2015 IEEE Symposium Series on Computational Intelligence

High-Dimensional Multi-objective Optimization Using Co-operative Vector-Evaluated Particle Swarm Optimization With Random Variable Grouping

Justin Maltese
Department of Computer Science
Brock University
St, Catharines, ON, Canada
jm10lh@brocku.ca

Andries. P. Engelbrecht
Department of Computer Science
University of Pretoria
Pretoria, South Africa
engel@cs.up.ac.za

Beatrice M. Ombuki-Berman
Department of Computer Science
Brock University
St Catharines, ON, Canada
bombuki@brocku.ca

***Abstract*—Vector-evaluated particle swarm optimization (VEPSO) is a particle swarm optimization (PSO) variant which employs multiple swarms to solve multi-objective optimization problems (MOPs). Each swarm optimizes a single objective and information is passed between swarms using a knowledge transfer strategy (KTS). The recently proposed co-operative VEPSO (CVEPSO) algorithm has been shown to improve the performance of VEPSO by decomposing the search space into subspaces of lower dimensionality. However, the effectiveness of CVEPSO is heavily dependent on the strategy used to group variables together, because isolating dependent variables leads to performance degradation. This paper explores the use of a random grouping technique within CVEPSO to increase the probability of allocating interacting variables to the same subcomponent. The results demonstrate that random grouping significantly improves performance of the CVEPSO algorithm, especially in high-dimensional environments. Additionally, CVEPSO with random grouping is shown to perform competitively with other top multi-objective optimization algorithms.**

I. Introduction

Particle swarm optimization (PSO) is a stochastic optimization algorithm modelled after the behaviours observed in bird flocks. Although the original PSO was designed solely for problems with one objective, it has been adapted for application to multi-objective problems (MOPs) [1], [2], [3]. One of the first applications of PSO to MOPs was proposed by Parsopoulous and Vrahatis [2] as the vector-evaluated particle swarm optimization (VEPSO) algorithm. VEPSO employs multiple swarms to solve MOPs by assigning a single swarm to optimize each objective. The problem is also optimized as a whole, as information is passed between swarms using a knowledge transfer strategy (KTS).

The co-operative VEPSO (CVEPSO) algorithm, introduced in [4], utilizes a co-operative coevolution (CC) method to improve the performance of VEPSO. CC approaches consist of a *divide-and-conquer* optimization approach, partitioning the decision vector into groups of variables which can then be optimized in a co-operative fashion. The CVEPSO-S, CVEPSO-S_K and CVEPSO-H_K variants were developed in [4], each based on a unique co-operative PSO (CPSO) approach developed by van den Bergh and Engelbrecht [5].

While each CVEPSO variant has been shown to perform well [4], success is limited by the static variable grouping methods used. Dependent variables have a low probability of being grouped together, thus leading to performance degradation. Optimizing interacting variables together is highly desirable, however since variable relationships are not known *a priori* this is a non-trivial task.

Yang *et al.* [6], [7] used a dynamic partitioning strategy based on a random grouping technique to address the interdependent variable problem. A differential evolution (DE) model was combined with this strategy and performance was evaluated on high-dimensional problems possessing a single objective. It was shown that random grouping elevates the probability that dependent variables will be grouped together. Li and Yao [8] applied the random grouping technique to the CPSO algorithm, evaluating performance against existing CPSO algorithms which incorporate static grouping strategies [5]. CPSO performed significantly better when utilizing a random grouping strategy, especially on functions with 500 or more dimensions.

This paper investigates whether the conclusions found by Li and Yao in [8] can be generalized to MOPs. Specifically, does random grouping serve as a better decomposition strategy for CVEPSO than those found in [4]? If so, is the performance gain more pronounced when faced with problems containing a higher number of dimensions? Another objective is to observe whether random grouping degrades performance for problems that are separable. To accomplish this, the performance of CVEPSO with random grouping is compared to the existing CVEPSO variants. Experiments are performed using nine diverse MOPs. CVEPSO with random grouping is also compared with well-known multi-objective PSO optimization algorithms.

The remainder of this paper is organized as follows: Section II contains background information about multi-objective optimization and algorithms such as PSO, VEPSO and CPSO. Section III provides an explanation of how random grouping is applied to the CVEPSO algorithm. Section IV describes the experimental setup used in this study. Section V presents the results of all experiments performed, including an analysis and a discussion of the observations. Finally, Section VI concludes the paper and suggests avenues for future research.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.186

II. Background

This section presents introductory background information pertaining to concepts used within this work. Topics covered include multi-objective optimization, PSO, CPSO, VEPSO and CVEPSO.

A. Multi-objective Optimization

Frequently, real-world optimization problems require simultaneous optimization of multiple distinct objectives. It is common for these objectives to conflict directly, presenting a challenging task for optimizers when dealing with such problems. A formal definition of a MOP is

$$\begin{aligned} &\text{minimize } \vec{f}(\vec{x}) \\ &\text{subject to } \vec{x} \in [x_{min}, x_{max}]^{n_x} \end{aligned}$$

where $\vec{f}(\vec{x}) = f_1(\vec{x}), f_2(\vec{x}), ..., f_{n_c}(\vec{x}))$, $\vec{x} = (x_1, x_2, ..., x_{n_x})$; n_c refers to the number of objectives and n_x is the search space dimensionality.

