Cooperative Coevolutionary Multi-Guide Particle Swarm Optimization Algorithm for Large-Scale Multi-Objective Optimization Problems

Amirali Madani[a], Andries Engelbrecht[b], Beatrice Ombuki-Berman[a]

[a]*Department of Computer Science, Brock University, St. Catharines, Ontario, Canada*
[b]*Department of Industrial Engineering and Computer Science Division, Stellenbosch University, Stellenbosch, South Africa*

Abstract

Many problems that are encountered in real-life applications consist of two or three conflicting objectives and many decision variables. Multi-guide particle swarm optimization (MGPSO) is a novel meta-heuristic 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 (and even many-objective) problems. However, a recent study has shown that MGPSO does not scale well when the number of decision variables is increased. This paper proposes a new scalable MGPSO-based algorithm, termed cooperative coevolutionary multi-guide particle swarm optimization (CCMGPSO), that incorporates ideas from cooperative coevolution (CC). CCMGPSO uses new techniques to spend less computational budget by periodically assigning only one CC-based subswarm to each objective (as opposed to using numerous CC-based subswarms). A detailed empirical study on well-known benchmark problems comparing the CCMGPSO with various state-of-the-art large-scale multi-objective optimization algorithms is done. Results show that the proposed CCMGPSO is highly competitive for high-dimensional problems with reference to the inverted generational distance (IGD) metric.

Keywords: Particle Swarm Optimization, Multi-Objective Optimization, Multi-Objective Evolutionary Algorithms, Large-Scale Multi-Objective Optimization, Cooperative Coevolution, Multi-Guide Particle Swarm Optimization

1. Introduction

A vast majority of optimization problems consist of more than one objective [1, 2]. 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 [3]. If an optimization problem has more than three objectives, it is referred to as a many-objective optimization problem. Multi-objective optimization problems are often encountered in real life. As an example, the work in [4] proposed an approach to optimize the (conflicting) objectives space mission cost and time as a multi-objective optimization problem. Midya *et al.* [5] 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 [6, 7, 3]. Multi-objective optimization has also been applied to the field of mechanics to minimize the equipment cost and the energy consumption [8, 9]. Generally speaking, MOOPs that have more than 100 decision variables are referred to as large-scale MOOPs [10, 11]. Multi-objective optimization evolutionary algorithms (MOEAs) typically face many difficulties (explained momentarily) 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 [12]. In fact, it has been shown that the majority of MOEAs, even those specifically proposed for large-scale problems, are incapable of solving large-scale MOOPs efficiently [10, 11, 13].
The multi-guide particle swarm optimization (MGPSO) algorithm [14, 15] is a new approach based on the particle swarm optimization algorithm (PSO) [16] for multi-objective optimization. In MGPSO, each objective is allocated a separate single-objective PSO and is optimized independently, meaning that the neighborhood best and personal best positions in each sub-swarm refer to the best values found for that specific objective. Different sub-swarms optimizing different objectives interact with each other using an archive of non-dominated solutions. MGPSO is similar to the coevolutionary multi-swarm PSO (CMPSO) [17] in the sense that they both optimize different objectives separately. However, they differ mostly in their archive maintenance mechanisms. For example, CMPSO uses random selection for selecting archive solutions, but MGPSO uses tournament selection. Furthermore,

CMPSO also continuously updates its archive using the elitist learning strategy (ELS).
In its original papers [14, 15], MGPSO had shown competitive performance when tested on low-dimensional multi-objective problems. Recently, Steenkamp and Engelbrecht [18] studied the scalability of MGPSO to many objectives and observed competitive performance from MGPSO when compared with other state-of-the-art approaches for many-objective optimization. Furthermore, MGPSO showed competitive performance when adapted for dynamic multi-objective problems [19]. Madani *et al.* [20] put its decision space scalability to the test by comparing MGPSO with four well-known PSO-based approaches for multi-objective optimization, namely optimized multi-objective particle swarm optimization (OMOPSO) [21], speed-constrained multi-objective particle swarm optimization (SMPSO) [22], multi-objective particle swarm optimization with multiple search strategies (MMOPSO) [23], and competitive mechanism-based multi-objective particle swarm optimization (CMOPSO) [24]. It was observed that, despite having competitive performance on low-dimensional problems, MGPSO scaled poorly compared for large-scale problems with the aforementioned approaches. More specifically, MGPSO and CMOPSO constantly had the worst scalability among the five studied algorithms.
Cooperative coevolution (CC) was first developed for genetic algorithms (GAs) [25] and shown to efficiently scale GAs to large-scale single-objective problems. Furthermore, CC has also been applied to PSO [26, 27, 28, 29] and differential evolution (DE) [30], and it has shown competitive performance in solving large-scale optimization problems. Moreover, Oldewage *et al.* [12] observed that PSO loses performance for large-scale problems since most particles leave the search space shortly after the start of the search which results in increased swarm diversity. Oldewage *et al.* [12] concluded that, for solving large-scale problems using PSO, the focus should shift to exploitation and not exploration. CC-based approaches solve a large-scale problem by partitioning it into many smaller sub-problems with fewer decision variables. This results in the exploitation of smaller sub-regions of the search space by the different sub-swarms, thereby offering a viable solution to the particle roaming behavior as observed by Oldewage *et al.* [12]. Therefore, in this paper, the CC decomposition approach is used to scale MGPSO to large-scale multi-objective problems.
This paper makes the following contributions:

- A detailed literature review covering different topics including some recent advances in large-scale multi-objective optimization algorithms, cooperative coevolution (CC) and its application to PSO, and other topics relevant to CC and large-scale optimization.

- A new CC-based MGPSO variant, termed cooperative coevolutionary multi-guide particle swarm optimization (CCMGPSO), which draws inspiration from previous CC-based approaches such as adaptive multi-context cooperative coevolving PSO (AM-CCPSO) [28] and hybrid cooperative PSO (CPSO-H_k) [29].

- A comparative empirical study involving CCMGPSO, and six state-of-the art algorithms, including the best-performing algorithm from [20], MGPSO itself [14, 15], and four state-of-the-art algorithms from the literature designed specifically for solving large-scale multi-objective optimization problems. It is observed that the CCMGPSO is highly competitive.

The remainder of this paper is organized as follows: Section 2 covers the necessary background information for understanding the context of this paper, including the basics of particle swarm optimization (PSO), multi-guide particle swarm optimization (MGPSO), and finally cooperative coevolution (CC) and its application to PSO. Section 3 covers related work including relevant approaches for large-scale single- and multi-objective optimization and other advances in CC. A detailed discussion of the proposed approach is provided in Section 4. Section 5 includes the experimental studies comparing the proposed CCMGPSO with six other state-of-the-art approaches for multi-objective optimization. In Section 7, the strengths and weaknesses of the proposed CCMGPSO are discussed. Section 8 presents the concluding remarks of this work along with some potential avenues of future work. Finally, Appendix A presents a small set of supplementary experiments designed to justify some of the design decisions made for the proposed CCMGPSO.

2. Background

This section covers the necessary background information for understanding the context of this paper including the basics of particle swarm optimization (PSO), multi-guide particle swarm optimization (MGPSO), and finally cooperative coevolution (CC) and its application to PSO.

2.1. Particle Swarm Optimization

Particle swarm optimization, introduced by Kennedy and Eberhart in 1995 [16], 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 particles are randomly initialized with n_x-dimensional decision vectors. Each particle contains a memory of a *personal best* position which is the best position found by the particle itself throughout the search process. Additionally, each particle remembers a *neighborhood best* position which is the best position found by all particles in its neighborhood throughout the search process. Using these personal and neighborhood best guides, each particle updates its velocity and position after each iteration. Shi and Eberhart [31], 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:

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

where t is the time step (the number of the current 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 a uniform distribution in $[0,1]^{n_x}$, 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. The amount of influence that the current velocity, the personal best position and the neighborhood best position have on the next velocity is controlled by ω, c_1, and c_2 respectively.
After updating the velocity, the position is updated using

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2}$$

2.2. Multi-Guide Particle Swarm Optimization

The multi-guide particle swarm optimization (MGPSO) algorithm [14, 15] is a new PSO-based approach for multi-objective optimization. In MGPSO each objective has its own separate sub-swarm and is optimized independently, meaning that the neighborhood best and personal best positions in each sub-swarm refer to the best values found for that specific objective. Because of this independence, a new guide, namely the archive guide, is introduced (hence the term multi-guide) and is used to establish communication about different objectives between different sub-swarms through a bounded archive of Pareto-optimal solutions. As a result, the velocity update equation is

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

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 [32] value wins the tournament. Positive coefficient c_3 controls the influence of the archive guide. The values of λ and $\mathbf{r}_3$ are chosen randomly from uniform distributions in $[0, 1]$ 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. However, other strategies for initializing and the updating λ have been investigated in [33].

2.3. Cooperative Coevolutionary Approaches

The idea of cooperative coevolutionary evolutionary algorithms (CCEAs) was first proposed by Potter and De Jong [25] in the form of a new algorithm termed the cooperative coevolutionary genetic algorithm (CCGA). The CCGA divides an n_x-dimensional problem to n_x one-dimensional problems, where each problem is optimized by a genetic algorithm (GA) sub-population. The fitness of each one-dimensional individual is calculated by forming an n_x-dimensional vector using the individual's decision vector value and other values from other sub-populations. Van den Bergh and Engelbrecht [29] proposed the first CC approach to PSO, namely cooperative particle swarm optimization (CPSO). The CPSO aims to address the shortcomings (explained momentarily) of stochastic optimization algorithms (such as PSO) in solving large-scale problems. The aforementioned shortcomings are caused by a phenomenon referred to as the *curse of dimensionality*. Since in these algorithms decision vectors are evaluated once all decision variables are updated,

Algorithm 1 CPSO-S[29]

```
1: procedure CPSO-S
2:     Initialize $n_x$ one-dimensional swarms
3:     Randomly initialize an $n_x$-dimensional context vector
4:     Define

   $\mathrm{b(j,z)}=(\mathrm{S}_1.\hat{\mathbf{y}}, S_2.\hat{\mathbf{y}}, \ldots, S_{j-1}.\hat{\mathbf{y}}, z, S_{j+1}.\hat{\mathbf{y}}, \ldots, S_{n_x}.\hat{\mathbf{y}})$

5:     for each sub-swarm $S_\ell$ $\ell = 1, \ldots n_x$ do
6:         for each particle $S_\ell.\mathbf{x}_i$ $i = 1, \ldots, n_s$ do
7:             if $f(\mathbf{b}(\ell, S_\ell.\mathbf{x}_i) < f(\mathbf{b}(\ell, S_\ell.\mathbf{y}_i)$ then
8:                 $S_\ell.\mathbf{y}_i \leftarrow S_\ell.\mathbf{x}_i$
9:             end if
10:            if $f(\mathbf{b}(\ell, S_\ell.\mathbf{y}_i) < f(\mathbf{b}(\ell, S_\ell.\hat{\mathbf{y}})$ then
11:                $S_\ell.\hat{\mathbf{y}} \leftarrow S_\ell.\mathbf{y}_i$
12:            end if
13:        end for
14:        for each particle $S_\ell.\mathbf{x}_i$ $i = 1, \ldots, n_s$ do
15:            Update particle's velocity using Eq. (1)
16:            Update particle's position using Eq. (2)
17:        end for
18:    end for
19: end procedure
```

the optimizer can be deceived by an objective function 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 mechanism to optimize different decision variables separately. CPSO splits any given n_x-dimensional problem to n_x one-dimensional problems. Referred to as the split-CPSO (CPSO-S), this algorithm uses a context vector for evaluating the one-dimensional solutions. A context vector is an n_x-dimensional vector that is initialized randomly and later used to combine the best positions (explained momentarily) found by different sub-swarms. The context vector holds the best values found for each dimension by its corresponding sub-swarm, and is used to evaluate low-dimensional solutions by putting them in their corresponding dimensions of the context vector. The quality of the sub-swarm particle is that of the new context vector formed by swapping the sub-swarm particle into the original context vector. The context vector is then updated (replaced by the new vector) if this results in an improved fitness value. The pseudo-code of CPSO-S is presented in Algorithm 1. Function $\mathbf{b}(j, z)$ is used for evaluating the low-dimensional solutions. This function builds an n_x-dimensional vector using the global best vectors of all sub-swarms except the j-th one. For the j-th sub-swarm, the vector z (a prospective low-dimensional position vector) is used.

Van den Bergh and Engelbrecht [29] also proposed the hybrid CPSO (CPSO-H$_k$). 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 other. Before each iteration, one of the n_x-dimensional particles is replaced by the context vector.

Li and Yao [26] proposed cooperative coevolving particle swarm optimization (CCPSO) as an approach to improve CPSO-S$_k$ by constantly regrouping the decision variables. This is done in the hope 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 of the search, and the resulting groups are kept the same for the entirety of the optimization process. However, CCPSO regroups the decision variables before each iteration.

Li and Yao [27] later improved CCPSO by proposing CCPSO2. CCPSO2 no longer uses the adaptive weighting scheme as in CCPSO. Instead, CCPSO2 uses a different decision variable grouping approach in the form of dynamic values for k. Unlike the random grouping of CCPSO before every iteration, CCPSO2 only regroups the decision variables if the fitness value of the context vector has not improved after all particles in all groups have been submitted to the context vector. Each group has s decision variables assigned to it, forming $k = \frac{n_x}{s}$ groups of decision variables in total. For each random regrouping, the value of s is randomly selected from the set $S = \{2, 5, 50, 100, 200\}$. Using CCPSO2 as a starting point, Tang *et al.* [28] proposed adaptive multi-context cooperative coevolving PSO (AM-CCPSO). AM-CCPSO improved the performance of CCPSO2 for 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. AM-CCPSO uses a roulette selection scheme to choose a context vector from this set (which are all initialized randomly) based on their respective fitness values.

3. Related Work

Many approaches have been proposed for large-scale multi-objective optimization in recent years [34, 35, 11]. Some approaches are based on the classification of variables by studying the potential relationships between them. In a multi-objective optimization problem, the decision variables can be classified into two groups: position-related variables (affecting diversity) and distance-related variables (affecting convergence) [36]. Based on this classification of decision variables, Ma *et al.* [10] proposed the multi-objective evolutionary algorithm based on decision variable analyses (MOEA/DVA). In MOEA/DVA, mixed variables are added to the set of diversity-related variables. Inspired by MOEA/DVA, Zhang *et al.* [13] proposed the evolutionary algorithm for large-scale many-objective optimization (LMEA) framework using k-means to cluster the variables into the aforementioned groups. In order to adapt CMA-ES [37] for large-scale many-objective optimization, Chen *et al.* [38] proposed S3-CMA-ES.

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.* [11]. This algorithm is heavily inspired by the competitive swarm optimizer (CSO) [39], where competitions are used to improve the overall diversity by pairing different solutions together and making the worse solution learn from the better one. Cheng *et al.* [40] proposed the large-scale many-objective PSO based

on the alpha-stable mutation (LMPSO). LMPSO draws inspiration from previous work in [41], where Levy flights were successfully integrated into the PSO, enabling the particles to do long jumps and to escape premature convergence. LMPSO uses the alpha-stable distribution, which is a more general family that includes Levy flights. 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.
Furthermore, Hiba *et al.* [42] proposed a center-based mutation operator for the third generalized differential evolution (CGDE3) algorithm. This novel center-based mutation approach, which replaces the traditional GDE3 [43, 44, 45, 46] mutation operator 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. Some non-decomposition approaches for large-scale multi-objective optimization 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.* [34] proposed an adaptive offspring generation framework, termed DGEA. DGEA uses two kinds of direction vectors (used for guiding new solutions) for offspring generation, one for convergence and the other for diversity improvement. Experimental results in [34] showed competitive performance of the DGEA framework for large-scale problems, when compared with five state-of-the-art approaches. Rizk-Allah *et al.* [47] proposed the multi-objective orthogonal opposition-based crow search algorithm (M2O-CSA). This algorithm uses the same position update of the crow search algorithm (CSA) [48], 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 [49]) for improved performance on large-scale problems.

