of view, this number can be reduced to two if the values are saved inside temporary variables. Moreover, in a regular n_x-dimensional PSO, this number can be reduced to only one function evaluation per particle per iteration, if the personal best value (in addition to position) of each particle is saved.

Table B.2: The effect of multiple context vectors with objective swapping on the 3-objective 1500-dimensional DTLZ problems, the listed values are the mean (over 20 independant runs) percentage of the total number of iterations where the corresponding objective of the assigned context vector was equal to zero

	Experiment	
Problem (Three Objectives)	CCMGPSO (1)	CCMGPSO (2)
DTLZ1	99.23%	52.33%
DTLZ2	65.46%	18.86%
DTLZ3	66.13%	25.64%
DTLZ4	65.73%	23.26%
DTLZ5	33.13%	7.19 %
DTLZ6	32.93%	12.53%
DTLZ7	65.33%	28.26%

(1) CCMGPSO n_m context vectors, where each objective only had access to one context vector,
(2)CCMGPSO n_m context vectors, where each objective had access to all context vectors by randomly choosing between them.

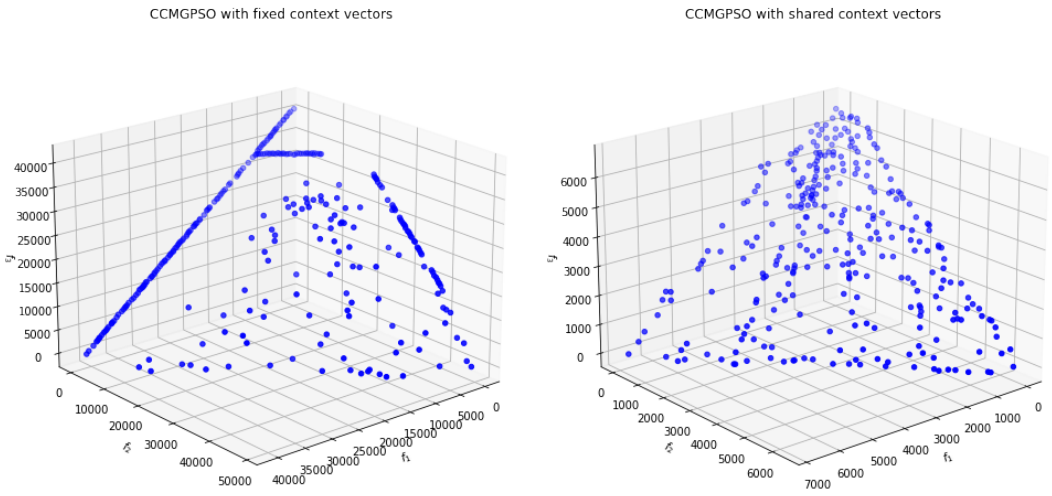

Figure B.1: The effect of sharing multiple context vectors between objectives on the 1500-dimensional 3-objective DTLZ1

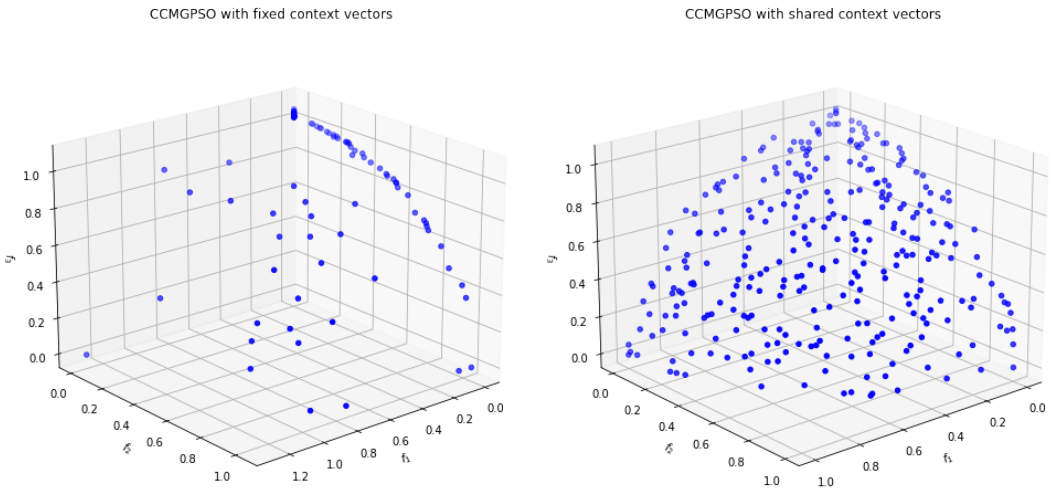

Figure B.2: The effect of sharing multiple context vectors between objectives on the 1500-dimensional 3-objective DTLZ2