Within the context of multi-objective optimization, the term *Pareto optimal* is used to refer to solutions which cannot improve any objective further without worsening any other objective. Obtaining the set of Pareto optimal solutions, referred to as the Pareto front, is the end goal of a multi-objective optimization algorithm. Solutions which do not belong to the Pareto front are undesirable due to the fact that one or more objectives can be further optimized without degrading other objectives. The Pareto front is commonly analyzed by a decision maker with domain knowledge, as solutions belonging to the Pareto front contain trade-offs in one or more objectives.

B. Particle Swarm Optimization

PSO is a metaheuristic optimization algorithm modelled after the flocking behavior of birds. Candidate solutions are represented as particles within the PSO algorithm. PSO attempts to iteratively improve a swarm of particles by moving their positions towards the position of the best candidate solution ever found by the particle and the best candidate solution ever found by the particle's neighbourhood. Initial particle positions are generated randomly within the boundaries of the search space.

Pseudocode for the standard global best (GBest) PSO algorithm is shown in Algorithm 1. Particle positions are updated in a synchronous fashion [9]. One should note that this pseudocode utilizes a star neighbourhood topology [10], employing the social component to draw particles towards the overall best position of the entire swarm. It is also possible to create neighbourhoods of particle attraction via the use of a ring topology [11]. Algorithm 1 makes use of two update equations on lines 12 and 13 defined as:

$$S.\vec{v}_i(t+1) = \omega S.\vec{v}_i(t) + c_1\vec{r}_1(S.\vec{y}_i(t) - S.\vec{x}_i(t)) + c_2\vec{r}_2(S.\hat{\vec{y}}(t) - S.\vec{x}_i(t)) \quad (1)$$

$$S.\vec{x}_i(t+1) = S.\vec{x}_i(t) + S.\vec{v}_i(t+1) \quad (2)$$

where $S.\vec{x}$ corresponds to the current position of particle i in swarm S, $S.\vec{v}_i$ refers to the velocity of particle i in swarm S, $S.\vec{y}_i$ is the personal best position of particle i in swarm S, $S.\hat{\vec{y}}$ is the global best position of swarm S, ω is the inertia weight, c_1 is the cognitive weight, c_2 represents the social weight, and $\vec{r}_1$ and $\vec{r}_2$ are vectors consisting of uniformly distributed random numbers within the range [0,1].

Algorithm 1 Standard GBest PSO

```
1: Create and initialize a swarm, S, with candidate solutions in $n_x$ dimensions
2: while $termination\ criterion\ not\ satisfied$ do
3:     for each particle $i$ in $S$ do
4:         if $f(S.\vec{x}_i) < f(S.\vec{y}_i)$ then
5:             $S.\vec{y}_i = S.\vec{x}_i$
6:         end if
7:         if $f(S.\vec{y}_i) < f(S.\hat{\vec{y}})$ then
8:             $S.\hat{\vec{y}} = S.\vec{y}_i$
9:         end if
10:    end for
11:    for each particle $i$ in $S$ do
12:        Update velocity of particle $i$ using Equation (1)
13:        Update position of particle $i$ using Equation (2)
14:    end for
15: end while
```

C. Co-operative Particle Swarm Optimization

Traditional PSO algorithms, along with many other stochastic optimization algorithms, experience significant performance degradation when faced with high-dimensional problems [12]. Van den Bergh and Engelbrecht addressed this problem in [13] by proposing a new variant of PSO based on concepts observed in co-operative systems, referred to as the co-operative PSO. This variant was later renamed to co-operative split PSO (CPSO-S).

CPSO-S builds upon the original PSO algorithm by decomposing the search space into lower dimensional subspaces. The single swarm which is attempting to optimize a vector of n_x dimensions is split into n_x subswarms each optimizing a single dimension. An overview of the CPSO-S algorithm is presented in Algorithm 2.

For particle quality evaluation purposes, CPSO-S maintains a vector consisting of global best position components of each subswarm, referred to as a *context vector* [14]. Initially, the context vector contains random dimensions selected from each subswarm. Within Algorithm 2, b represents a function which takes a subswarm index and particle as input, returning a vector via the following process:

1) Clone the context vector. The cloned vector is denoted as $\vec{z}$.
2) Within $\vec{z}$, substitute the value at index l with the current positional value of particle i.
3) Return $\vec{z}$.

Using CPSO-S over standard PSO has several advantages. Solution quality is evaluated after each vector component is updated rather than when the entire vector has changed as seen in traditional PSO. This helps to prevent the PSO algorithm from experiencing the "two steps forward, one step back" phenomenon [5]. Additionally, diversity of solutions is improved as a result of the large number of solution combinations formed using different members of CPSO-S subswarms [5].

Algorithm 2 CPSO-S

```
 1: Create and initialize $n_x$ 1-dimensional subswarms
 2: while $termination\ criterion\ not\ satisfied$ do
 3:     for each subswarm $S_l$, $l = 1, ..., n_x$ do
 4:         for each particle $i$ in $S_l$ do
 5:             if $f(b(l, S_l.\vec{x}_i)) < f(b(l, S_l.\vec{y}_i))$ then
 6:                 $S_l.\vec{y}_i = S_l.\vec{x}_i$
 7:             end if
 8:             if $f(b(l, S_l.\vec{y}_i)) < f(b(l, S_l.\vec{\hat{y}}))$ then
 9:                 $S_l.\vec{\hat{y}} = S_l.\vec{y}_i$
10:             end if
11:         end for
12:     end for
13:     for each subswarm $S_l$, $l = 1, ..., n_x$ do
14:         for each particle $i$ in $S_l$ do
15:             Update velocity of particle i using Equation (1)
16:             Update position of particle i using Equation (2)
17:         end for
18:     end for
19: end while
```