The CC-based approaches can be quite costly, especially if the problems are partitioned 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.* [50], proposed the operational decomposition (OD) framework which aims to improve the crossover operations in MOEAs that produce new offspring using crossover operators. 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. Moreover, in recent years contribution-based approaches [51, 52, 53, 54] have been developed to address the issue of the high consumption of computational budget. These contribution-based approaches define contribution metrics, and allocate more computational budget to sub-populations that have better scores. The main goal of such approaches is to dynamically allocate computational resources to sub-swarms that have contributed more to the search.

In recent years, work has also been done on information feedback models. In most meta-heuristic algorithms, a lot of search information from previous iterations is not considered when updating individuals. Recently, Wang and Tan [55] proposed the idea of using information feedback models (IFMs) in meta-heuristic algorithms to make use of potential useful information from previous iterations. More recently, Gu and Wang [56] incorporated IFM into NSGA-III [57] (another state-of-the-art reference-point-based approach for many-objective optimization proposed by Deb and Jain [57]) for large-scale many-objective optimization. For large-scale many-objective optimization using IFMs and MOEA/D [58], Zhang *et al.* [59] proposed the MOEA/D-IFM.

4. Cooperative Coevolutionary MGPSO

This section presents the cooperative coevolutionary multi-guide particle swarm optimization (CCMGPSO) algorithm which is a decomposition-based approach to the MGPSO algorithm.

4.1. Challenges and Motivation

In this section, some important challenges in solving large-scale multi-objective optimization problems and the motivation behind CCMGPSO to overcome them are discussed:

- MGPSO has outperformed the state-of-the-art algorithms for various types of problems, such as static multi-objective problems [14, 15], static many-objective problems [18], and dynamic problems [19]. However, Madani *et al.* [20] observed that the performance of MGPSO for multi-objective optimization quickly worsens when the number of dimensions is increased. Therefore, MGPSO was used as the starting point of this research.

- Oldewage *et al.* [12] observed that PSO-based algorithms become less efficient in solving large-scale problems since most particles leave the search space shortly after the start of the search which results in increased swarm diversity. Therefore, Oldewage *et al.* [12] stated that for solving large-scale problems using PSO, the focus should shift to exploitation rather than exploration. Furthermore, CC-based approaches solve a large-scale problem by partitioning it into many smaller sub-problems with fewer decision variables. This results in the exploitation of smaller sub-regions of the search space by the different sub-swarms. Therefore, in this paper the CC decomposition approach is used to scale MGPSO to large-scale multi-objective problems due to the focus on exploitation and its past success in scaling different meta-heuristics to large-scale problems [29, 26, 27, 28, 30, 25].

- The discussed CC-based approaches (in Section 2.3) are mostly used for solving single-objective problems. There are important distinctions between single-objective and multi-objective problems. For example, some solutions to a multi-objective problem might have excellent values for some objectives and poor values for the others. Therefore, using a single context vector (as defined in Section 2.3) might not result in the best trade-off between the objectives. To address this issue, the idea of using multiple context vectors and randomly assigning them to each objective is used in CCMGPSO. This idea is explained in greater detail in Section 4.2 and Section 4.3. Moreover, Appendix A provides some supplementary experiments that study the effects and benefits of using multiple context vectors.

- As discussed in Section 3, the CC decomposition approach can be costly with reference to the computational budget. Therefore, many contribution-based approaches have been proposed [51, 52, 53, 54] to allocate more budget to sub-problems that have contributed more to the objective optimization. Unlike the contribution-based approaches, CCMGPSO uses new techniques to spend less computational budget by periodically assigning only one CC-based subswarm to each objective (as opposed to using numerous CC-based subswarms). This component of CCMGPSO and its benefits are discussed extensively in Section 4.2.

4.2. Spending Less Computational Budget

Due to the success of the CC framework in scaling PSO [29, 26, 27, 28] to large-scale optimization problems and the success of the MGPSO in solving low-dimensional multi- [15, 14] and many-objective [18] optimization problems, the motivation behind CCMGPSO is to efficiently tackle large-scale multi-objective optimization problems, while preserving computational budget (explained momentarily). Since many approaches based on the CC framework use k dimension groups (and $\frac{n_x}{k}$-dimensional individuals) for optimizing the n_x-dimensional objective function, 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 in which too much of the computational budget is spent. 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 CCMGPSO, which is discussed in greater detail in the following sections, aims to share a single CPSO between different objectives in cycles of equal length. An example for an arbitrary large-scale 2-objective optimization problem with n_x decision variables is depicted in Figure 1. To solve the aforementioned large-scale problem, Figure 1 illustrates the following options:

1. MGPSO [14, 15] (Figure 1(a)) in which each objective is optimized separately using an n_x-dimensional PSO sub-swarm. Using this approach, the particles in each sub-swarm only optimize one objective and define their personal and global best positions with respect to that specific objective. Furthermore, all particles interact with an archive of non-dominated solutions by constantly sending and receiving non-dominated solutions. However, it was previously shown that the MGPSO has poor scalability on large-scale optimization problems [20].

2. To improve the performance and scalability of MGPSO, one idea is to combine CC-based approaches with MGPSO as these approaches have been shown to have competitive performance on large-scale optimization problems [26, 27, 28, 29]. A basic way of creating a CC-based variant of MGPSO is depicted in Figure 1(b), where n_x decision variables are classified into k groups. This basic variant assigns n_m (the number

of objectives) CPSO swarms to MGPSO such that each CPSO swarm (corresponding to one objective of the problem) has k sub-swarms. At the bottom level, each of these sub-swarms is a single-objective $\frac{n_x}{k}$-dimensional PSO. One major problem with this approach is its high consumption of the computational budget (i.e. the number of function evaluations). If each sub-swarm has n_s particles, a single iteration of this approach will use $\mathcal{O}(n_m\ k\ n_s)$ function evaluations.

3. To overcome the computational budget obstacle, the idea of a single CPSO swarm is proposed in the CCMGPSO (Figure 1(c)). In this approach, there is only one CPSO swarm (consisting of k $\frac{n_x}{k}$-dimensional sub-swarms) irrespective of the total number of objectives. More specifically, the single CPSO swarm is shared by different objectives in cycles of equal length (γ iterations per objective), where after every γ iterations the CPSO swarm changes its assigned objective. Furthermore, and inspired by AM-CCPSO [28], there is a pool of n_c context vectors in CCMGPSO. Every time the CPSO swarm changes objectives, it randomly picks one of the context vectors and modifies it for the next γ iterations based on improvements in the value of its assigned objective. If each sub-swarm has n_s particles, a single iteration of this approach will use $\mathcal{O}(k\ n_s)$ function evaluations which saves a factor of n_m compared with the basic approach (described above).

Additionally, the proposed CCMGPSO also uses n_m full-dimensional sub-swarms (one per objective). This is inspired by CPSO-H$_k$, where combining full-dimensional and $\frac{n_x}{k}$-dimensional particles resulted in competitive performance.

4.3. Design Details

The proposed CCMGPSO adds a single CPSO on top of the MGPSO (Figure 1), meaning that the CCMGPSO consists of n_x- and $\frac{n_x}{k}$-dimensional individuals, inspired by the CPSO-H$_k$ where combining full-dimensional and $\frac{n_x}{l}$-dimensional particles resulted in competitive performance [29]. The pseudocode for CCMGPSO is provided in Algorithm 2. At the beginning of the algorithm, CCMGPSO randomly initializes n_c context vectors in the decision space. Note that n_c is not necessarily equal to n_m. It is shown in the following sections that setting n_c to values greater than n_m resulted in improved performance for some problems. 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 $\widehat{\mathbf{C}}$ in Algorithm 2). Without storing the objective vectors, the algorithm would need to re-evaluate the context vector every time (refer to line 10 in Algorithm 4) even if its objective vector remained unchanged. Every run of the n_x-dimensional sub-swarms is followed by a single run of the CPSO (the $\frac{n_x}{k}$-dimensional sub-swarms) on the current objective o (explained momentarily). The current objective to be optimized (o) is controlled using a control parameter named iterations per objective (γ) (line 21 in Algorithm 2). For example, for a bi-objective optimization problem with $\gamma = 10$, in the first 10 iterations every run of the n_x-dimensional sub-swarms is followed by a run of the $\frac{n_x}{k}$-dimensional sub-swarms on the first objective. For the next 10 iterations, every run of the n_x-dimensional sub-swarms is followed by a run of the $\frac{n_x}{k}$-dimensional sub-swarms 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 re-initialized according to the new dimension indices (lines 14 to 16 in Algorithm 2). This is done by randomly partitioning the $\frac{n_x}{k}$ decision value indices $\{1, \ldots, n_x\}$ into k groups such that each group is assigned to exactly one CPSO sub-swarm. An example of this random regrouping procedure is depicted in Figure 2, where nine dimension indices are randomly assigned to three CPSO sub-swarms. Since each decision variable (represented by an index) is bounded between real lower and upper bounds, after the decision variables are regrouped the decision variables of each particle are also randomly re-initialized according to their assigned indices (and their respective bounds).

It is worth mentioning that when the CPSO switches objectives, the personal best objective function values of the $\frac{n_x}{k}$-dimensional particles are reset (set to $+\infty$) (line 18 in Algorithm 2). After γ iterations the current objective (o) is changed, and the personal best values of particles must be reset, because they refer to the previous objective. Moreover, the CPSO is randomly assigned a context vector from the pool of context vectors (line 19 in Algorithm 2) after being assigned a new objective o. The CPSO then holds onto this context vector for the next γ iterations, and it will update this context vector based on improvements in the assigned objective (Algorithm 4). Note that the provided pseudo-codes all assume that the underlying optimization problem to be a minimization problem.

4.4. Velocity and Position Update Equations

The CCMGPSO also utilizes the original MGPSO velocity and position update equations. Because there are n_c different context vectors that can each be seen as a global best position, for the particles belonging to the CPSO sub-swarms the global best position is extracted from the assigned context vector ($\mathbf{C}_l$ in Algorithm 2 and Algorithm 4). The archive guide for these low-dimensional particles is also selected from the archive using tournament selection and *trimmed* into the correct decision variables. When updating the CPSO particles, the trimming procedure is executed to extract the relevant subset of a larger vector. Note that the context vectors and the archive guides are all n_x-dimensional vectors, so for updating a given CPSO particle, the relevant $\frac{n_x}{k}$-dimensional sub-vectors must be extracted. An example of the archive guide implementation and the trimming procedure for the $\frac{n_x}{k}$-dimensional particles are depicted in Figure 2 through an arbitrary 9-dimensional problem whose decision variables are partitioned into three different groups.

Many PSO-based approaches for multi-objective optimization (such as SMPSO [22] and OMOPSO [21]) 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 for 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 [21] and MMOPSO [23]. Therefore, for c_3 in CCMGPSO the same range of possible random values was used.

Note that CCMGPSO particle position vectors are randomly re-initialized after each decision variable regrouping (lines 14 to 16 in Algorithm 2). The reasoning behind this re-initialization was derived from observations on some large-scale problems such as DTLZ6. It was observed that the values of some decision variables tend to become 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 many solutions are added to the archive on the basis of small modifications to the context vectors (line 19 in Algorithm 4), in these situations the archive (which is bounded with size n_A) would become replete with solutions that have the same values for the aforementioned decision variables. Therefore, some cases were observed in which the current position, the personal best position, the global best position, and the archive guide position vectors all had 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. A similar pattern (to CCMGPSO) was previously observed by Van den Bergh and Engelbrecht [60] for PSO where for a given particle, the current position, the personal best position, and the global best position had identical values which resulted in the particle stopping its movement over time if the velocity was close to zero. Hence, the random re-initialization of the position vectors generally showed effectiveness in helping the particles avoid these situations and was used in CCMGPSO. Note that the global best values remain unchanged, since for the CPSO particles they are defined based on the pool of n_c context vectors.

5. Empirical Process

This section presents an analysis of the CCMGPSO in comparison with six other multi-objective optimization algorithms from the literature evaluated on three well-known benchmark suites.

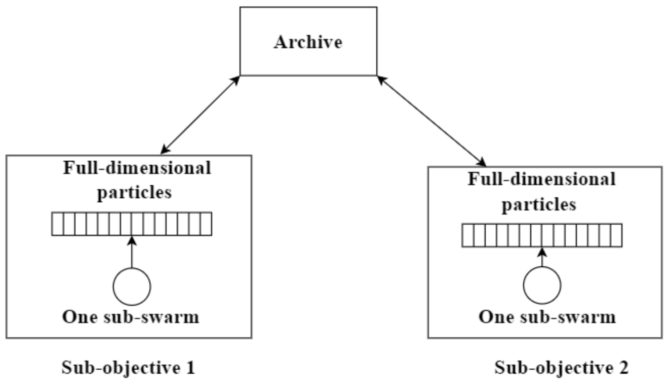
5.1. Implementation Details

The CCMGPSO was compared with MGPSO [14, 15] (without using the random parameters of CCMGPSO), MMOPSO [23] (the algorithm with the best scalability from the previous scalability study found in [20]), WOF-NSGAII [61], LMOCSO [11], S3-CMA-ES [38], and D-IBEA [34] (four approaches proposed for large-scale multi-objective optimization). All algorithms except WOF-NSGAII and S3-CMA-ES were implemented in Java. For WOF-NSGAII, the source code was obtained from the authors' website[1] and for S3-CMA-ES, the implementation in the PlatEMO framework [62] 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. For all algorithms, the benchmark suites of the jMetal [63] framework were used. For all algorithms, if the value of one dimension of the position vector became greater than the upper bound or less than the

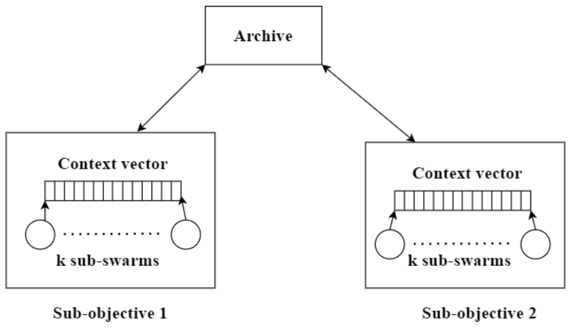
[1]https://www.ci.ovgu.de/Research/Codes.html