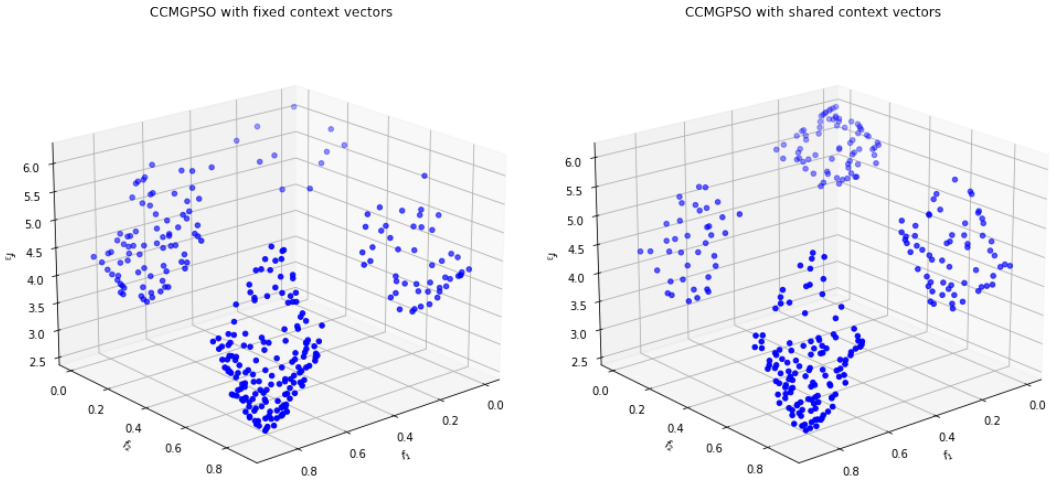

Figure B.3: The effect of sharing multiple context vectors between objectives on the 1500-dimensional 3-objective DTLZ7

However, in a $\frac{n_x}{k}$-dimensional context, keeping track of the personal best values to save computational budget without losing search accuracy is not possible. In a regular n_x-dimensional PSO, once the personal best value is updated, its value will always remain accurate because it is a result of evaluating a full n_x-dimensional decision vector. On the other hand, in $\frac{n_x}{k}$-dimensional PSO, the personal best position of each particle only represents a part of the context vector (a full-dimensional vector), and its objective value might change depending on what particles from other subswarms have done to the other parts of the context vector.

As mentioned above, a lot of PSO-based approaches for cooperative co-evolution, such as CCPSO [43], CCPSO2 [44], and AM-CCPSO [72], use the same approach regarding updating the personal best position of each particle. The question remains as to whether spending double the amount of computational budget on accurate values is always the best approach. A closer look at the snippet in Algorithm 17 and lines 6 to 9 in Algorithm 16 reveals that, CCMGPSO does not recalculate the personal best values at every iteration. In other words, CCMGPSO saves the personal best value and neglects the changes made to the other parts of the context vector (since the last $pBest$ update) for the sole purpose of saving computational budget. The effect of this approach is studied in the following section.

B.2.1 Experimental Setup and Results

In this section, a small set of experiments is conducted to study the effects of saving function evaluations per iteration by using less accurate personal best values.

Experimental Setup

In this study, the ZDT problems with 1000, 1500, and 2000 dimensions were used. Three experiments were conducted on each test instance, the first experiment used the accurate personal best update mechanism (Algorithm 17), and the second experiment used the less accurate personal best update approach but without the dominance-based update approach (lines 13-17 in Algorithm 16). Finally, the third experiment was based on the CCMGPSO as listed in the pseudo-codes and discussed in previous sections. Each experiment was run 30 times on each test instance, the number of maximum function evaluations was set to 3×10^5 and 4.5×10^5 for 1000 and more than 1000 decision variables respectively. The total number of particles and the archive size were both set to 300 in all experiments, and these 300 particles were split between the subswarms in the following way:

Algorithm 17 The personal best update of CPSO-S

```
1: procedure CPSO-S
2:     ⋮
3:     ⋮
4:     ⋮
5:     if f(b(j, P_j.x_i) < f(b(j, P_j.y_i) then
6:         P_j.y_i ← P_j.x_i
7:     end if
8:     if f(b(j, P_j.y_i) < f(b(j, P_j.ŷ) then
9:         P_j.ŷ ← P_j.y_i
10:    end if
11:    ⋮
12:    ⋮
13:    ⋮
14: end procedure
```

- 10 n_x-dimensional particles per objective,
- $k = 280$ $\frac{n_x}{280}$-dimensional particles.

Finally, the Mann-Whitney U test with a confidence level of 95% was used to check for statistical significance.