Two additional CPSO variants, CPSO-S_K and CPSO-H_K, were proposed in [5] as a result of various shortcomings of the original CPSO-S algorithm. In an attempt to avoid optimizing dependent decision variables in isolation, the CPSO-S_K algorithm blindly splits the n_x-dimensional search space into k parts. The CPSO-H_K algorithm addresses the stagnation problem experienced by CPSO-S and CPSO-S_K when presented with deceptive functions. By executing both CPSO-S_K and regular PSO sequentially, CPSO-H_K is able to overcome deceptive locations in the search space [5]. A KTS is used within CPSO-H_K to exchange information between the CPSO-S_K and PSO swarms. This information exchange is a form of co-operation described in [15]. To transfer knowledge, the highest quality solution from the CPSO-S_K swarm is injected into the PSO swarm and vice-versa. Injection requires overwriting a particle's dimensions by replacing them with the dimensions of a target particle. However, constraints [13] as to which particles can be overwritten exist to ensure that the CPSO-S_K and PSO algorithms maintain a diverse set of solutions. These constraints are as follows:

1) The particle chosen can not be the best particle of a CPSO-S_K subswarm or the PSO swarm.
2) A *legal candidate* is defined as a particle which can be overwritten during knowledge transfer. A particle which is not a legal candidate is explicitly protected from overwrites. Only half of the total particles are considered legal candidates, selected randomly prior to optimization.

It is worth noting that other KTSs are possible, since the knowledge flow of CPSO-H_K is flexible. It is also possible to structure CPSO-H_K as an algorithm which executes CPSO-S_K until CPSO-S_K has become trapped, and then switches to PSO [13]. However, for real-world problems it would be difficult to design heuristics capable of detecting when a switch is appriopriate. Additional work on applying co-operative principles to PSO can be found in [16], [17], [18].

D. Vector Evaluated Particle Swarm Optimization

The VEPSO algorithm, proposed by Parsopoulos and Vrahatis in [2], serves as a PSO variant for solving MOPs. Particle fitness in VEPSO is evaluated as a vector of sub-objective fitnesses as opposed to a single scalar value. VEPSO utilizes the concept of Pareto domination by implementing a structure, referred to as the *archive*, to store non-dominated solutions for future usage. The archive typically has a maximum capacity defined by the user prior to optimization. Upon reaching the archive size limit, solutions are removed either randomly or according to some user-defined removal criteria.

One of the defining features of VEPSO is that multiple swarms are employed, where each swarm is given its own objective to optimize. Information is passed from swarm to swarm using a KTS, ensuring that the problem as a whole is optimized. Proper selection of the KTS is crucial to the performance of VEPSO, as Matthysen *et al.* [19] have shown that certain strategies are prone to stagnation.

Within VEPSO, a KTS defines how global guides for each swarm are determined. A global guide is defined to be a particle whose dimensions are substituted in Equation (1) for $\vec{\hat{y}}$, essentially replacing the global best particle position. Global guides play a crucial role within VEPSO, allowing swarms to optimize multiple objectives simultaneously. Several existing KTSs for VEPSO are defined in [2], [20], [21].

E. Co-operative Vector Evaluated Particle Swarm Optimization

As a result of the performance gain observed for co-operative PSO models [5], the CVEPSO algorithm was introduced in [4] as an extension to the VEPSO algorithm. CVEPSO incorporates co-operative principles by performing optimization of each objective using a CPSO variant. The following algorithms are concrete implementations of CVEPSO:

- **CVEPSO-S**: CPSO-S subswarms are used to optimize each objective.
- **CVEPSO-S_K**: CPSO-S_K swarms are used to optimize each objective.
- **CVEPSO-H_K**: CPSO-H_K swarms are used to optimize each objective.

All CVEPSO variants employ KTSs in the same way as traditional VEPSO. The ring, random and PCX GBest KTSs require a "global best" particle to determine the global guides. Because there is no global best position in the CPSO variants, these KTSs use the context vector instead of a global best particle.

Due to the search space decompositioning of each CVEPSO variant, archive addition and maintenance are slightly altered. Whenever a context vector of a CVEPSO swarm is non-dominated with respect to all current solutions in the archive, that context vector is inserted into the archive. In the case of a full archive, a user-selected removal strategy is executed to return the archive to its size limit and the context vector is then added into the archive. Removal strategies should ideally promote solution diversity within the archive. Distance-based removal is an example of one such strategy.

III. Incorporating Random Grouping within Co-operative Vector-Evaluated Particle Swarm Optimization

Within CC models, it is highly desirable to place interdependent variables within the same groups when performing decomposition. Doing so ensures that these variables are optimized together, positively impacting optimization performance. Each existing CVEPSO variant [4] performs search space decomposition *a priori* and maintains variable groupings statically throughout optimization. If dependent variables are not initially grouped together, they will be erroneously considered independent of one another for the entirety of the optimization process. Problems which exhibit non-separability will then create significant obstacles for the optimizer.

To address this problem, we define a new CVEPSO variant which employs a dynamic random grouping technique for search space partitioning, denoted CVEPSO-$\mathrm{R_K}$. Similar to CVEPSO-$\mathrm{S_K}$, CVEPSO-$\mathrm{R_K}$ randomly groups the n_x-dimensional search space into k subgroups. However, this partitioning process is repeated at each iteration, dynamically changing the grouping structure during the optimization process.