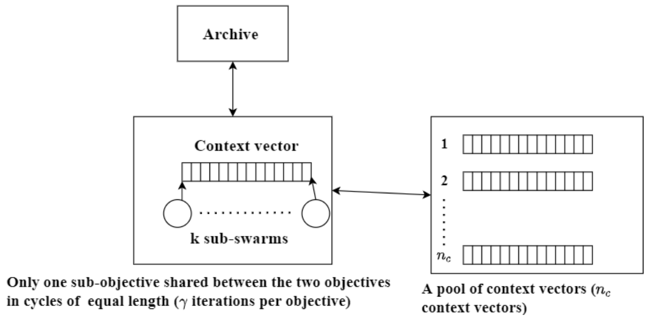
Algorithm 2 The pseudo-code of CCMGPSO

```
1: procedure CCMGPSO
2:     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), $\gamma$ (the number of CPSO iterations per objective), $n_T$ (the tournament size), $\mathbf{f}$ (the optimization problem)
3:     Output: non-dominated front ($\mathbf{F}$).
4:     Randomly initialize $n_c$ context vectors in the decision space in a set called $\mathbf{C}$;
5:     $\mathbf{P} \leftarrow$ Initialize $n_m$ sub-swarms each having $n_x$ decision variables;
6:     Initialize a crowding distance-based archive of size $n_A$;
7:     $l \leftarrow$ a randomly selected context vector index from the set $\mathbf{C}$;
8:     For each context vector $\mathbf{C}_{l'}$, $1 \leq l' \leq n_c$, initialize an objective vector $\widehat{\mathbf{C}}_{l'}$ of size $n_m$, set all objective values of $\widehat{\mathbf{C}}_{l'}$ to $+\infty$
9:     for $iteration = 0 : maxIterations - 1$ do
10:        for each objective $i = 1 : n_m$ do
11:            Randomly select a particle from $\mathbf{P}[i]$ (from line 5), replace it with $\mathbf{C}_l$;
12:            Optimize $i$-th objective in $\mathbf{P}[i]$ using MGPSO (using Eq. (2) and Eq. (3) and update the archive if necessary;
13:        end for
14:        if $iteration \% (n_m \times \gamma) = 0$ then
15:            $\mathbf{S} \leftarrow$ Randomly group the decision variables into $k$ groups, initialize sub-swarms with $\frac{n_x}{k}$ decision variables;
16:        end if
17:        if $iteration \% \gamma = 0$ then
18:            For all particles in the sub-swarms of $\mathbf{S}$, set the $pBest$ value to $+\infty$;
19:            $l \leftarrow$ a randomly selected context vector index from the set $\mathbf{C}$;
20:        end if
21:        $o \leftarrow (\lfloor \frac{iteration}{\gamma} \rfloor \% n_m) + 1$;    ▷ $o$ is the objective number that the CPSO is currently optimizing
22:        Optimize $o$ using $\mathbf{C}_l$ and the sub-swarms in $\mathbf{S}$, update $\mathbf{C}_l$ and the archive if needed;    ▷ This is what Algorithm 4 represents.
23:    end for
24: end procedure
```



(a) MGPSO in which each objective has a PSO sub-swarm and different sub-swarms interact with each other using an archive of non-dominated solutions




Archive
Context vector
k sub-swarms
Only one sub-objective shared between the two objectives in cycles of equal length (γ iterations per objective)
1
2
n_c
A pool of context vectors (n_c context vectors)

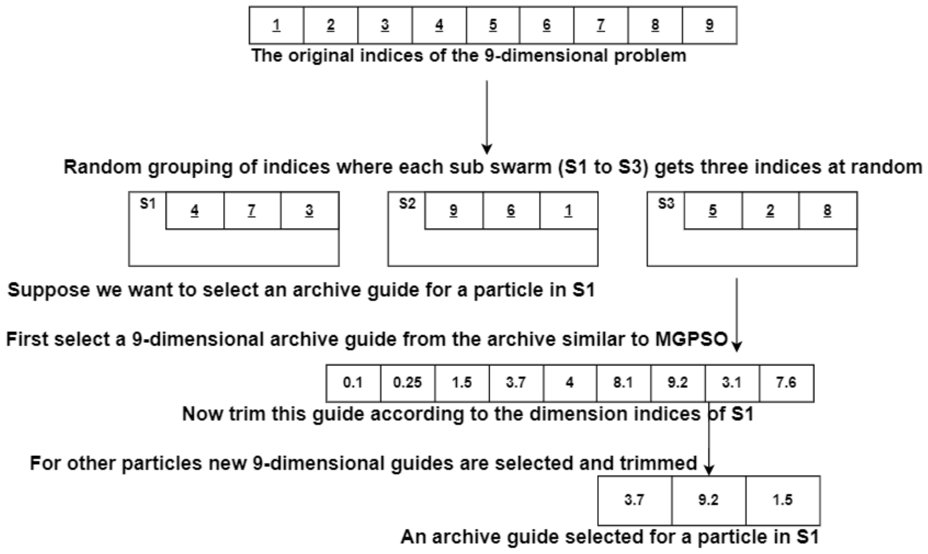
(b) A basic CC-based variant of MGPSO where each sub-objective has a CPSO swarm (consisting of k sub-swarms) assigned to it

(c) The proposed CCMGPSO

Figure 1: Three approaches for solving an arbitrary large-scale 2-objective optimization problem: (a) MGPSO where each objective has an n_x-dimensional PSO sub-swarm and all sub-swarms interact with an archive of non-dominated solutions, (b) A basic CC-based variant of MGPSO in which each objective has its own CPSO swarm (consisting of k sub-swarms) and (c) The proposed CCMGPSO in which there is only one CPSO swarm (consisting of k sub-swarms) that is shared between all objectives. In the CCMGPSO, the CPSO swarm is shared by all objectives in cycles of equal length (γ iterations per objective) to save computational budget.

Algorithm 3 The modified context vector replacement of the CCMGPSO

```
1: procedure B
2:     This is a function used in lines 5, 12, and 19 of Algorithm 4 which returns an $n_x$-dimensional decision vector from a given $\frac{n_x}{k}$-dimensional solution. The resulting $n_x$-dimensional vector is fed into the optimization problem.
3:     Input: $\mathbf{C}$ (set of context vector decision vectors), $l$ (the current context vector index),$j$ (the sub-swarm index), $\mathbf{z}$ (an arbitrary $\frac{n_x}{k}$-dimensional vector belonging to the $j$-th sub-swarm;
4:     Return $(\mathbf{C}_{l1}, \mathbf{C}_{l2}, \ldots, \mathbf{C}_{lj-1}, \mathbf{z}, \mathbf{C}_{lj+1}, \ldots, \mathbf{C}_{lk})$
5:             ▷ $\mathbf{C}_{ij}$ is the the $j$-th decision variable group of the $i$-th context vector, such that $1 \leq j \leq k$ and $1 \leq i \leq n_c$.
6: end procedure
```



1 2 3 4 5 6 7 8 9
The original indices of the 9-dimensional problem
Random grouping of indices where each sub swarm (S1 to S3) gets three indices at random
S1 4 7 3
S2 9 6 1
S3 5 2 8
Suppose we want to select an archive guide for a particle in S1
First select a 9-dimensional archive guide from the archive similar to MGPSO
0.1 0.25 1.5 3.7 4 8.1 9.2 3.1 7.6
Now trim this guide according to the dimension indices of S1
For other particles new 9-dimensional guides are selected and trimmed
3.7 9.2 1.5
An archive guide selected for a particle in S1

Figure 2: The archive guide implementation of CCMGPSO for $\frac{n_x}{k}$-dimensional particles explained through an arbitrary 9-dimensional problem whose decision variables are partitioned into three sub-populations. This process is repeated per particle.

Algorithm 4 The cooperative part of the CCMGPSO

```
1: procedure Cooperative
2:     Input: $\mathbf{C}$ (the set of context vector decision vectors), $\widehat{\mathbf{C}}$ (the set of context vector objective vectors), $l$ (the current context vector index), $o$ (the objective that is currently being optimized by the CPSO), $\mathbf{S}$ (the set of the $\frac{n_x}{k}$-dimensional sub-swarms), $f$ (the optimization problem)  $\triangleright$ This algorithm is used in line 22 of Algorithm 1.
3:     for each sub-swarm $j = 1, \ldots k$ do
4:         for each particle $i = 1, \ldots, n_s$ do
5:             $\widehat{\mathbf{x}} \leftarrow f(\mathbf{B}(\mathbf{C}, l, j, \mathbf{S}_j.\mathbf{x}_i))$  $\triangleright$ See Algorithm 3, also $\widehat{\mathbf{x}}$ is the objective vector corresponding to $\mathbf{B}(C, l, j, \mathbf{S}_j.\mathbf{x}_i)$.
6:             if $\widehat{\mathbf{x}}_o < \mathbf{S}_j.pBest_i$ then  $\triangleright$ $\widehat{\mathbf{x}}_o$ is the $o$-th objective value of the objective vector $\widehat{\mathbf{x}}$
7:                 $\mathbf{S}_j.\mathbf{y}_i \leftarrow P_j.\mathbf{x}_i$
8:                 $\mathbf{S}_j.pBest_i \leftarrow \widehat{\mathbf{x}}_o$
9:             end if
10:            if $\widehat{\mathbf{x}}_o < \widehat{\mathbf{C}}_{l_o}$ then
11:                $\widehat{\mathbf{C}}_l \leftarrow \widehat{\mathbf{x}}$  $\triangleright$ Update the current context vector's objective values.
12:                $\mathbf{C}_l \leftarrow \mathbf{B}(C, l, j, \mathbf{S}_j.\mathbf{x}_i)$  $\triangleright$ Update the current context vector's decision variables.
13:            else if $\widehat{\mathbf{x}}_o = \widehat{\mathbf{C}}_{l_o}$ then
14:                if $\widehat{\mathbf{C}}_l$ does not dominate $\widehat{\mathbf{x}}$ then
15:                    $\widehat{\mathbf{C}}_l \leftarrow \widehat{\mathbf{x}}$
16:                    $\mathbf{C}_l \leftarrow \mathbf{B}(C, l, j, S_j.\mathbf{x}_i)$
17:                end if
18:            end if
19:            $AddToArchive(\mathbf{B}(C, l, j, \mathbf{S}_j.\mathbf{x}_i), \widehat{\mathbf{x}})$  $\triangleright$ Check if the new vector $\mathbf{B}(C, l, j, \mathbf{S}_j.\mathbf{x}_i), X)$ has an undominated objective vector. Also see Algorithm 3 for more information about the function.
20:        end for
21:    end for
22: end procedure
```

lower bound, it was assigned the upper bound and the lower bound respectively.

5.2. Parameters

All parameters for all algorithms were set to their recommended values as per their respective authors. For S3-CMA-ES [38], the sample size for variable classification was set to 50, and the sub-population size was set to 5. IBEA was embedded into the DGEA framework (referred to as D-IBEA) because it had previously shown competitive results in [34] 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 [23]. For LMOCSO [11], 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.

Since the authors of MGPSO only recommended benchmark-specific parameters in [14] and [15] for WFG and ZDT problems, the MGPSO parameters were tuned for the DTLZ suite. For MGPSO for the WFG and ZDT problems, the same parameters as in [14] and [15] were used. For the DTLZ problems, MGPSO showed better performance when the tournament size was equal to 2 or 3; therefore, for the ease of parameter tuning and as recommend in [14], the tournament size of MGPSO was set to 3 for the DTLZ problems. Other MGPSO parameters (c_1, c_2, c_3, and ω) were tuned for 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 [14]:

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

The resulting parameters are listed in Table 1.

The number of individuals was set to 300 in all algorithms. For MGPSO, these 300 particles were split evenly between the MGPSO sub-swarms,

$\{150, 150\}$ and $\{100, 100, 100\}$ for two and three objectives respectively. For CCMGPSO, these 300 particles were split between the sub-swarms 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 for all algorithms, because the algorithms that do not use external archives keep a population of non-dominated solutions that will eventually be compared with the solutions in the external archives of other algorithms. Therefore, the archive size was set to the population size for all algorithms to ensure fair comparisons. 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 2, 3, 5, 6, and 8 and were obtained using empirical observations.

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

5.3. Benchmark Suites

Four benchmark suites (a total of 30 functions) were used in this study, WFG [64, 65], ZDT [66], DTLZ [67], and LSMOP [68]. With reference to the WFG problems, 500-, 750-, and 1000-dimensional 2- and 3-objective problems (similar to the previous scalability study [20]) were used. For the 500-, 750- and 1000-dimensional WFG problems the number of position parameters [65] was set to 100, 150, and 200 respectively. For other problems, 1000-, 1500-, and 2000-dimensional problems were used. For the DTLZ and LSMOP problems, two and three objectives were used; however, for the ZDT problems only two objectives were used because the ZDT problems are not scalable objective-wise.

5.4. Performance Measures

The inverted generational distance (IGD) [69, 70] 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 [63]. Note that, unlike [20], the hypervolume measure (HV) [71] was excluded from this study due to the difficulties of selecting an appropriate reference point. As previously mentioned in [40], using the HV measure when the number of decision variables is relatively large might pose some challenges (defined momentarily) for picking a suitable reference point. For large-scale optimization problems MOEAs usually fail to converge to the true fronts [40]. This is more specifically the case for the large-scale instances of some DTLZ problems, such as DTLZ1, DTLZ3, and DTLZ6. Even when the problem is not large scale, picking an appropriate and fair reference point to calculate the HV can be challenging. It has been shown that the results of HV comparisons between different algorithms depend on the location of the reference point [72, 73]. Ishibuchi *et al.* [74] noted that some studies use a point slightly worse than the nadir point 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 [75] used a point 1.01 times worse than the nadir point, while the reference points used by Maltese *et al.* [76] and Wagner *et al.* [77] were 1.1 and 1.4 times worse than the nadir point respectively. Ishibuchi *et al.* [74] 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 point may not always be suitable. Therefore, only IGD is used in this study for performance evaluation.

5.5. Statistical Significance

All algorithms were run 30 times for each test instance. The algorithms were compared using the two-tailed Mann-Whitney U test [78] 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 2 to 9. This method of comparing different algorithms is discussed in more detail in [79].

5.6. Sensitivity Analysis of CCMGPSO Parameters

CCMGPSO has three main parameters, the tournament size (n_T) (inherited from MGPSO), the number of context vectors (n_c), and the number of CPSO iterations per objective (γ). As previously discussed in [14, 15] a tournament size of 2 or 3 produced the best results for MGPSO. For CCMGPSO, the tournament size was therefore set to 2 for all problems. For each problem, different sets of control parameters were evaluated and compared with each other. Conclusions regarding the sensitivity of each control parameter were drawn from the mean of the IGD values over 30 independent runs. For all problems, competitive performance was observed when at least n_m context vectors were used. For the ZDT and DTLZ problems, better performance was observed when exactly n_m context vectors were used. For most of the 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 3. As seen in the plot, CCMGPSO with 10 context vectors resulted in the best final IGD values. In the experiments, using too many context vectors hurt the overall convergence as some context vectors did not receive enough iterations under a limited computational budget. Therefore, n_c was set to 10 for all WFG functions.

Regarding the γ parameter, $\gamma = 10$ produced competitive performance for all the DTLZ and ZDT problems. For the WFG problems, $\gamma = 10$ or $\gamma = 20$ resulted in good performance; however, slightly better performance was observed with $\gamma = 20$ for some functions. Two examples of the evaluated values of γ for the 2-objective 2000-dimensional DTLZ6 and the 2-objective 1000-dimensional WFG6 are depicted in Figure 4. As seen in the plots, for 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. For DTLZ6, slightly better results were observed with $\gamma = 10$. As for the WFG6 problem in Figure 4, $\gamma = 10$ 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 and ended up with better IGD values. Similar to n_c, in the experiments using γ values that were too large hurt the overall performance by preventing objectives and context vectors from receiving enough CPSO cycles under a limited computational budget. Hence, the value of γ was set to 20 for all WFG problems.