Results

The mean and standard deviation of the IGD values obtained by the aforementioned three experiments (over 30 independent runs) are listed in Table B.3. As seen in Table B.3, CCMGPSO **(3)** had better mean IGD values than both CCMGPSO **(1)** and CCMGPSO **(2)**. For CCMGPSO **(1)**, this difference in performance was always statistically significant. CCMGPSO **(2)** only had statistically similar results to CCMGPSO **(3)** on the 1000- and 1500-dimensional ZDT6. These results could have several implications:

- The fact that CCMGPSO **(3)** and CCMGPSO **(2)** always had better IGD values than CCMGPSO **(1)** (where for the former this was always statistically significant) could imply that spending more computational budget only to obtain the most up-to-date values might not always be the best approach, at least when the computational budget is not infinite.
- CCMGPSO **(3)** statistically outperforming CCMGPSO **(2)** on most test instances could imply the aforementioned difference in designing context vector-

Table B.3: The effect of using less accurate personal best values to save computational budget, in terms of the mean and standard deviation values obtained for the IGD measure for three different experiments over 30 independent runs

		Experiment		
Problem	Dimension	CCMGPSO **(1)**	CCMGPSO **(2)**	CCMGPSO **(3)**
ZDT1	1000	1.02E-4(3.24e-05)-	5.47E-05(6.49E-06)-	4.93E-5(3.37E-06)
	1500	6.77E-05(1.40E-05)-	4.70E-05(1.38E-06)-	4.60E-05(8.39E-07)
	2000	8.31E-05(1.98E-05)-	5.11E-05(5.22E-06)-	4.68E-05(1.39E-06)
ZDT2	1000	1.53E-4(4.81E-05)-	6.41E-05(1.05E-05)-	5.42E-05(4.83E-06)
	1500	1.03E-4(4.28E-05)-	5.71E-05(8.43E-06)-	5.12E-05(3.08E-06)
	2000	1.29E-4(4.19E-05)-	6.19E-05(1.02E-05)-	5.47E-05(8.69E-06)
ZDT3	1000	2.17E-4(3.69E-05)-	1.28E-4(2.89E-05)-	8.06E-05(1.12E-05)
	1500	1.58E-4(2.79E-05)-	8.25E-05(1.21E-05)-	6.38E-05(3.94E-06)
	2000	1.79E-4(2.85E-05)-	1.09E-4(2.49E-05)-	6.83E-05(5.43E-06)
ZDT4	1000	6.62E+1(4.08E+0)-	4.95E+1(3.39E+0)-	3.02E+1(2.78E+0)
	1500	1.38E+2(5.11E+0)-	1.03E+2(6.23E+0)-	7.43E+1(6.96E+0)
	2000	2.48E+2(1.17E+1)-	1.87E+2(1.18E+1)-	1.50E+2(1.26E+1)
ZDT6	1000	7.12E-4(1.83E-3)-	1.75E-4(4.72E-4)=	5.33E-05(2.37E-05)
	1500	3.91E-4(8.22E-4)-	3.17E-4(1.15E-3)=	4.43E-05(1.29E-05)
	2000	1.43E-3(2.58E-3)-	4.41E-4(1.10E-3)-	1.17E-4(3.82E-4)

(1) CCMGPSO with accurate personal best calculations (more function evaluations per iteration).
(2) CCMGPSO with less accurate personal best calculations (fewer function evaluations per iteration) but without the dominance-based update mechanism (lines 13-17 in Algorithm 16).
(3) CCMGPSO with less accurate personal best calculations (fewer function evaluations per iteration) and the dominance-based update mechanism
The experiment with the best mean IGD over 30 independent runs is highlighted in grey. "=" means statistically similar to CCMGPSO **(3)**, "-" means statistically worse than CCMGPSO **(3)**, and "+" means statistically better than CCMGPSO **(3)**

based approaches for single- and multi-objective optimization. As mentioned in previous sections, updating the context vector based on only one objective value might lead to performance loss especially on multi-modal problems. Therefore, in CCMGPSO, when the corresponding objective value of the context vector and the particle are equal, the context vector is replaced if its objective vector does not dominate that of the particle's. Inspired by OMOPSO's [69] personal best update approach, the motivation behind this approach is to consider objective vectors that do not update the assigned objective value, but have valuable improvements for other objectives.

Therefore, in CCMGPSO personal best values are not recalculated at every iteration to save computational budget. It may be worth noting that CCMGPSO **(1)** and CCMGPSO **(3)** could have statistically similar results under a computational budget which is large enough, but in this proposed version $pBest$ is kept track of at all times since the condition of having an infinite computational budget is not always satisfied.