To help illustrate the random grouping process, consider a problem with $n_x = 500$. Assuming $k = 10$ and 100 iterations are used, variables will be randomly re-grouped at each iteration into 10 groups of size $\frac{n_x}{k} = \frac{500}{10} = 50$. Thus, variables will be regrouped a total of 100 times, increasing the probability that dependent variables will be optimized together [7].

IV. Experimental Setup

This section covers topics such as performance measures, statistical analysis methodology, algorithm parameters and benchmark functions.

A. Performance Measures

The objective of performance measures used within this paper is to provide an unbiased assessment of performance without actually knowing the true Pareto front. This is done using common criteria such as distribution of the non-dominated solution set, number of different non-dominated solutions and similarity of the approximation front to a best known estimation. The performance measures used in this work incorporate several of these elements.

Hypervolume: Zitzler and Thiele [22] defined the hypervolume metric as a measure of the amount of space covered by a set. The hypervolume metric consists of a single scalar value, calculated as the sum of sub-cuboids created by solution points. It was shown in [23] that the hypervolume metric is maximized when the solution set consists of equidistant Pareto-optimal points. Additionally, the hypervolume metric can be used to prove that a solution set is not worse than some other target solution set for all solution pairs [24].

Solution Distribution: The non-dominated solution distribution metric introduced in [25] provides an indication of the spacing density of the solutions along the discovered front. The solution distribution metric S is calculated as

$$S = \frac{1}{n_d}\sqrt{\frac{1}{n_d}\sum_{i=1}^{n_d}(d_i - \vec{d})^2}, \text{ with } \vec{d} = \frac{1}{n_d}\sum_{i=1}^{n_d} \quad (3)$$

where d_i represents the Euclidean distance between solution i and its closest neighbour in objective space and n_d is the number of non-dominated solutions. One should note that a larger number of solutions may produce a more desirable distribution score.

B. Statistical Methods

Pairwise Mann-Whitney-Wilcoxon rank sum tests [26] were used to test for a significant difference in performance between algorithms. For each pairwise test, if a statistically significant difference existed the algorithm with the higher mean over 30 independent runs was given a win and the algorithm with the lower mean was given a loss. A confidence level of 95% was used for each Mann-Whitney-Wilcoxon rank sum test.

C. Algorithm Parameters

The number of swarms used in each algorithm was equal to the number of objectives for the problem at hand. Each archive was restricted to 250 solutions, with solutions being removed when this size limit was reached. Selection of the solution to remove was distance-based to promote archive diversity. The two solutions with the smallest distance between each other were determined and one of them was removed randomly.

To ensure unbiased comparisons, each CVEPSO variant was assigned the same number of particles, i.e 50, before splitting into subswarms. When the subswarm split was performed based on the CVEPSO variant, the particles were divided as evenly as possible across all subswarms. Since the total number of particles used for all CVEPSO variants was equal, the number of fitness function evaluations per iteration were identical for each algorithm. Initial particle velocity was set to zero to comply with the recommendations given in [27]. Concerning the weight values, ω was set to 0.729844, c_1 was set to 1.496180 and c_2 was set to 1.496180. Note that these values were not set arbitrarily; they were chosen due to the fact that they ensure convergence [28]. To ensure that particles remain feasible, particles were re-initialized upon violating a boundary constraint.

The k value of CVEPSO-$\mathrm{S_K}$ and CVEPSO-$\mathrm{R_K}$ was set to 6 for all experiments, motivated by previous work in [5] and validated empirically. The CPSO-$\mathrm{S_K}$ swarms within CVEPSO-$\mathrm{H_K}$ also used a k value of 6.

D. Benchmark Suites

The Walking Fish Group (WFG) suite, introduced in [29], contains a set of nine minimization problems that are defined in terms of a vector of parameters. This vector is derived through a series of transition and shape functions, creating uniquely shaped Pareto fronts. Functions within the WFG suite are diverse, incorporating challenging shapes and modalities to help simulate real-world function landscapes.

TABLE I. Overview of WFG Functions Used

Function	**Separability**	**Bias**	**Shape**	**Modality**
1	Separable	Polynomial, Flat	Convex	Uni
2	Non-Separable	-	Convex, Disconnected	Uni, Multi
3	Non-Separable	-	Linear, Degenerate	Uni
4	Separable	-	Concave	Multi
5	Separable	-	Concave	Deceptive
6	Non-Separable	-	Concave	Uni
7	Separable	Parameter Dependent	Concave	Uni
8	Non-Separable	Parameter Dependent	Concave	Uni
9	Non-Separable	Parameter Dependent	Concave	Multi, Deceptive

The WFG functions used within this work possess a total of three objectives and p decision parameters. The value of p is experiment-dependent, with values of 30 and 500 seen within the work. An overview of the WFG function set is given in Table I. Additional information on problems contained within the WFG suite can be found in [29], [30].

V. Experimental Results and Discussion

Within this section, experimental results are presented and discussed. The CVEPSO-S, CVEPSO-S_K and CVEPSO-H_K algorithms were compared to the proposed CVEPSO-R_K algorithm using the hypervolume and solution distribution metrics. Experiments were performed over the WFG function set using 30 dimensions and 500 dimensions. The PCX GBest KTS [21] is used to transfer knowledge between swarms in all experiments as it has been shown to perform well with CVEPSO variants [4].