For the LSMOP problems, different values for γ and n_c showed optimal performance for each individual problem. The parameter values used for these

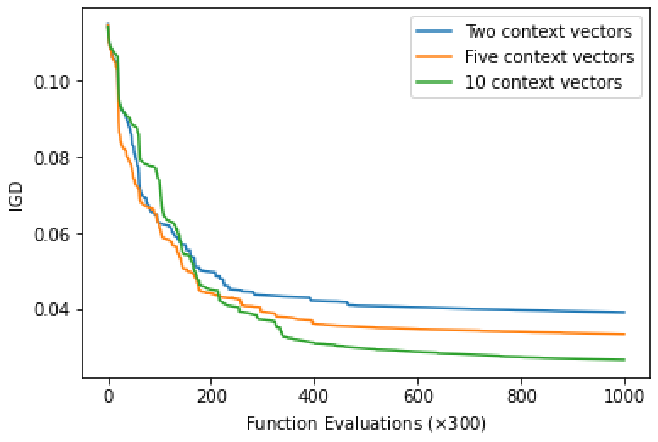
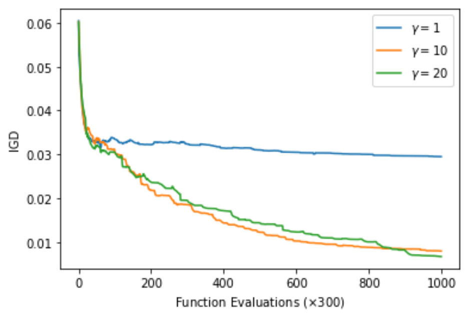
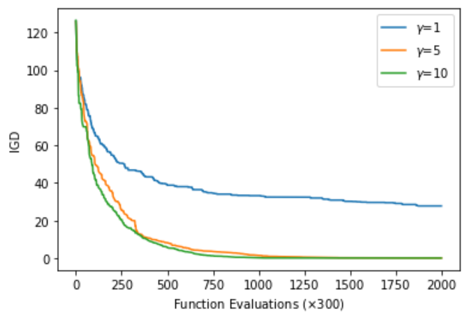


Figure 3: Sensitivity of the n_c control parameter demonstrated for the 2-objective 1000-dimensional WFG2.

problems (as listed in Table 11) were picked after testing 1000 different pairs (λ, n_c) for each problem, and choosing the best ones according to the IGD metric.



(a) The 2-objective 1000-dimensional WFG6 with 10 context vectors



(b) The 2-objective 2000-dimensional DTLZ6 with two context vectors

Figure 4: Sensitivity of the γ control parameter demonstrated for the 2-objective 1000-dimensional WFG6 and the 2-objective 2000-dimensional DTLZ6.

6. Results and Discussion

The experimental results of this study are listed in Tables 2 to 9 and discussed in the following subsections.

Table 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

6.1. The WFG Test Suite

The results related to the WFG test suite are provided in Tables 2, 3, and 4. Tables 2 and 3 list the results for WFG1 to WFG4 and WFG5 to WFG9 respectively, whereas Table 4 provides overall results for the WFG suite based on aggregated scores.

Generally speaking, CCMGPSO had the best performance for most WFG problems, including WFG1, WFG3, WFG4, WFG5, and WFG7. On the other hand, S3-CMA-ES had the worst possible performance for all functions. This is due to the fact that S3-CMA-ES uses many 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 Pareto-optimal front (POF) could outperform the ones that use separate function evaluations for decision variable analysis.

For WFG1, CCMGPSO had the best performance for all 2- and 3-objective test instances; however, despite not being outperformed with statistical significance, CCMGPSO achieved a worse mean than that of WOF-NSGAII's for 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 for WFG1, constantly being outperformed by LMOCSO, WOF-NSGAII, and CCMGPSO. MMOPSO again showed good scalability for WFG1 similar to the previous scalability study. Previous studies [76] had also shown the efficiency of algorithms using the penalty-based boundary intersection method (PBI) (such as MOEA/D [58] or MMOPSO [23]) in solving biased multi-objective optimization problems with mixed POF geometries (such as WFG1).

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

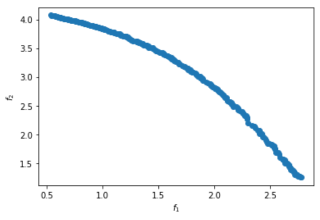
For WFG3 and the multi-modal WFG4, CCMGPSO was the best-performing algorithm for all test instances except the 500-dimensional 2-objective WFG3. Similar to WFG2, D-IBEA had a loss in relative IGD performance from two to three objectives and LMOCSO had poor dimension-wise scalability. LMOCSO outperformed D-IBEA and MGPSO for the 2- and 3-objective WFG4. It may be worth mentioning that using large values for k (number of dimension groups) and dividing the large-scale problems into smaller sub-problems resulted in improved performances for the separable and multi-modal WFG4. An example demonstrating this for the 2-objective 1000-dimensional WFG4 is provided in Figure 5. With $k = 1$ and sub-swarms having 280 particles, CCMGPSO obtained a POF similar to that of the MGPSO's, which ranked fifth in Table 2 with reference to IGD. However, with increase in the value of k, the obtained POFs visibly became better.

With few exceptions, CCMGPSO had the best performance for most test instances of WFG5 to WFG7. D-IBEA and MGPSO showed poor scalability for WFG5, LMOCSO showed poor scalability for WFG5 for two objectives, but outperformed D-IBEA and MGPSO for three objectives. For 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. For WFG6, D-IBEA was the best algorithm after CCMGPSO, WOF-NSGAII, and MMOPSO, constantly outperforming MGPSO, LMOCSO, and S3-CMA-ES.

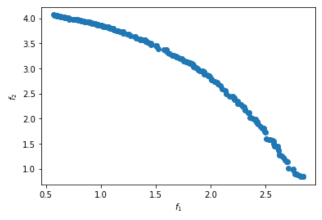
For WFG7, CCMGPSO was the best-performing algorithm for all test instances, obtaining the maximum possible score (+6) for all of the instances. D-IBEA and LMOCSO failed to show competitive performance for WFG7, as they were often outperformed by all algorithms except S3-CMA-ES. For the non-separable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

Finally, the worst CCMGPSO performance for an individual problem was observed for 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 for 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 (with statistical significance) the best-performing algorithm for WFG9, ranking first for all test instances and only failing to outperform MMOPSO for 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 for almost all test instances of WFG9.
A visual comparison of these algorithms for the 1000-dimensional 2-objective WFG3 is depicted in Figure 6. As seen in Figure 6, CCMGPSO showed both better convergence and diversity for WFG3, followed by WOF-NSGAII. MMOPSO found a relatively well-converged POF; however, the diversity was not on par with CCMGPSO or WOF-NSGAII.

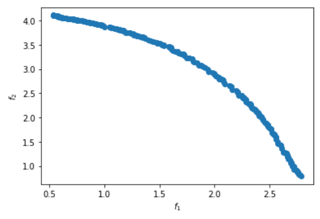
[2]As defined in [64], in a deceptive objective function the majority of the search space leans towards a deceptive optimum instead of the true one.



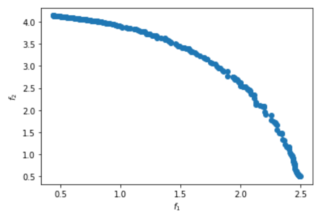
(a) MGPSO



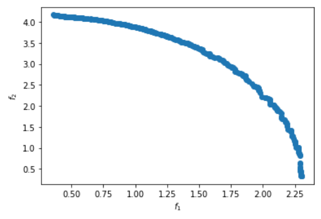
(b) CCMGPSO with one sub-problem having 280 particles



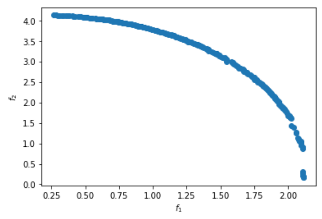
(c) CCMGPSO with two sub-problems each having 140 particles



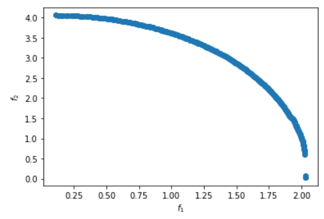
(d) CCMGPSO with 10 sub-problems each having 28 particles



(e) CCMGPSO with 20 sub-problems each having 14 particles

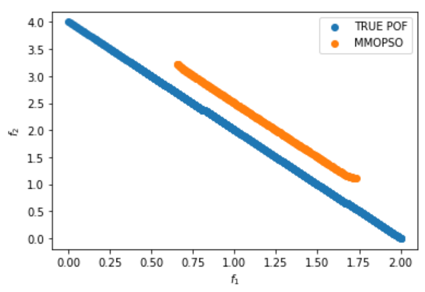


(f) CCMGPSO with 70 sub-problems each having four particles

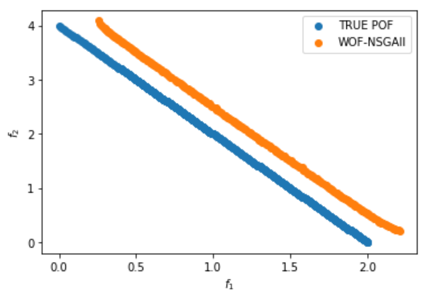


(g) CCMGPSO with 280 sub-problems each having one particle

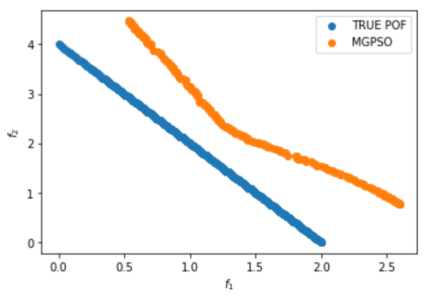
Figure 5: The efficiency of dividing a large-scale multi-objective optimization problem into smaller sub-problems demonstrated for the 2-objective 1000-dimensional WFG4. The figures are all Pareto-optimal fronts.



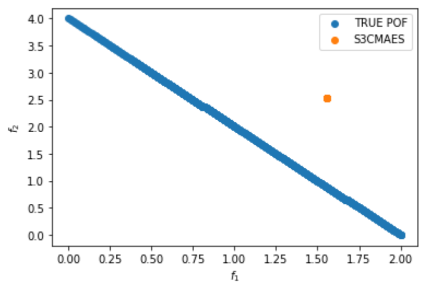
(a) MMOPSO



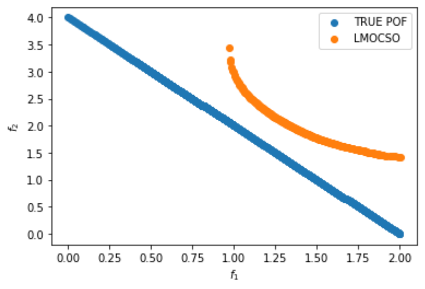
(b) WOF-NSGAII



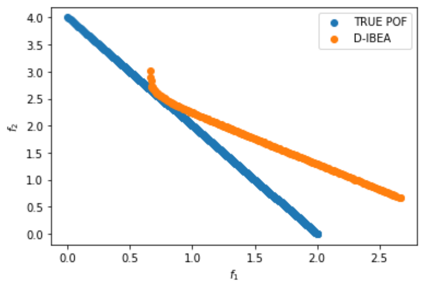
(c) MGPSO



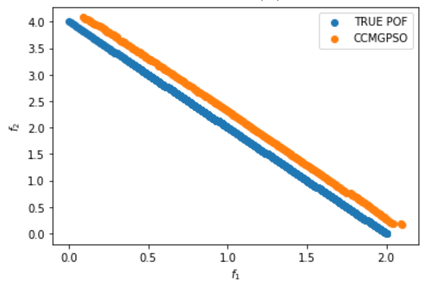
(d) S3CMAES



(e) LMOCSO

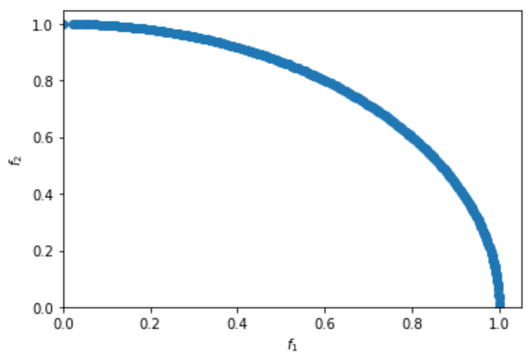


(f) D-IBEA

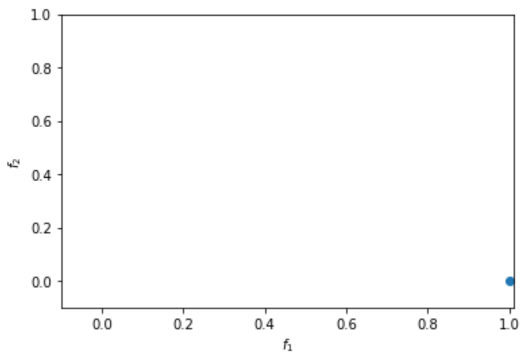


(g) CCMGPSO

Figure 6: A visual comparison of the obtained POFs by the algorithms for the 1000-dimensional 2-objective WFG3.