Several figures presented in this section reference the term *difference*, which is simply the difference between pairwise wins and losses (described in Section IV. B) for each algorithm. Additionally, the *rank* of each algorithm is shown which denotes performance ranking in comparison to all other algorithms with respect to WFG function performance.

A. 30-dimensional functions

Table II presents an overview of algorithmic performance using functions of 30 dimensions, while Table III displays a statistical summary of the information present in Table II. Table II shows that the CVEPSO-R_K algorithm was dominant with respect to the hypervolume metric. CVEPSO-R_K performed significantly better than all other algorithms for all functions except WFG4, WFG7 and WFG9. It is noted that both WFG4 and WFG7 are separable functions which are not impacted as much by the random grouping mechanism, likely why CPSO-S outperformed CVEPSO-R_K for both functions.

Observing the performances of CVEPSO-S_K and CVEPSO-R_K in Table II and III, it is clear that the added random grouping present in CVEPSO-R_K led to a more desirable hypervolume in nearly all cases. CVEPSO-R_K possessed the lowest mean rank of all algorithms, indicating that it was consistently the best CVEPSO variant. CVEPSO-S was the second best variant, performing better than both CVEPSO-S_K and CVEPSO-H_K as evidenced by Table II. The CVEPSO-H_K struggled most often, yielding poor hypervolume performance in comparison to the other CVEPSO variants.

The proposed CVEPSO-R_K algorithm also distributed non-dominated solutions exceptionally well. Both

TABLE II. Mann-Whitney Wins and Losses Over The WFG Functions Using 30 Dimensions

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
CVEPSO-R_K	Hypervolume	Wins	2	3	3	1	3	3	1	3	2
		Losses	1	0	0	1	0	0	2	0	1
		Difference	+1	+3	+3	0	+3	+3	-1	+3	+1
		Rank	1	1	1	2	1	1	2	1	2
	Distribution	Wins	2	2	3	0	3	0	1	3	0
		Losses	0	1	0	1	0	1	1	0	0
		Difference	+2	+1	+3	-1	+3	-1	0	+3	0
		Rank	1	2	1	3	1	2	2	1	2
CVEPSO-S	Hypervolume	Wins	0	1	2	3	2	2	2	2	3
		Losses	2	2	1	0	1	1	0	1	0
		Difference	-2	-1	+1	+3	+1	+1	+2	+1	+3
		Rank	3	3	2	1	2	2	1	2	1
	Distribution	Wins	0	3	2	3	2	3	3	2	1
		Losses	3	0	1	0	1	0	0	1	0
		Difference	-3	+3	+1	+3	+1	+3	+3	+1	+1
		Rank	4	1	2	1	2	1	1	2	1
CVEPSO-S_K	Hypervolume	Wins	2	2	1	1	1	1	2	1	1
		Losses	1	1	2	1	2	2	0	2	2
		Difference	+1	+1	-1	0	-1	-1	+2	-1	-1
		Rank	1	2	3	2	3	3	1	3	3
	Distribution	Wins	1	1	0	0	1	0	0	0	0
		Losses	2	2	2	2	2	1	1	3	1
		Difference	-1	-1	-2	-2	-1	-1	-1	-3	-1
		Rank	3	3	3	4	3	2	3	4	4
CVEPSO-H_K	Hypervolume	Wins	1	0	0	0	0	0	0	0	0
		Losses	2	3	3	3	3	3	3	3	3
		Difference	-1	-3	-3	-3	-3	-3	-3	-3	-3
		Rank	2	4	4	3	4	4	4	4	4
	Distribution	Wins	2	0	0	1	0	0	0	1	0
		Losses	0	3	2	1	3	1	2	2	0
		Difference	+2	-3	-2	0	-3	-1	-2	-1	0
		Rank	1	4	3	2	4	2	4	3	2

TABLE III. Algorithm Rank Summary For 30 Dimensions

		Algorithm			
Metric	**Measure**	CVEPSO-R_K	CVEPSO-S	CVEPSO-S_K	CVEPSO-H_K
Hypervolume	Mean	**1.333**	1.889	2.333	3.667
	Maximum	2	3	3	4
	Minimum	1	1	1	2
Distribution	Mean	**1.667**	**1.667**	3.222	2.778
	Maximum	3	4	4	4
	Minimum	1	1	2	1

CVEPSO-R_K and CVEPSO-S possessed the best rank on average with a mean ranking of 1.667. The separable WFG4 function presented the most difficulty for CVEPSO-R_K, as it was outperformed by both CVEPSO-S and CVEPSO-H_K. The CVEPSO-S_K algorithm performed noticeably poor, yielding a much less desirable distribution of non-dominated solutions in comparison to CVEPSO-S and CVEPSO-R_K. The large performance disparity between CVEPSO-S_K and CVEPSO-R_K implies that the random grouping method produced a more distributed set of non-dominated solutions with respect to the 30-dimensional WFG function set.

B. 500-dimensional functions

Results of experiments utilizing functions with 500 dimensions are given in Table IV. Table V displays the mean, maximum and minimum rankings of each algorithm in Table IV. Effectiveness of the random grouping method is much more pronounced than on 30-dimensional functions as evidenced by the overall performance of the CVEPSO-R_K algorithm. With respect to the hypervolume metric, CVEPSO-R_K outperformed every other CVEPSO variant for all functions aside from the separable WFG7 function. Random grouping was highly

TABLE IV. MANN-WHITNEY WINS AND LOSSES OVER THE WFG FUNCTIONS USING 500 DIMENSIONS

Algorithm	Metric	Result	WFG Function 1	2	3	4	5	6	7	8	9
CVEPSO-R_K	Hypervolume	Wins	3	3	3	3	3	3	1	3	3
		Losses	0	0	0	0	0	0	2	0	0
		Difference	+3	+3	+3	+3	+3	+3	-1	+3	+3
		Rank	1	1	1	1	1	1	2	1	1
	Distribution	Wins	2	3	3	3	3	0	1	3	0
		Losses	0	0	0	0	0	1	1	0	0
		Difference	+2	+3	+3	+3	+3	-1	0	+3	0
		Rank	1	1	1	1	1	2	2	1	2
CVEPSO-S	Hypervolume	Wins	0	1	2	1	2	2	2	2	1
		Losses	2	2	1	1	1	1	0	1	2
		Difference	-2	-1	+1	0	+1	+1	+2	+1	-1
		Rank	3	3	2	2	2	2	1	2	2
	Distribution	Wins	0	2	2	2	2	3	3	2	1
		Losses	3	1	1	1	1	0	0	1	0
		Difference	-3	+1	+1	+1	+1	+3	+3	+1	+1
		Rank	4	2	2	2	2	1	1	2	1
CVEPSO-S_K	Hypervolume	Wins	1	2	1	1	1	1	2	1	2
		Losses	2	1	2	2	2	2	0	2	1
		Difference	-1	+1	-1	-1	-1	-1	+2	-1	+1
		Rank	2	2	3	3	3	3	1	3	2
	Distribution	Wins	1	1	0	0	1	0	0	1	0
		Losses	2	2	2	3	2	1	1	2	1
		Difference	-1	-1	-2	-3	-1	-1	-1	-1	-1
		Rank	3	3	3	4	3	2	3	3	4
CVEPSO-H_K	Hypervolume	Wins	1	0	0	0	0	0	0	0	0
		Losses	2	3	3	3	3	3	3	3	3
		Difference	-1	-3	-3	-3	-3	-3	-3	-3	-3
		Rank	2	4	4	3	4	4	4	4	4
	Distribution	Wins	2	0	0	1	0	0	0	0	0
		Losses	0	3	2	2	3	1	2	3	0
		Difference	+2	-3	-2	-1	-3	-1	-2	-3	0
		Rank	1	4	3	3	4	2	4	4	2

TABLE V. ALGORITHM RANK SUMMARY FOR 500 DIMENSIONS

Metric	Measure	Algorithm CVEPSO-R_K	CVEPSO-S	CVEPSO-S_K	CVEPSO-H_K
Hypervolume	Mean	**1.111**	2.111	2.444	3.667
	Maximum	2	3	3	4
	Minimum	1	1	1	2
Distribution	Mean	**1.333**	1.889	3.111	3.000
	Maximum	2	4	4	4
	Minimum	1	1	2	1

effective when tasked with non-separable WFG functions, as CVEPSO-R_K outperformed every other algorithm for all functions that were not separable.

As shown in Table V, CVEPSO-R_K possessed the lowest mean ranking in comparison to all other CVEPSO variants, indicating that the algorithm scaled well to 500 dimensions. It is likely that far more variable relationships were captured by the dynamic random grouping strategy, placing interacting variables within the same subcomponent frequently. As a result, CVEPSO-R_K produced a significantly more desirable hypervolume on nearly every WFG function in comparison to the other CVEPSO variants. CVEPSO-S scaled notably bad as a result of considering dependent variables in isolation. The worst performing algorithm was CVEPSO-H_K, as it yielded the overall worst hypervolume for all functions aside from WFG1 and WFG4.

With regards to the non-dominated solution distribution metric, the performance of the CVEPSO-R_K algorithm improved considerably in comparison to the 30-dimensional experiments. Seen in Table V, the maximum ranking of CVEPSO-R_K was two, being outperformed by CVEPSO-S for WFG6, WFG7 and WFG9. CVEPSO-R_K possessed an overall mean ranking of 1.333, indicating that it consistently produced a better non-dominated solution spread than all other CVEPSO variants. CVEPSO-S_K was often outperformed by CVEPSO-R_K, indicating that the static grouping strategy of CVEPSO-S_K may experience difficulty in obtaining a diverse set of non-dominated solutions in high-dimensional environments. Overall, the observations present in the 30 and 500 dimension experiments provide strong empirical evidence for the use of random grouping within CVEPSO, especially in high-dimensional environments.

C. *Comparison to Other Algorithms*

While the analysis in sections V.A and V.B revealed random grouping to improve overall performance of the CVEPSO algorithm, it is desirable to discover how well CVEPSO-R_K fares against other well-known multi-objective optimization algorithms. For this purpose, the CVEPSO-R_K algorithm was compared with optimized multi-objective PSO (oMOPSO) [31] and speed constrained multi-objective PSO (SMPSO) [32]. An equal number of function iterations were used across all algorithms to avoid any bias, with 30 dimensions used for each WFG function. Tables VI and VII give insight into the results of these experiments.

With reference to the hypervolume metric, the CVEPSO-R_K algorithm consistently outperformed the SMPSO algorithm. CVEPSO-R_K produced equal or better hypervolume than SMPSO for all functions over the WFG suite. SMPSO generally performed the worst out of all three algorithms, achieving a mean rank of 2.556. Comparison of oMOPSO and CVEPSO-R_K was more even, with both algorithms possessing similar mean hypervolume ranks in Table VII. CVEPSO-R_K tended to perform better on non-separable functions (WFG6, WFG8, WFG9). However, considerable success was seen on separable functions as well (WFG1, WFG4, WFG5). CVEPSO-R_K performed worst on the separable WFG7 function while oMOPSO experienced considerable difficulty when presented with the non-separable WFG8 function.