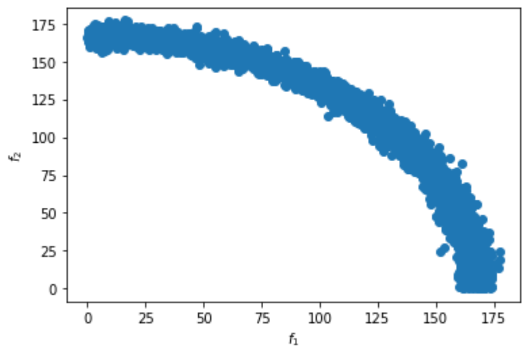


(a) A good run of CCMGPSO for DTLZ4

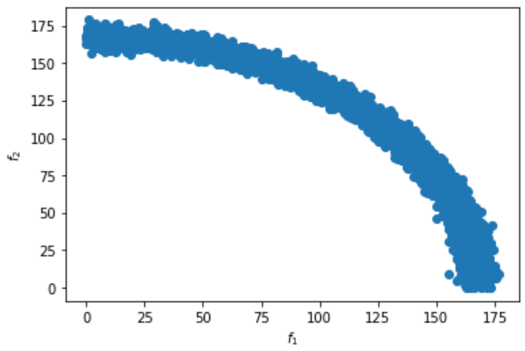


(b) A bad run of CCMGPSO for DTLZ4

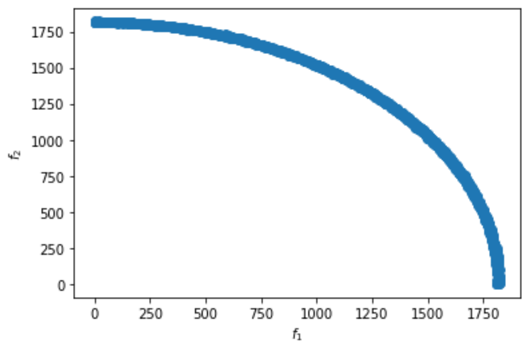
Figure 7: Examples of a good and a bad run of CCMGPSO for the 2000-dimensional 2-objective DTLZ4



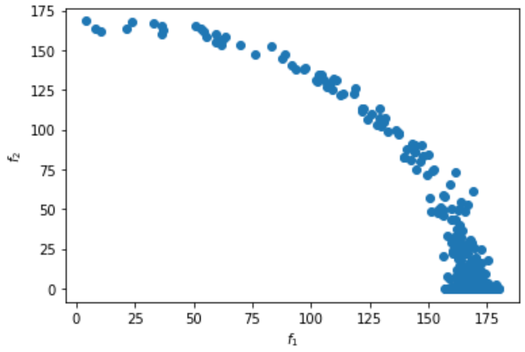
(a) DTLZ2



(b) DTLZ5

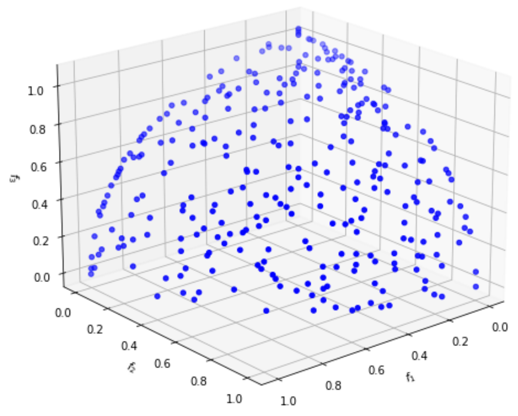


(c) DTLZ6

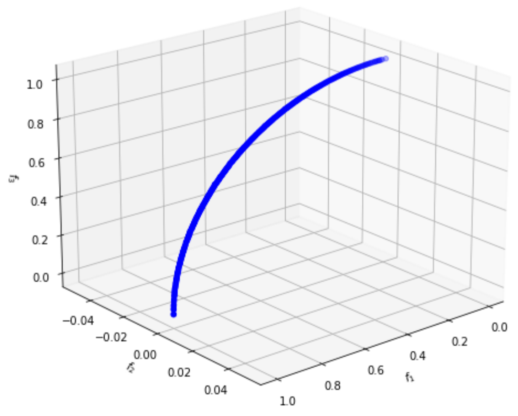


(d) DTLZ4

Figure 8: 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

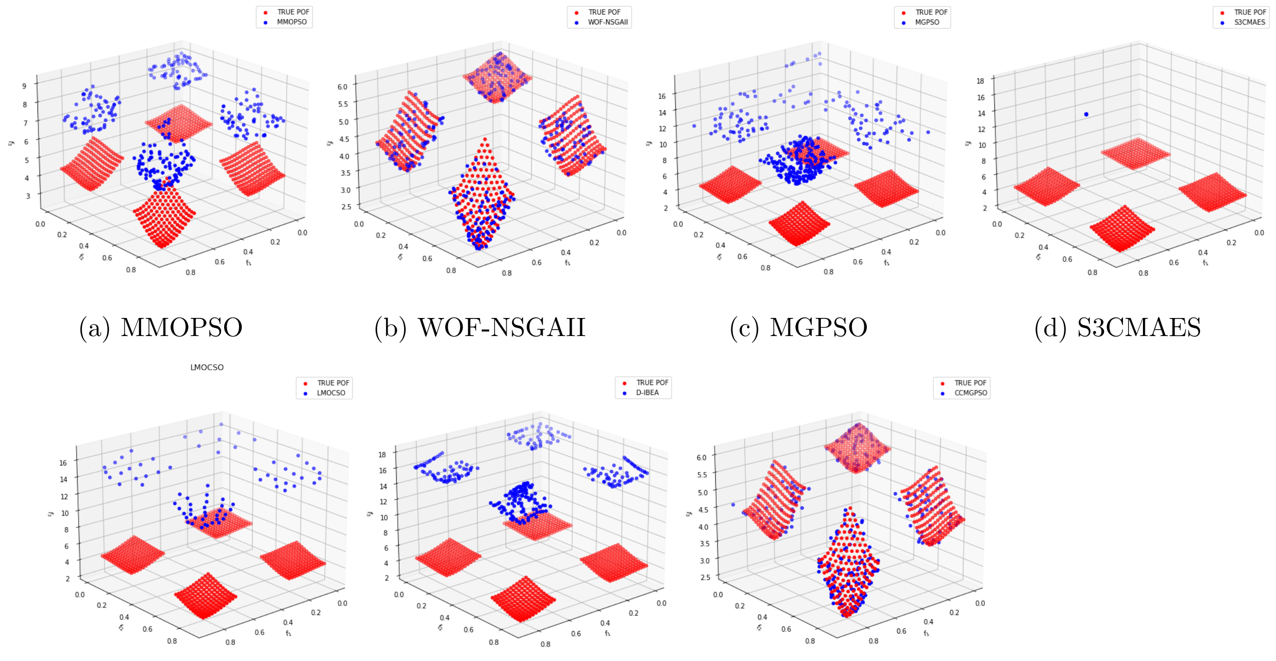


(a) A good run of CCMGPSO for DTLZ4



(b) A bad run of CCMGPSO for DTLZ4

Figure 9: Examples of a good and a bad run of CCMGPSO for the 2000-dimensional 3-objective DTLZ4



(a) MMOPSO (b) WOF-NSGAII (c) MGPSO (d) S3CMAES

(e) LMOCSO (f) D-IBEA (g) CCMGPSO

Figure 10: A visual comparison of the obtained POFs by the algorithms for the 2000-dimensional 3-objective DTLZ7.

6.2. Summary of Results for the WFG Test Suite

The overall IGD rankings of the algorithms (based on the aggregated scores) for the WFG test suite are listed in Table 4. As seen in Table 4, CCMGPSO had the best overall score for 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 2, 3, and 4, LMOCSO, MGPSO, and D-IBEA all showed poor scalability for the WFG suite. The velocity update mechanism of LMOCSO (based on the competitive swarm optimizer (CSO)) that was proposed to improve diver-

Table 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 5.5.

					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$^{\approx}$	-4	-6	0	-2	5$^{\approx}$
				Rank	2	1$^{\approx}$	5	6	3	4	1$^{\approx}$
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 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 5.5.

					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	$4^{\approx}$	$4^{\approx}$	-4	-6	-2	0	$4^{\approx}$
				Rank	$1^{\approx}$	$1^{\approx}$	4	5	3	2	$1^{\approx}$
	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	$5^{\approx}$	-2	-6	-4	2	$5^{\approx}$
				Rank	3	$1^{\approx}$	4	6	5	2	$1^{\approx}$
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	$5^{\approx}$	-2	-6	-4	0	$5^{\approx}$
				Rank	2	$1^{\approx}$	4	6	5	3	$1^{\approx}$
		1000	3.00e+5	Difference	2	$5^{\approx}$	-2	-6	-4	0	$5^{\approx}$
				Rank	2	$1^{\approx}$	4	6	5	3	$1^{\approx}$
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	$5^{\approx}$	$5^{\approx}$	-3	-6	-3	0	2
				Rank	$1^{\approx}$	$1^{\approx}$	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 4: The overall performance of each algorithm for 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 5.5.

				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

sity, showed unsatisfactory convergence for the large-scale instances of the WFG problems. However, this approach showed minor signs of scalability for the multi-modal WFG4, with LMOCSO outperforming all algorithms except the first three (CCMGPSO, WOF-NSGAII, and MMOPSO) for most test instances. D-IBEA managed to outperform MGPSO and LMOCSO (for WFG3, WFG6, and WFG9), MMOPSO for the 750-dimensional 3-objective WFG6, and CCMGPSO for some instances of WFG9; however, it showed really poor performance for other problems such as WFG4, WFG5, and WFG7, resulting in an overall mediocre performance.

CCMGPSO had the best overall scores for all experiments except one, as discussed above and listed in Table 4. With reference to individual problems and the different types of problems, CCMGPSO showed extremely competitive performance for separable and uni-modal problems, achieving the best performance for all experiments involving WFG1 and WFG7. The experiments in which CCMGPSO was outperformed all involved problems that were multi-modal (such as being outperformed by WOF-NSGAII for some instances of WFG2, and all instances of WFG9), deceptive (such as being outperformed by WOF-NSGAII for one instance of WFG5, and by MMOPSO and D-IBEA for some instances of WFG9) or non-separable (such as being outperformed for some instances of WFG6, WFG8, and WFG9). Perhaps unsurprisingly, CCMGPSO's worst performance for an individual optimization problem was observed for WFG9, a problem which has all of the aforementioned properties simultaneously (multi-modal, non-separable and deceptive). Another interesting observation was for WFG4 (separable and multi-modal), where CCMGPSO had the best performance for all experiments.

6.3. The DTLZ Test Suite

The results related to the DTLZ test suite are provided in Tables 5, 6, and 7. Tables 5 and 6 list the results for DTLZ1 to DTLZ4 and DTLZ5 to DTLZ7 respectively, whereas Table 7 provides overall results for the DTLZ suite based on aggregated scores.

For DTLZ1, CCMGPSO had the best performance, outperforming its counterparts in most experiments. For 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 for the multi-modal DTLZ1, constantly being outperformed by the other algorithms except S3-CMA-ES.

Similar to DTLZ1, CCMGPSO also had the best performance for DTLZ2, obtaining the best possible score (+6) for all experiments. D-IBEA and MGPSO both showed poor scalability for DTLZ2, whereas WOF-NSGAII and MMOPSO were the second- and third-best performing algorithms after CCMGPSO for DTLZ2 respectively. For 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) for the 2000-dimensional 2-objective DTLZ3. Similar to DTLZ1 (another multi-modal problem), MGPSO showed poor scalability for DTLZ3. Despite competitive performance for almost all problems, WOF-NSGAII was often outperformed for the test instances of DTLZ3; for example, WOF-NSGAII ranked fourth for 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. For the 2-objective DTLZ4, CCMGPSO was outperformed mainly due to some bad runs (according to the mean IGD values over 30 independent runs), despite having some good runs as well. As depicted in Figure 7, a bad run of CCMGPSO for the 2-objective DTLZ4 typically involved a POF that only covered a very specific (and small) portion of the true POF. Interestingly enough, DTLZ4 possesses the *bias* property [65]. As previously defined in

[65], for 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 for DTLZ4 [40]. 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 8. As seen in Figure 8, DTLZ4 had a more non-uniform set of objective vectors compared with its counterparts for DTLZ2, DTLZ5, and DTLZ6, where one specific area of the plot (close to $\mathbf{f}_2 = 0$) was more crowded with vectors. As seen in Figure 7, for bad runs of CCMGPSO for the 2-objective DTLZ4 only one Pareto-optimal solution was found in the same region (close to $\mathbf{f}_2 = 0$). This was a result of all context vectors ending up in one specific area of the objective space where $\mathbf{f}_2$ is close to zero; in such cases, CCMGPSO will keep optimizing $\mathbf{f}_1$ until it is equal or very close to 1, producing only one Pareto-optimal point. The same phenomenon was also observed for the 3-objective DTLZ4 (as shown in Figure 9); however, it did not happen as often and therefore CCMGPSO was able to outperform all other algorithms for the 3-objective DTLZ4 with statistical significance.

For DTLZ5, CCMGPSO was the best-performing algorithm for all experiments, whereas D-IBEA and MGPSO both showed poor scalability. CCMGPSO and WOF-NSGAII were the best-performing algorithms for DTLZ6 and DTLZ7. For the 2-objective DTLZ6, CCMGPSO and WOF-NSGAII had statistically similar performances, with the latter obtaining the best mean over 30 independent runs. For the 3-objective DTLZ6, CCMGPSO outperformed WOF-NSGAII for 1000 and 1500 dimensions with statistical significance but was outperformed for 2000 dimensions. Similar to DTLZ5, MGPSO and D-IBEA showed poor performances for DTLZ6 and were constantly outperformed by CCMGPSO, WOF-NSGAII and LMOCSO. LMOCSO also managed to outperform MMOPSO for the 3-objective DTLZ6.

Finally, for DTLZ7, CCMGPSO and WOF-NSGAII again were the best performers. CCMGPSO outperformed all algorithms for two objectives, but had a performance statistically similar to that of WOF-NSGAII for three objectives. MGPSO showed relatively good scalability for DTLZ7, outperforming both LMOCSO and D-IBEA. A visual comparison of these algorithms for the 2000-dimensional 3-objective DTLZ7 is depicted in Figure 10. As seen in Table 6 and Figure 10, CCMGPSO and WOF-NSGAII were the best performers for this problem, with the best convergence and diversity in their obtained POFs. Moreover, it can be seen that the solutions

Table 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 5.5.

					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	$3^{\approx}$	-4	-6	$3^{\approx}$	$3^{\approx}$	$3^{\approx}$
				Rank	2	$1^{\approx}$	3	4	$1^{\approx}$	$1^{\approx}$	$1^{\approx}$
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	$4^{\approx}$	2	$4^{\approx}$
				Rank	3	3	4	5	$1^{\approx}$	2	$1^{\approx}$
		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	$5^{\approx}$	$5^{\approx}$
				Rank	3	4	5	6	2	$1^{\approx}$	$1^{\approx}$
		1500	5.10e+5	Difference	-1	-1	-4	-6	2	$5^{\approx}$	$5^{\approx}$
				Rank	3	3	4	5	2	$1^{\approx}$	$1^{\approx}$
		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 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 $\approx$ 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 5.5.

					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	$5^{\approx}$	-2	-6	2	-4	$5^{\approx}$
				Rank	3	$1^{\approx}$	4	6	2	5	$1^{\approx}$
		1500	4.50e+5	Difference	0	$5^{\approx}$	-3	-6	2	-3	$5^{\approx}$
				Rank	3	$1^{\approx}$	4	5	2	4	$1^{\approx}$
		2000	6.00e+5	Difference	0	$5^{\approx}$	-3	-6	2	-3	$5^{\approx}$
				Rank	3	$1^{\approx}$	4	5	2	4	$1^{\approx}$
	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	$5^{\approx}$	0	-6	-2	-4	$5^{\approx}$
				Rank	2	$1^{\approx}$	3	6	4	5	$1^{\approx}$
		2000	6.00e+5	Difference	2	$5^{\approx}$	0	-6	-2	-4	$5^{\approx}$
				Rank	2	$1^{\approx}$	3	6	4	5	$1^{\approx}$

(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 7: The overall performance of each algorithm for 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 5.5.

				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

found by LMOCSO were more equally-spaced than those of CCMGPSO or MMOPSO, 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 there are many solutions in one specific area of the POF (covered by a limited number of reference vectors), most of them are discarded. Since LMOCSO does not use an external archive, discarding solutions based on diversity can potentially hurt the overall performance when the solutions are not well converged yet.

6.4. Summary of Results for the DTLZ Test Suite

The overall results for the DTLZ suite (based on aggregated scores) are provided in Table 7. As seen in Table 7, CCMGPSO achieved the best overall scores for DTLZ1 to DTLZ7 for all six experiments, followed by WOF-NSGAII and MMOPSO at second and third respectively. LMOCSO mostly ranked fourth; however, for the 2000-dimensional 3-objective DTLZ problems, LMOCSO was tied with MMOPSO at third, possibly due to its superiority for DTLZ6.

MGPSO and D-IBEA both failed to show competitive performance for most DTLZ problems. D-IBEA showed scalable performance for fully multi-modal problems (DTLZ1 and DTLZ3), whereas MGPSO managed to outperform LMOCSO and D-IBEA for DTLZ7. The promising results for multi-modal problems produced by D-IBEA may be due to 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 for DTLZ1 and DTLZ3. The reason behind the poor performance of MGPSO for DTLZ1 is provided in Appendix A. In short, for DTLZ1 at the very early stages of optimization, MGPSO tends to find a decision vector whose corresponding objective vector has the minimum value for one objective, but poor values for the others. Such a vector is then set to the global best position with respect to that specific objective until the end of the optimization process (the global best position remains unchanged). On the other hand, CCMGPSO constantly shares different global best positions (context vectors) between objectives, and constantly resets the personal best positions, resulting in the best performance for DTLZ1 as well as competitive scalability for DTLZ3. This dynamism in global and best positions is 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.

6.5. The ZDT Test Suite

The IGD rankings for the ZDT test suite are provided in Table 8, whereas Table 9 lists the overall rankings based on aggregated scores.

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

CCMGPSO experienced a major performance drop for the multi-modal ZDT4, ranking third for the 1500- and 2000-dimensional ZDT4, having had the best performance for 1000 dimensions. LMOCSO was the best-performing algorithm for 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 for ZDT4 and it was outperformed by all algorithms except D-IBEA.

For 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.

Table 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 $\approx$ 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 5.5.

				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	$5^{\approx}$	0	-6	-2	-4	$5^{\approx}$
			Rank	2	$1^{\approx}$	3	6	4	5	$1^{\approx}$
	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 9: The overall performance of each algorithm for 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 5.5.

			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

6.6. Summary of Results for the ZDT Test Suite

The overall results for the ZDT suite (based on aggregated scores) are provided in Table 9.

Similar to the DTLZ suite, CCMGPSO had the best overall (aggregated) scores for all experiments for the ZDT suite as well, with WOF-NSGAII and MMOPSO finishing second and third respectively. Despite being outperformed by MGPSO for ZDT1 to ZDT3, LMOCSO obtained better overall scores than MGPSO as a result of its superiority for ZDT6 and the big difference in performance between them for ZDT4.

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

6.7. The LSMOP Test Suite

The results related to the LSMOP test suite are provided in Tables 12, 13, and 14. Tables 12 and 13 list the results for LSMOP1 to LSMOP4 and LSMOP5 to LSMOP9 respectively, whereas Table 14 provides overall results for the LSMOP suite based on aggregated scores. The properties of the LSMOP problems are listed in Table 10.
For the LSMOP problems, WOF-NSGAII showed competitive performance for every individual problem, having obtained the best overall score for almost every test instance. WOF-NSGAII was the best-performing algorithm

for all problems except LSMOP2 and LSMOP4, and a few instances of LSMOP8 and LSMOP9. Remarkably, WOF-NSGAII significantly outperformed all other algorithms for all instances of LSMOP5 to LSMOP7. Despite its overall competitive performance, WOF-NSGAII showed poor objective-wise scalability for LSMOP2, LSMOP4, and LSMOP9 by obtaining worse overall rankings for these problems when the number of objectives was increased from two to three. The most noticeable instance of this trend happened for the LSMOP4 problem. As seen in Table 12, WOF-NSGAII was constantly the second-worst algorithm for the 3-objective LSMOP4, despite significantly outperforming all other algorithms for all instances of the 2-objective LSMOP4.

After WOF-NSGAII, D-IBEA and CCMGPSO also showed competitive performances for most of the LSMOP problems. CCMGPSO showed poor performance for LSMOP2, LSMOP4, and the 3-objective LSMOP7. CCMGPSO and D-IBEA were the best performers after WOF-NSGAII for LSMOP1, with the former outperforming the latter for the 2-objective LSMOP1. CCMGPSO consistently had poor performance for LSMOP4 and the 2-objective LSMOP2. However, the relative performance (the ranking) of CCMGPSO slightly improved for the 3-objective LSMOP2 (compared with its 2-objective instances). For LSMOP7, CCMGPSO showed poor objective-wise scalability. More specifically, when the number of objectives was increased from two to three, the overall ranking of CCMGPSO worsened significantly.

Regarding the strong points of CCMGPSO, it was the second-best algorithm for most instances of LSMOP5 to LSMOP8, second to only WOF-NSGAII. Moreover, CCMGPSO had the best performance for the 1000-dimensional 3-objective LSMOP8, and it outperformed all algorithms for all instances of the 3-objective LSMOP9. CCMGPSO outperformed all algorithms except WOF-NSGAII for the 2-objective LSMOP7 (a multi-modal problem). Unsurprisingly, CCMGPSO showed competitive performance for the fully separable and unimodal LSMOP1 by outperforming all algorithms except WOF-NSGAII. For the 2000-dimensional 2-objective functions, CCMGPSO was the second-best algorithm for six (out of nine) problems (LSMOP1, LSMOP5, LSMOP6, LSMOP7, LSMOP8, and LSMOP9). As seen in Table 14, CCMGPSO obtained the second-best (second to only WOF-NSGAII) overall scores for all instances except the 1000-dimensional 3-objective problems, where it obtained the best overall results.

D-IBEA also showed competitive performance for the LSMOP test suite. Along with CCMGPSO, D-IBEA was often the second-best algorithm, out-

Table 10: The separability and modality of the LSMOP experiments (as listed in [80] and [68])

Function	Separability	Modality
LSMOP1	Fully Separable	Unimodal
LSMOP2	Partially Separable	Mixed
LSMOP3	Mixed	Multi-modal
LSMOP4	Mixed	Mixed
LSMOP5	Fully Separable	Unimodal
LSMOP6	Partially Separable	Mixed
LSMOP7	Mixed	Multi-modal
LSMOP8	Mixed	Mixed
LSMOP9	Fully Separable	Mixed

performed mostly by WOF-NSGAII. Unlike CCMGPSO, D-IBEA showed competitive performance for LSMOP2, LSMOP4, and the 3-objective LSMOP7. In fact, D-IBEA significantly outperformed all algorithms for the 3-objective LSMOP4. Furthermore, D-IBEA showed poor performance for LSMOP9 (for which CCMGPSO showed competitive performance). Precisely, D-IBEA was consistently the second-worst algorithm (outperforming only S3-CMA-ES) for the 3-objective LSMOP9.
Further observations can be made regarding other algorithms. S3-CMA-ES was once again the worst performer for all test instances due to the same reason of using too many function evaluations for variable grouping without leaving enough computational budget for the actual objective optimization. MGPSO and MMOPSO showed mediocre overall performances, and generally speaking, LMOCSO was outperformed mostly by WOF-NGSAII, CCMGPSO, and D-IBEA. However, there were individual functions for which LMOCSO was the best performer, such as the 2-objective LSMOP9 and the 3-objective LSMOP2.

6.8. Summary of Results for the LSMOP Test Suite

The overall IGD rankings of the algorithms (based on the aggregated scores) for the LSMOP test suite are listed in Table 14. As seen in Table 14, WOF-NSGAII obtained the best overall scores for all test instances except the 3-objective 1000-dimensional functions, for which CCMGPSO was

Table 11: The CCMGPSO parameters tuned for the LSMOP test suite

Function	n_c	λ
LSMOP1	3	5
LSMOP2	3	10
LSMOP3	10	10
LSMOP4	3	20
LSMOP5	3	15
LSMOP6	10	20
LSMOP7	3	5
LSMOP8	3	5
LSMOP9	3	5

the best performer. After WOF-NSGAII, CCMGPSO always obtained the second-best overall scores for all test instances, although its score was tied with D-IBEA for the 3-objective 2000-dimensional functions. CCMGPSO was mostly followed by D-IBEA, LMOCSO, and MMOPSO, as the third- to fifth-best algorithms with reference to overall scores. Finally, MGPSO and S3-CMA-ES were the worst overall algorithms, with the former outperforming the latter.

7. The Proposed Approach: Strengths and Weaknesses

In this section, the experimental data from four different benchmark suites are analyzed to describe the strengths and weaknesses of the proposed CCMGPSO.

7.1. Strengths

- Generally speaking, the proposed CCMGPSO showed competitive performance for fully separable and unimodal problems. The overall rankings for the ZDT and DTLZ problems back this claim, as well as individual performances for the likes of WFG1, WFG7, DTLZ2, DTLZ4, and DTLZ6. Since CCMGPSO decomposes the different variables of a problem into many smaller groups, it performed significantly better than its competitors for separable and unimodal problems.

Table 12: IGD rankings for LSMOP1 to LSMOP4. 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 5.5.

					Algorithm						
Problem$^{(1)}$	Objectives	Dimensions	FEs	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
LSMOP1	2	1000	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
		1500	3.00e+5	Difference	0	6	-4	-6	-2	2	4
				Rank	4	1	6	7	5	3	2
		2000	3.00e+5	Difference	0	6	-4	-6	-2	2	4
				Rank	4	1	6	7	5	3	2
	3	1000	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
		1500	3.00e+5	Difference	-2	6	-4	-6	0	4	2
				Rank	5	1	6	7	4	2	3
		2000	3.00e+5	Difference	-2	6	-4	-6	0	4	2
				Rank	5	1	6	7	4	2	3
LSMOP2	2	1000	3.00e+5	Difference	0	6	-4	-6	2	4	-2
				Rank	4	1	6	7	3	2	5
		1500	3.00e+5	Difference	0	4	-4	-6	2	6	-2
				Rank	4	2	6	7	3	1	5
		2000	3.00e+5	Difference	0	4	-4	-6	2	6	-2
				Rank	4	2	6	7	3	1	5
	3	1000	3.00e+5	Difference	-2	-4	0	-6	6	4	2
				Rank	5	6	4	7	1	2	3
		1500	3.00e+5	Difference	-2	-4	0	-6	6	4	2
				Rank	5	6	4	7	1	2	3
		2000	3.00e+5	Difference	-2	-4	0	-6	6	4	2
				Rank	5	6	4	7	1	2	3
LSMOP3	2	1000	3.00e+5	Difference	-2	4	-4	-6	6	2	0
				Rank	5	2	6	7	1	3	4
		1500	3.00e+5	Difference	-2	5	-4	-6	4	3	0
				Rank	5	1	6	7	2	3	4
		2000	3.00e+5	Difference	-2	6	-4	-6	2	4	0
				Rank	5	1	6	7	3	2	4
	3	1000	3.00e+5	Difference	-1	5	-4	-6	-1	5	2
				Rank	3	1	4	5	3	1	2
		1500	3.00e+5	Difference	0	5	-4	-6	-2	5	2
				Rank	3	1	5	6	4	1	2
		2000	3.00e+5	Difference	0	5	-4	-6	0	5	0
				Rank	2	1	3	4	2	1	2
LSMOP4	2	1000	3.00e+5	Difference	-1	6	-4	-6	-1	4	2
				Rank	4	1	5	6	4	2	3
		1500	3.00e+5	Difference	-2	6	-4	-6	0	4	2
				Rank	5	1	6	7	4	2	3
		2000	3.00e+5	Difference	-2	6	-4	-6	0	4	2
				Rank	5	1	6	7	4	2	3
	3	1000	3.00e+5	Difference	-1	-4	1	-6	4	6	0
				Rank	5	6	3	7	2	1	4
		1500	3.00e+5	Difference	-2	-4	2	-6	4	6	0
				Rank	5	6	3	7	2	1	4
		2000	3.00e+5	Difference	-2	-4	2	-6	4	6	0
				Rank	5	6	3	7	2	1	4

Table 13: IGD rankings for LSMOP5 to LSMOP9. 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 5.5.

					Algorithm						
Problem(1)	Objectives	Dimensions	FEs	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
LSMOP5	2	1000	3.00e+5	Difference	1	6	-2	-6	1	-4	4
				Rank	3	1	4	6	3	5	2
		1500	3.00e+5	Difference	2	6	-2	-6	0	-4	4
				Rank	3	1	5	7	4	6	2
		2000	3.00e+5	Difference	2	6	-2	-6	0	-4	4
				Rank	3	1	5	7	4	6	2
	3	1000	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
		1500	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
		2000	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
LSMOP6	2	1000	3.00e+5	Difference	0	6	-1	-6	2	-4	3
				Rank	4	1	5	7	3	6	2
		1500	3.00e+5	Difference	0	6	-3	-6	-1	2	2
				Rank	3	1	5	6	4	2	2
		2000	3.00e+5	Difference	-2	6	-4	-6	0	3	3
				Rank	4	1	5	6	3	2	2
	3	1000	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
		1500	3.00e+5	Difference	-2	5	-4	-6	0	3	4
				Rank	5	1	6	7	4	3	2
		2000	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
LSMOP7	2	1000	3.00e+5	Difference	0	6	-2	-6	2	-4	4
				Rank	4	1	5	7	3	6	2
		1500	3.00e+5	Difference	0	6	-3	-6	2	-3	4
				Rank	4	1	5	6	3	5	2
		2000	3.00e+5	Difference	0	6	-4	-6	2	-2	4
				Rank	4	1	6	7	3	5	2
	3	1000	3.00e+5	Difference	-4	6	-2	-6	4	2	0
				Rank	6	1	5	7	2	3	4
		1500	3.00e+5	Difference	0	6	2	-6	4	-4	-2
				Rank	4	1	3	7	2	6	5
		2000	3.00e+5	Difference	0	6	2	-6	3	-3	-2
				Rank	4	1	3	7	2	6	5
LSMOP8	2	1000	3.00e+5	Difference	-2	6	-2	-6	1	-1	4
				Rank	5	1	5	6	3	4	2
		1500	3.00e+5	Difference	-2	6	-4	-6	0	2	4
				Rank	5	1	6	7	4	3	2
		2000	3.00e+5	Difference	-1	6	-4	-6	-1	2	4
				Rank	4	1	5	6	4	3	2
	3	1000	3.00e+5	Difference	-2	3	-4	-6	0	4	5
				Rank	5	3	6	7	4	2	1
		1500	3.00e+5	Difference	-2	5	-4	-6	0	4	3
				Rank	5	1	6	7	4	2	3
		2000	3.00e+5	Difference	-2	5	-4	-6	1	3	3
				Rank	4	1	5	6	3	2	2
LSMOP9	2	1000	3.00e+5	Difference	-2	4	0	-6	6	-4	2
				Rank	5	2	4	7	1	6	3
		1500	3.00e+5	Difference	-1	4	-1	-6	6	-4	2
				Rank	4	2	4	6	1	5	3
		2000	3.00e+5	Difference	0	6	-2	-6	2	-4	4
				Rank	4	1	5	7	3	6	2
	3	1000	3.00e+5	Difference	-2	2	0	-6	4	-4	6
				Rank	5	3	4	7	2	6	1
		1500	3.00e+5	Difference	-2	2	0	-6	4	-4	6
				Rank	5	3	4	7	2	6	1
		2000	3.00e+5	Difference	-2	2	0	-6	4	-4	6
				Rank	5	3	4	7	2	6	1

Table 14: The overall performance of each algorithm for the LSMOP test suite, based on the aggregated scores. The best-performing algorithms were highlighted in grey background. The ranking scheme is explained in Section 5.5.