A similar set of conclusions is seen for the non-dominated solution distribution metric. SMPSO generally produced a significantly less desirable spread of solutions in comparison to oMOPSO and CVEPSO-R_K as evidenced by its mean rank of 2.444 in Table VII. CVEPSO-R_K and oMOPSO yielded relatively equal non-dominated solution performance, with neither algorithm dominating the other. It is noted that oMOPSO again struggled severely for WFG8, indicating that this function may present significant problems for the oMOPSO algorithm. Contrary to this observation, CVEPSO-R_K did not seem to experience difficulties for any function in particular, indicating reliable and consistent performance. Table VII shows that the CVEPSO-R_K algorithm possesses the lowest mean ranking overall, performing slightly better than oMOPSO on average and considerably better than SMPSO. Overall, the CVEPSO-R_K is highly competitive with both oMOPSO and SMPSO, often performing better than both algorithms for various WFG functions.

VI. CONCLUSION

This work investigated the application of a dynamic random grouping technique to aid in optimizing dependent

TABLE VI. MANN-WHITNEY WINS AND LOSSES OVER THE WFG FUNCTIONS FOR VARIOUS MULTI-OBJECTIVE PSO ALGORITHMS

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
CVEPSO-R_K	Hypervolume	Wins	2	1	1	1	2	2	0	2	1
		Losses	0	1	1	0	0	0	2	0	0
		Difference	+2	0	0	+1	+2	+2	-2	+2	+1
		Rank	1	2	2	1	1	1	3	1	1
	Distribution	Wins	2	2	0	1	0	2	1	1	1
		Losses	0	0	1	1	1	0	0	1	1
		Difference	+2	+2	-1	0	-1	+2	+1	0	0
		Rank	1	1	2	2	2	1	1	2	2
oMOPSO	Hypervolume	Wins	0	2	2	1	0	1	2	0	1
		Losses	1	0	0	0	1	1	1	2	0
		Difference	-1	+2	+2	+1	-1	0	+1	-2	+1
		Rank	2	1	1	1	2	2	1	3	1
	Distribution	Wins	1	1	0	2	2	1	1	0	2
		Losses	1	1	1	0	0	1	0	2	0
		Difference	0	0	-1	+2	+2	0	+1	-2	+2
		Rank	2	2	2	1	1	2	1	3	1
SMPSO	Hypervolume	Wins	0	0	0	0	0	0	1	1	0
		Losses	1	2	2	2	1	2	1	1	2
		Difference	-1	-2	-2	-2	-1	-2	0	-1	-2
		Rank	2	3	3	3	2	3	2	2	3
	Distribution	Wins	0	0	2	0	0	0	0	2	0
		Losses	2	2	0	2	1	2	2	0	2
		Difference	-2	-2	+2	-2	-1	-2	-2	+2	-2
		Rank	3	3	1	3	2	3	3	1	3

TABLE VII. ALGORITHM RANK SUMMARY FOR VARIOUS MULTI-OBJECTIVE PSO ALGORITHMS

		Algorithm		
Metric	**Measure**	CVEPSO-R_K	oMOPSO	SMPSO
Hypervolume	Mean	**1.444**	1.556	2.556
	Maximum	3	3	3
	Minimum	1	1	2
Distribution	Mean	**1.556**	1.667	2.444
	Maximum	2	3	3
	Minimum	1	1	1

variables within the CVEPSO algorithm. The CVEPSO-R_K algorithm was formally proposed as a CVEPSO variant utilizing random variable grouping to repartition the search space at each iteration. The performance of the CVEPSO-R_K algorithm was compared to existing CVEPSO variants found in [4]. Experiments comparing CVEPSO-R_K to well-known multi-objective PSO algorithms were also performed.

Results demonstrated random grouping to be a powerful tool capable of enhancing the performance of CVEPSO. The proposed CVEPSO-R_K algorithm frequently outperformed all other existing CVEPSO variants in terms of the hypervolume and non-dominated solution distribution metrics. Dynamic random grouping was demonstrated to perform better than static grouping as evidenced by CVEPSO-R_K's significant outperformance of CVEPSO-S_K. The effectiveness of CVEPSO-R_K was even more pronounced in high-dimensional space, as the random grouping technique aided in optimizing dependent variables together. It was noted that the CVEPSO-S strategy experienced significant performance degradation in high-dimensional space as a result of considering dependent variables in isolation.

The CVEPSO-R_K algorithm was also observed to perform competitively with several well-known multi-objective PSO algorithms, namely oMOPSO and SMPSO. CVEPSO-R_K consistently possessed a more desirable hypervolume and non-dominated solution distribution in comparison to the SMPSO algorithm. Comparisons between CVEPSO-R_K and oMOPSO were much more competitive, as both algorithms experienced approximately the same level of performance. CVEPSO-R_K was observed to perform better on non-separable functions, likely due to its random grouping decomposition method. Overall, considerable empirical evidence within this work supports the use of random grouping within CVEPSO, especially on high-dimensional non-separable problems.

Several interesting opportunities for future work within this area exist. Arguably the most important research offshoot is further addressing the variable dependency problem within the co-operative coevolution model. Techniques which aim to learn variable relationships *a priori* can be used to partition the search space, grouping dependent variables together in subcomponents. Further research into the scalability of CVEPSO-R_K as the number of dimensions and/or objectives increases is also one area that can be investigated.

REFERENCES

[1] Y. Jin, M. Olhofer, and B. Sendhoff, "Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how?" in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*. Morgan Kaufmann, 2001, pp. 1042–1049. [Online]. Available: documents/edwa.pdf

[2] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, ser. SAC '02. New York, NY, USA: ACM, 2002, pp. 603–607. [Online]. Available: http://doi.acm.org/10.1145/508791.508907

[3] C. A. Coello Coello and M. S. Lechuga, "Mopso: A proposal for multiple objective particle swarm optimization," in *Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume 02*, ser. CEC '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 1051–1056. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251972.1252327

[4] J. Maltese, B. Ombuki-Berman, and A. P. Engelbrecht, "Co-operative vector-evaluated particle swarm optimization," 2015, submitted to IEEE SIS.

[5] F. Van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 3, pp. 225–239, June 2004.

[6] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, Sept 2007, pp. 3523–3530.

[7] ——, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985 – 2999, 2008, nature Inspired Problem-Solving. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002002550800073X

[8] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, May 2009, pp. 1546–1553.

[9] J. Rada-Vilela, M. Zhang, and W. Seah, "A performance study on synchronous and asynchronous updates in particle swarm optimization," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 21–28. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001581

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948.

[11] R. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools*. San Diego, CA, USA: Academic Press Professional, Inc., 1996.

[12] T. Hendtlass, “Particle swarm optimisation and high dimensional problem spaces,” in *Evolutionary Computation, 2009. CEC ’09. IEEE Congress on*, May 2009, pp. 1988–1994.

[13] F. van den Bergh and A. Engelbrecht, “Cooperative Learning in Neural Networks using Particle Swarm Optimizers,” pp. 84–90, 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.4265

[14] N. Unger, B. Ombuki-Berman, and A. Engelbrecht, “Cooperative particle swarm optimization in dynamic environments,” in *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, April 2013, pp. 172–179.

[15] S. H. Clearwater, T. Hogg, and B. A. Huberman, “Cooperative problem solving,” in *COMPUTATION: THE MICRO AND THE MACRO VIEW*. World Scientific, 1992, pp. 33–70.

[16] M. El-Abd, “Cooperative models of particle swarm optimizers,” Ph.D. dissertation, Waterloo, Ont., Canada, Canada, 2008, aAINR43264.

[17] M. El-Abd, H. Hassan, M. Anis, M. S. Kamel, and M. Elmasry, “Discrete cooperative particle swarm optimization for fpga placement,” *Appl. Soft Comput.*, vol. 10, no. 1, pp. 284–295, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2009.07.011

[18] K. E. Parsopoulos, “Parallel cooperative micro-particle swarm optimization: A masterslave model,” *Applied Soft Computing*, vol. 12, no. 11, pp. 3552 – 3579, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494612003134

[19] W. Matthysen, A. Engelbrecht, and K. Malan, “Analysis of stagnation behavior of vector evaluated particle swarm optimization,” in *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, April 2013, pp. 155–163.

[20] J. Grobler, A. P. Engelbrecht, and V. S. S. Yadavalli, “Multi-objective de and pso strategies for production scheduling,” in *2008 IEEE World Congress on Computational Intelligence*, J. Wang, Ed., IEEE Computational Intelligence Society. Hong Kong: IEEE Press, 1-6 June 2008, pp. –.

[21] K. R. Harrison, B. Ombuki-Berman, and A. P. Engelbrecht, “Knowledge transfer strategies for vector evaluated particle swarm optimization,” in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, R. Purshouse, P. Fleming, C. Fonseca, S. Greco, and J. Shaw, Eds. Springer Berlin Heidelberg, 2013, vol. 7811, pp. 171–184. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37140-0_16

[22] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach,” *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, Nov 1999.

[23] M. Fleischer, “The measure of pareto optima applications to multi-objective metaheuristics,” in *Proceedings of the 2Nd International Conference on Evolutionary Multi-criterion Optimization*, ser. EMO’03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 519–533. [Online]. Available: http://dl.acm.org/citation.cfm?id=1760102.1760146

[24] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, “Performance assessment of multiobjective optimizers: an analysis and review,” *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, April 2003.

[25] C. Goh and K. Tan, “An investigation on noisy environments in evolutionary multiobjective optimization,” *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 3, pp. 354–381, June 2007.

[26] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 03 1947. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177730491

[27] A. Engelbrecht, “Particle swarm optimization: Velocity initialization,” in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, June 2012, pp. 1–8.

[28] F. Van Den Bergh, “An analysis of particle swarm optimizers,” Ph.D. dissertation, Pretoria, South Africa, South Africa, 2002, aAI0804353.

[29] S. Huband, L. Barone, L. While, and P. Hingston, “A scalable multi-objective test problem toolkit,” in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, C. Coello Coello, A. Hernndez Aguirre, and E. Zitzler, Eds. Springer Berlin Heidelberg, 2005, vol. 3410, pp. 280–295. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31880-4_20

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

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

[32] A. Nebro, J. Durillo, J. Garcia-Nieto, C. Coello Coello, F. Luna, and E. Alba, “Smpso: A new pso-based metaheuristic for multi-objective optimization,” in *Computational intelligence in miulti-criteria decision-making, 2009. mcdm ’09. ieee symposium on*, March 2009, pp. 66–73.