				Algorithm						
Overall	Objectives	Dimensions	Result	MMOPSO	WOF-NSGAII	MGPSO	S3-CMA-ES	LMOCSO	D-IBEA	CCMGPSO
LSMOP1 to LSMOP9	2	1000	Difference	-8	50	-23	-54	19	-5	21
			Rank	5	1	6	7	3	4	2
		1500	Difference	-5	49	-29	-54	11	8	20
			Rank	5	1	6	7	3	4	2
		2000	Difference	-5	52	-32	-54	5	11	23
			Rank	5	1	6	7	4	3	2
	3	1000	Difference	-18	26	-21	-54	17	23	27
			Rank	5	2	6	7	4	3	1
		1500	Difference	-14	27	-16	-54	16	20	21
			Rank	5	1	6	7	4	3	2
		2000	Difference	-14	28	-16	-54	18	19	19
			Rank	4	1	5	6	3	2	2

- Regarding the modality of separable problems, CCMGPSO also showed competitive performance even when the problems were not unimodal. Examples of this point are observed for the likes of WFG4, WFG5, LSMOP1, LSMOP5, LSMOP9, DTLZ1, and DTLZ7, which are separable problems whose modalities are either multi-modal, deceptive, or mixed.

- The competitive performance of CCMGPSO was not solely limited to fully separable and unimodal problems. CCMGPSO showed competitive performance for the likes of WFG2, WFG6, WFG3, ZDT4, LSMOP6, the 2-objective LSMOP7, and LSMOP8 that have different separability (partially separable, non-separable, and mixed) and modality (multi-modal and deceptive) attributes.

- As one of its main objectives, CCMGPSO outperformed MGPSO for almost all test problems. More specifically, among 30 test problems over four benchmark suites (WFG, DTLZ, ZDT, and LSMOP), CCMGPSO significantly outperformed MGPSO for all problems except LSMOP4.

- In addition to MGPSO, CCMGPSO consistently outperformed other PSO-based algorithms (LMOCSO and MMOPSO) for different test problems. MMOPSO was the best-performing algorithm in the previous scalability study [20] that involved four other PSO-based approaches, including MGPSO.

7.2. Weaknesses

This section lists some of the observed weaknesses of the proposed CCMGPSO, most of which are left open for future investigation and research.

- CCMGPSO showed poor performance for some non-separable and partially separable problems. The most prominent examples of this point are observed for WFG9, some instances of WFG2, LSMOP2, and LSMOP4. In future work, the reasons behind this performance loss need to be investigated.

- CCMGPSO also showed poor performance for some multi-modal and deceptive problems. The most noticeable examples of this point can be seen for the likes of WFG9, LSMOP2, the 3-objective LSMOP7, and LSMOP4.

- Regarding the objective-wise scalability of CCMGPSO, its performance is highly susceptible to worsening for many-objective problems. More specifically, since CCMGPSO optimizes the objectives of a multi-objective problem successively, there is a possibility that it loses performance when a problem has more than two objectives. In such cases (problems with more objectives), CCMGPSO will spend more computational budget on the first few objectives, leaving the others under-optimized. Some examples of this poor objective-wise scalability can be seen for WFG2 and LSMOP7, where the performance significantly worsened when the number of objectives was increased from two to three.

7.3. Random Grouping: A Simple Solution

As mentioned in Section 7.1 and Section 7.2, CCMGPSO uses random grouping by randomly assigning decision variables to different groups. Despite its simple nature, random grouping comes with many advantages, some of which are discussed in this section.

As mentioned before, a major motivation behind the design of CCMGPSO was preserving the computational budget, with reference to which almost all cooperative coevolutionary (CC)-based algorithms are costly. Many approaches in recent years [51, 52, 53, 54, 50] have been proposed to provide more efficient ways of spending the computational budget, and CCMGPSO is no exception. For example, CCMGPSO successively optimizes different objectives by assigning a single CPSO swarm to each objective for pre-defined intervals.

Unlike random grouping, almost all other grouping techniques (such as [81, 82, 36]) take up an additional amount of computational budget and this amount also increases when problems have more variables. Under a limited amount of computational budget, this can significantly damage the performance of an algorithm. For example, in the experiments of this paper, S3-CMA-ES was constantly (with no exceptions) the worst-performing algorithm for all test instances of all problems (from the WFG, ZDT, DTLZ, and LSMOP test suites). S3-CMA-ES spends an additional number of function evaluations on decision variable grouping, leaving little to no room for the actual optimization of objectives. Therefore, this observation inspired the use of random grouping in CCMGPSO as a simple yet effective way of grouping decision variables. Moreover, when used to solve large-scale single-objective problems, random grouping has shown competitive performance for separable, multi-modal, and non-separable problems [83, 26, 27, 28].

8. Conclusion & Future Work

To adapt MGPSO to scale to large-scale multi-objective optimization problems, this paper proposed a new algorithm termed cooperative coevolutionary multi-guide particle swarm optimization (CCMGPSO). CCMGPSO draws inspiration from AM-CCPSO [28] in using multiple context vectors, and sharing them between different objectives in cycles of equal length. Furthermore, CCMGPSO is inspired by the CPSO-H$_k$ in using both n_x and $\frac{n_x}{k}$-dimensional particles where n_x and k are the total number of decision variables and the total number of decision variable groups respectively.

Additionally, an empirical study was done in order to compare CCMGPSO with six other algorithms from the literature, namely MGPSO [15, 14], MMOPSO [23], WOF-NSGAII [61], LMOCSO [11], S3-CMA-ES [38], and D-IBEA [34] which were evaluated on the WFG, ZDT, DTLZ, and LSMOP benchmark suites. CCMGPSO scaled exceptionally well for separable and uni-modal problems. However, it was outperformed for some instances including problems that were either multi-modal, deceptive, or non-separable. In fact, CCMGPSO's worst performance for an individual problem was observed for WFG9, a problem possessing all of the aforementioned properties simultaneously. Moreover, CCMGPSO was also outperformed for the 2-objective DTLZ4 due to the existence of bias, where in some runs all context vectors became stuck in an area of the POF towards which DTLZ4 was biased. However, with reference to overall scores, CCMGPSO was only outperformed once for the 3-objective 500-dimensional WFG problems, and it was the second-best algorithm for the LSMOP test suite (second to only WOF-NSGAII). Therefore, it can be concluded that the proposed CCMG-PSO is highly competitive.

In the final set of experiments, more observations were also made regarding other algorithms. For example, for 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 for some multi-modal and deceptive functions such as WFG2, WFG8, and DTLZ3, it did not obtain satisfactory convergence for most of the large-scale instances of uni-modal problems.

With reference to future work, different considerations can be made. CCMGPSO's susceptibility to performance loss for multi-modal, non-separable, deceptive, or biased problems can be further studied. More studies can also be conducted on the order in which CCMGPSO optimizes different objectives. For example, for 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 coevolutionary 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 coevolution during each iteration.

Finally, this work only considered 2- and 3-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 for 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. Moreover, the CCMGPSO can be adapted to solve dynamic large-scale multi- and many-objective problems.

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Y. Collette, P. Siarry, Multiobjective optimization: principles and case studies, Springer Science & Business Media, 2004.

[2] Masuduzzaman, G. Rangaiah, Multi-objective optimization applications in chemical engineering, Multi-Objective Optimization: Techniques and Applications in Chemical Engineering (2009) 27–59.

[3] M. G. C. Tapia, C. A. C. Coello, Applications of multi-objective evolutionary algorithms in economics and finance: A survey, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 532–539.

[4] F. A. Zotes, M. S. Peñas, Particle swarm optimisation of interplanetary trajectories from earth to jupiter and saturn, Engineering Applications of Artificial Intelligence 25 (1) (2012) 189–199.

[5] S. Midya, A. Roy, K. Majumder, S. 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 (2018) 58–84.

[6] J. Horn, N. Nafpliotis, D. 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, Ieee, 1994, pp. 82–87.

[7] E. H. Ruspini, I. S. Zwir, Automated qualitative description of measurements, in: Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference (Cat. No. 99CH36309), Vol. 2, IEEE, 1999, pp. 1086–1091.

[8] K. Deb, R. Datta, Hybrid evolutionary multi-objective optimization and analysis of machining operations, Engineering Optimization 44 (6) (2012) 685–706.

[9] M. Sessarego, K. Dixon, D. Rival, D. Wood, A hybrid multi-objective evolutionary algorithm for wind-turbine blade optimization, Engineering Optimization 47 (8) (2015) 1043–1062.

[10] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, M. Gong, A multi-objective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables, IEEE Transactions on Evolutionary Computation 20 (2) (2015) 275–298.

[11] Y. Tian, X. Zheng, X. Zhang, Y. Jin, Efficient large-scale multiobjective optimization based on a competitive swarm optimizer, IEEE Transactions on Cybernetics 50 (8) (2019) 3696–3708.

[12] E. T. Oldewage, A. P. Engelbrecht, C. W. Cleghorn, The merits of velocity clamping particle swarm optimisation in high dimensional spaces, in: Proceedings of the IEEE Symposium Series on Computational Intelligence, IEEE, 2017, pp. 1–8.

[13] X. Zhang, Y. Tian, R. Cheng, Y. Jin, A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization, IEEE Transactions on Evolutionary Computation 22 (1) (2016) 97–112.

[14] C. Scheepers, Multi-guided particle swarm optimization: A multi-objective particle swarm optimizer, Ph.D. thesis, University of Pretoria (2017).

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

[16] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the International conference on neural networks, Vol. 4, IEEE, 1995, pp. 1942–1948.

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

[18] C. Steenkamp, A. P. Engelbrecht, A scalability study of the multi-guide particle swarm optimization algorithm to many-objectives, Swarm and Evolutionary Computation 66 (2021) 100943. doi:https://doi.org/10.1016/j.swevo.2021.100943.
URL `https://www.sciencedirect.com/science/article/pii/S221065022100105X`

[19] P. Joćko, B. M. Ombuki-Berman, A. P. Engelbrecht, Multi-guide particle swarm optimisation archive management strategies for dynamic optimisation problems, Swarm Intelligence 16 (2) (2022) 143–168.

[20] A. Madani, B. Ombuki-Berman, A. Engelbrecht, Decision space scalability analysis of multi-objective particle swarm optimization algorithms, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2021, pp. 2179–2186.

[21] M. R. Sierra, C. A. C. Coello, Improving pso-based multi-objective optimization using crowding, mutation and-dominance, in: Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2005, pp. 505–519.

[22] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. C. Coello, F. Luna, E. Alba, Smpso: A new pso-based metaheuristic for multi-objective optimization, in: Proceedings of the IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, IEEE, 2009, pp. 66–73.

[23] Q. Lin, J. Li, Z. Du, J. Chen, Z. Ming, A novel multi-objective particle swarm optimization with multiple search strategies, European Journal of Operational Research 247 (3) (2015) 732–744.

[24] X. Zhang, X. Zheng, R. Cheng, J. Qiu, Y. Jin, A competitive mechanism based multi-objective particle swarm optimizer with fast convergence, Information Sciences 427 (2018) 63–76.

[25] M. A. Potter, K. A. De Jong, A cooperative coevolutionary approach to function optimization, in: Proceedings of the International Conference on Parallel Problem Solving from Nature, Springer, 1994, pp. 249–257.

[26] X. Li, X. Yao, Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 1546–1553.

[27] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Transactions on Evolutionary Computation 16 (2) (2011) 210–224.

[28] R.-L. Tang, Z. Wu, Y.-J. Fang, Adaptive multi-context cooperatively coevolving particle swarm optimization for large-scale problems, Soft Computing 21 (16) (2017) 4735–4754.

[29] F. Van den Bergh, A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 225–239.

[30] Y.-j. Shi, H.-f. Teng, Z.-q. Li, Cooperative co-evolutionary differential evolution for function optimization, in: International Conference on Natural Computation, Springer, 2005, pp. 1080–1088.

[31] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE, 1998, pp. 69–73.

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

[33] K. Erwin, A. Engelbrecht, Control parameter sensitivity analysis of the multi-guide particle swarm optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2019, pp. 22–29.

[34] C. He, R. Cheng, D. Yazdani, Adaptive offspring generation for evolutionary large-scale multiobjective optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52 (2) (2022) 786–798. doi:10.1109/TSMC.2020.3003926.

[35] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, X. Yao, Accelerating large-scale multiobjective optimization via problem reformulation, IEEE Transactions on Evolutionary Computation 23 (6) (2019) 949–961.

[36] B. Cao, J. Zhao, Y. Gu, Y. Ling, X. Ma, Applying graph-based differential grouping for multiobjective large-scale optimization, Swarm and Evolutionary Computation 53 (2020) 100626.

[37] N. Hansen, The cma evolution strategy: A tutorial, arXiv preprint arXiv:1604.00772 (2016).

[38] H. Chen, R. Cheng, J. Wen, H. Li, J. Weng, Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations, Information Sciences 509 (2020) 457–469.

[39] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Transactions on Cybernetics 45 (2) (2014) 191–204.

[40] S. Cheng, H. Zhan, H. Yao, H. Fan, Y. Liu, Large-scale many-objective particle swarm optimizer with fast convergence based on alpha-stable mutation and logistic function, Applied Soft Computing 99 (2021) 106947.

[41] H. Haklı, H. Uğuz, A novel particle swarm optimization algorithm with levy flight, Applied Soft Computing 23 (2014) 333–345.

[42] H. Hiba, A. A. Bidgoli, A. Ibrahim, S. Rahnamayan, Cgde3: An efficient center-based algorithm for solving large-scale multi-objective optimization problems, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 350–358.

[43] S. Kukkonen, J. Lampinen, An extension of generalized differential evolution for multi-objective optimization with constraints, in: Proceedings of the International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 752–761.

[44] S. Kukkonen, J. Lampinen, Gde3: The third evolution step of generalized differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation, Vol. 1, IEEE, 2005, pp. 443–450.

[45] J. Lampinen, et al., De's selection rule for multiobjective optimization, Lappeenranta University of Technology, Department of Information Technology, Tech. Rep (2001) 03–04.

[46] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.

[47] R. M. Rizk-Allah, A. E. Hassanien, A. Slowik, Multi-objective orthogonal opposition-based crow search algorithm for large-scale multi-objective optimization, Neural Computing and Applications 32 (17) (2020) 13715–13746.

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

[49] H. R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vol. 1, IEEE, 2005, pp. 695–701.

[50] L. M. Antonio, C. A. C. Coello, M. A. R. Morales, S. G. Brambila, J. F. González, G. C. Tapia, Coevolutionary operations for large scale multi-objective optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–8.

[51] M. N. Omidvar, B. Kazimipour, X. Li, X. Yao, Cbcc3—a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2016, pp. 3541–3548.

[52] M. N. Omidvar, X. Li, X. 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, 2011, pp. 1115–1122.

[53] M. Yang, M. N. Omidvar, C. Li, X. Li, Z. Cai, B. Kazimipour, X. Yao, Efficient resource allocation in cooperative co-evolution for large-scale global optimization, IEEE Transactions on Evolutionary Computation 21 (4) (2016) 493–505.

[54] M. Yang, A. Zhou, C. Li, J. Guan, X. Yan, Ccfr2: A more efficient cooperative co-evolutionary framework for large-scale global optimization, Information Sciences 512 (2020) 64–79.

[55] G.-G. Wang, Y. Tan, Improving metaheuristic algorithms with information feedback models, IEEE Transactions on Cybernetics 49 (2) (2017) 542–555.

[56] Z.-M. Gu, G.-G. Wang, Improving nsga-iii algorithms with information feedback models for large-scale many-objective optimization, Future Generation Computer Systems 107 (2020) 49–69.

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

[58] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[59] Y. Zhang, G.-G. Wang, K. Li, W.-C. Yeh, M. Jian, J. Dong, Enhancing moea/d with information feedback models for large-scale many-objective optimization, Information Sciences 522 (2020) 1–16.

[60] F. Van den Bergh, A. P. Engelbrecht, A new locally convergent particle swarm optimiser, in: IEEE International conference on systems, man and cybernetics, Vol. 3, IEEE, 2002, pp. 6–pp.

[61] H. Zille, H. Ishibuchi, S. Mostaghim, Y. Nojima, A framework for large-scale multiobjective optimization based on problem transformation, IEEE Transactions on Evolutionary Computation 22 (2) (2017) 260–275.

[62] Y. Tian, R. Cheng, X. Zhang, Y. Jin, Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum], IEEE Computational Intelligence Magazine 12 (4) (2017) 73–87.

[63] A. J. Nebro, J. J. Durillo, M. Vergne, Redesigning the jmetal multi-objective optimization framework, in: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, 2015, pp. 1093–1100.

[64] S. Huband, L. Barone, L. While, P. Hingston, A scalable multi-objective test problem toolkit, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2005, pp. 280–295.

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

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

[67] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable multi-objective optimization test problems, in: Proceedings of the IEEE Congress on Evolutionary Computation., volume=1, pages=825–830, year=2002, organization=IEEE.

[68] R. Cheng, Y. Jin, M. Olhofer, et al., Test problems for large-scale multiobjective and many-objective optimization, IEEE transactions on cybernetics 47 (12) (2016) 4108–4121.

[69] C. A. C. Coello, M. R. Sierra, A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm, in: Proceedings of the Mexican International Conference on Artificial Intelligence, Springer, 2004, pp. 688–697.

[70] M. R. Sierra, C. A. C. Coello, A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms, Technical Report of CINVESTAV-IPN (2004).

[71] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms—a comparative case study, in: Proceedings of the International Conference on Parallel Problem Solving from Nature, Springer, 1998, pp. 292–301.

[72] H. Ishibuchi, N. Akedo, Y. Nojima, Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems, IEEE Transactions on Evolutionary Computation 19 (2) (2014) 264–283.

[73] H. Ishibuchi, Y. Setoguchi, H. Masuda, Y. Nojima, Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes, IEEE Transactions on Evolutionary Computation 21 (2) (2016) 169–190.

[74] H. Ishibuchi, R. Imada, Y. Setoguchi, Y. Nojima, How to specify a reference point in hypervolume calculation for fair performance comparison, Evolutionary Computation 26 (3) (2018) 411–440.

[75] H. Seada, K. Deb, A unified evolutionary optimization procedure for single, multiple, and many objectives, IEEE Transactions on Evolutionary Computation 20 (3) (2015) 358–369.

[76] J. Maltese, B. M. Ombuki-Berman, A. P. Engelbrecht, A scalability study of many-objective optimization algorithms, IEEE Transactions on Evolutionary Computation 22 (1) (2016) 79–96.

[77] T. Wagner, N. Beume, B. Naujoks, Pareto-, aggregation-, and indicator-based methods in many-objective optimization, in: Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2007, pp. 742–756.

[78] P. E. McKnight, J. Najab, Mann-whitney u test, The Corsini Encyclopedia of Psychology (2010) 1–1.

[79] M. Helbig, A. P. Engelbrecht, Analysing the performance of dynamic multi-objective optimisation algorithms, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1531–1539.

[80] H. Zille, S. Mostaghim, Comparison study of large-scale optimisation techniques on the lsmop benchmark functions, in: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2017, pp. 1–8.

[81] M. N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Transactions on Evolutionary Computation 18 (3) (2013) 378–393.

[82] M. N. Omidvar, M. Yang, Y. Mei, X. Li, X. Yao, Dg2: A faster and more accurate differential grouping for large-scale black-box optimization, IEEE Transactions on Evolutionary Computation 21 (6) (2017) 929–942.

[83] J. Douglas, A. Engelbrecht, B. Ombuki-Berman, Merging and decomposition variants of cooperative particle swarm optimization: New algorithms for large scale optimization problems, in: Proceedings of the 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, 2018, pp. 70–77.

[84] J. Maltese, B. Ombuki-Berman, A. Engelbrecht, Co-operative vector-evaluated particle swarm optimization for multi-objective optimization,

in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 1294–1301.

[85] M. Helbig, Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation, Ph.D. thesis, University of Pretoria (2012).

Appendix A. Supplementary Experiments: The Effects of Using Multiple Context Vectors (The Seesaw Effect)

This section presents a small set of experiments designed to study the following:

- The shortcomings of optimizing different objectives individually in MGPSO, especially for multi-modal problems.
- The benefits of using multiple context vectors and sharing them among objectives in CCMGPSO, especially for multi-modal problems.

A potential problem with the original MGPSO implementation is the use of individual objective values to update the personal best and the global best positions. For single-objective optimization problems where the ultimate goal is to minimize one objective, the use of this single-objective approach is more justified. However, for multi-objective optimization, this may not always be advantageous. For example, for the 2-objective WFG4 the objective vector $(0, 4)$ is a boundary point of the true POF. However, if a particle in the first sub-swarm (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 sub-swarm 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, for multi-modal problems the performance of MGPSO may sill take a hit if no sub-swarms find high-quality Pareto-optimal solutions. In this section, the motivation behind using multiple context vectors alongside objective switching is discussed through a small experimental study. The main inspiration for using multiple context vectors was drawn from AM-CCPSO [28], where multiple context vectors were used for large-scale single-objective optimization. Moreover, Maltese *et al.* [84] proposed CVEPSO by incorporating the cooperative framework into the vector-evaluated particle swarm optimization (VEPSO) [85]. Although the work in [84] used separate context vectors per objective, the context vectors were shared between different sub-swarms through a knowledge transfer strategy (KTS), enabling the different objectives of a problem to share context vectors.

Appendix A.1. Experimental Setup and Results

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* to one context vector in the sense that for each objective a specific context vector was used throughout the entire optimization process and the context vectors were not shared among different objectives. The second experiment used n_m context vectors where each objective had access to all context vectors and the CPSO chose between them at random. Each experiment was run 20 times for 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 sub-swarms 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 updating the context vectors, because the objective values in the DTLZ problems can only accept non-negative values.

Appendix A.1.1. Results

The results for the 1500-dimensional 2- and 3-objective DTLZ functions are listed in Tables A.15 and A.16 respectively. Without any exceptions and for 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$_1$). For the multi-modal DTLZ1 this observation was more noticeable, CCMGPSO with fixed context vectors spent on average 98.34% and 99.23% of the entire duration of the search without updating the context vectors for two and three objectives respectively.
Performance wise, the numbers for CCMGPSO$_1$ in Tables A.15 and A.16 had several impacts. For multi-modal problems, CCMGPSO$_1$ was frequently trapped in local optima where the objective values were way worse than those of the true POF. For uni-modal problems, this impact on performance ranged from not being significant at all to finding a well-converged POF with with poor spread.
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) are provided in Figures A.11, A.12, and A.13 respectively. As seen in Figure A.11, the objective values obtained by CCMGPSO$_1$ were noticeably worse than those of CCMGPSO$_2$, and for the former a lot of the solutions were clustered in hyperplanes where individual objective values were close or equal to zero. Based on Figure A.11 and the numbers in Tables A.15 and A.16, one could surmise that for DTLZ1, CCMGPSO$_1$ initially found n_m objective vectors where the value of at least one objective was equal to zero (with poor values for other objectives), set them as context vectors, and never updated them until the end of the optimization process. For DTLZ2 (Figure A.12) which is a uni-modal problem, CCMGPSO$_1$ obtained well-converged solutions, but with visibly inferior distribution compared with those of CCMGPSO$_2$. As seen in Figure A.12, a noticeable set of the obtained solutions was clustered around the hyperplane where the value of $\mathbf{f}_1$ was close to zero. As seen in Figure A.13, for the 3-objective DTLZ7 (where the third objective is multi modal) CCMGPSO$_1$ had visibly worse distribution for the third objective compared with its counterpart (CCMGPSO$_2$).
Now the reason behind smaller numbers for CCMGPSO$_2$ in Tables A.15 and A.16 is explained using an example. Let $\hat{\mathbf{f}}$ be an arbitrary bi-objective minimization problem, where the minimum possible value for each objective is zero. Suppose after γ CPSO iterations on the first objective, the objective vector of the first context vector ($\widehat{\mathbf{C}}_1$) is equal to (0,2). Moreover, suppose for the next γ iterations $\mathbf{C}_1$ is assigned to the second objective and updated based on an improvement in the second objective, making $\widehat{\mathbf{C}}_1$ equal to $(1, 1.5)$. Because $\widehat{\mathbf{C}}_{11}$ (the first objective value of the first context vector) is no longer

Table A.15: 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$_1$ **(1)**	CCMGPSO$_2$ **(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%

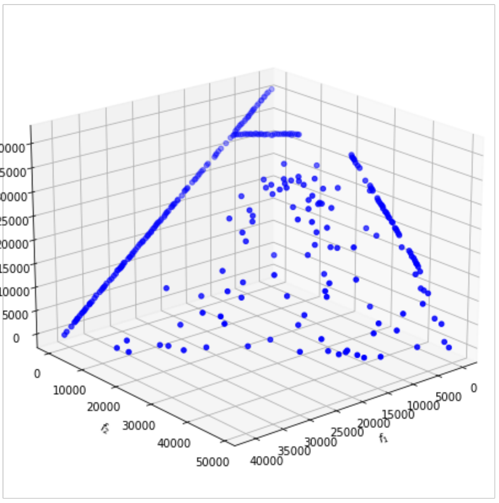
(1) CCMGPSO$_1$: CCMGPSO with n_m context vectors, where each objective only had access to one context vector,
(2) CCMGPSO$_2$: CCMGPSO with n_m context vectors, where each objective had access to all context vectors by randomly choosing between them.

equal to zero, more opportunities arise for the next time $\mathbf{C}_1$ is assigned to the first objective. As previously explained and seen in Tables A.15 and A.16, and Figure A.11, this seesaw-like pattern could specifically be of use for multi-modal problems. In order to further address the issue of stagnant global best positions in MGPSO, a dominance-based update mechanism is also incorporated into CCMGPSO (lines 13-17 in Algorithm 4). Inspired by the personal best update approach in OMOPSO [21], if the corresponding value of the new vector is equal to that of the context vector, the context vector is replaced by the new solution if its objective vector does not dominate that of the new solution.

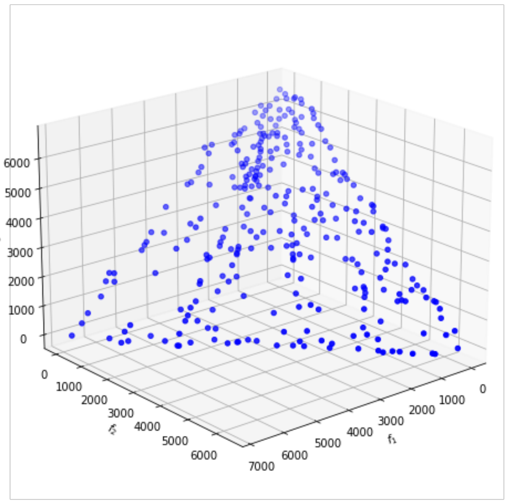
Table A.16: The effect of multiple context vectors with objective swapping on the 3-objective 1500-dimensional DTLZ problems, the listed values are the mean (over 20 independent runs) percentage of the total number of iterations where the corresponding objective of the assigned context vector was equal to zero

	Experiment	
Problem (Three Objectives)	CCMGPSO$_1$ **(1)**	CCMGPSO$_1$ **(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$_1$: CCMGPSO with n_m context vectors, where each objective only had access to one context vector,
(2) CCMGPSO$_2$: CCMGPSO with n_m context vectors, where each objective had access to all context vectors by randomly choosing between them.

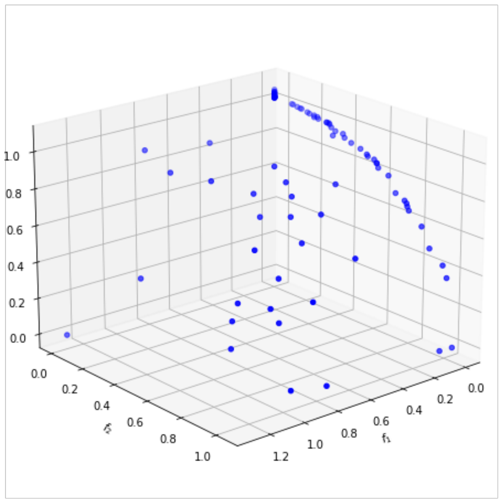


(a) CCMGPSO with fixed context vectors

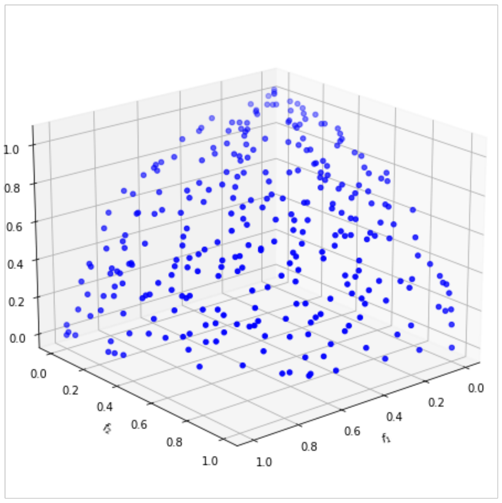


(b) CCMGPSO with shared context vectors

Figure A.11: The effect of sharing multiple context vectors between objectives on the 1500-dimensional 3-objective DTLZ1

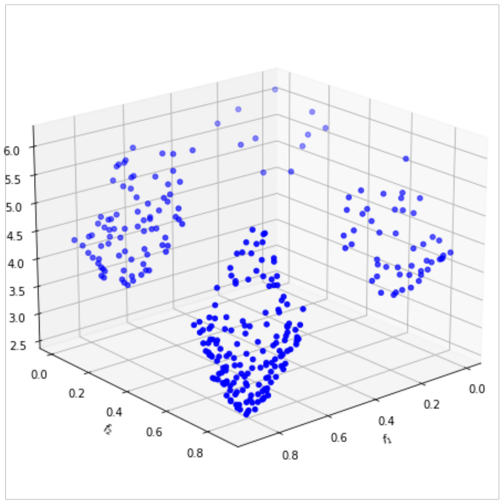


(a) CCMGPSO with fixed context vectors

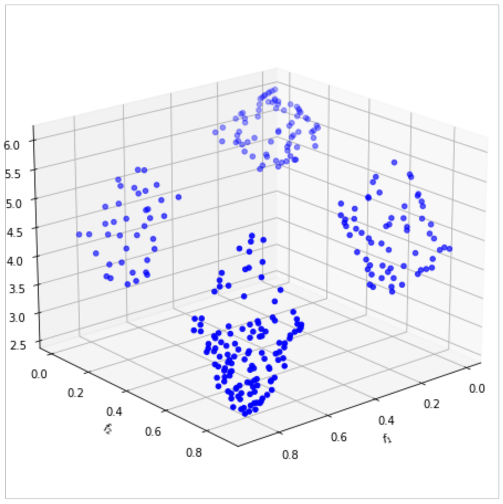


(b) CCMGPSO with shared context vectors

Figure A.12: The effect of sharing multiple context vectors between objectives on the 1500-dimensional 3-objective DTLZ2



(a) CCMGPSO with fixed context vectors



(b) CCMGPSO with shared context vectors

Figure A.13: The effect of sharing multiple context vectors between objectives on the 1500-dimensional 3-objective DTLZ7