Merging and Decomposition Variants of Cooperative Particle Swarm Optimization

Empirical Analysis

Jay Douglas
Brock University
St. Catharines, Ontario
jd11ps@brocku.ca

Andries Engelbrecht
Institute for Big Data and Data Science, University of Pretoria
Pretoria, South Africa
engel@cs.up.ac.za

Beatrice Ombuki-Berman
Brock University
St. Catharines, Ontario
bombuki@brocku.ca

ABSTRACT

Many optimization algorithms suffer under the curse of dimensionality - a problem that describes a decrease in performance as the number of problem variables increases. Particle swarm optimization (PSO) is no exception to this performance degradation. Cooperative methods have since been introduced for PSO in order to increase its effectiveness for high dimensional problems by following a divide and conquer approach to large dimensional problems. Performance still suffers for such cooperative PSO (CPSO) variants, mostly due to the dependencies among variables. This paper will demonstrate the performance of two new variants of CPSO, namely decomposition and merging cooperative swarm optimization, referred to as DCPSO and MCPSO respectively. The goal of these variants will be to improve performance for large scale problems with variable dependencies. Empirical results show that DCPSO and MCPSO were able to perform better than standard CPSO-S$_k$, CPSO-H$_k$ and a random grouping variant, CCPSO, for specific problem classes. While empirical results show that the best strategy between merging and decomposition is very much problem dependent, the MCPSO generally converged earlier than the DCPSO.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

Particle swarm optimization, cooperative particle swarm optimization, large-scale optimization, random grouping, merging, decomposition

ACM Reference Format:
Jay Douglas, Andries Engelbrecht, and Beatrice Ombuki-Berman. 2018. Merging and Decomposition Variants of Cooperative Particle Swarm Optimization: Empirical Analysis. In *Proceedings of 2018 2nd International Conference on Intelligent Systems (ISMSI 2018),* Andries Engelbrecht and Beatrice Ombuki-Berman (Eds.). ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The merit of any meta-heuristic comes from its ability to provide reasonably good solutions for a taxonomy of optimization problems. While most meta-heuristics will have little issue optimally solving lower dimensional problems, one class of problems that becomes increasingly difficult to optimize are large scale optimization problems (LSOPs). Examples of LSOPs can be seen in the training of support vector machines (SVMs) [11] or deep neural networks (DNNs) [9] where there is a large number of decision variables to optimize. Algorithms optimizing these problems are prone to suffering from the "curse of dimensionality", seen in the deterioration of performance as the number of decision variables increases. An additional issue that arises with LSOPs is the existence of variable dependencies, meaning that certain variables are reliant on the results of other variables, and thus should be optimized together. While variable dependencies can exist for smaller dimensional problems, the grouping of related variables is not an issue, as smaller dimensional problems do not suffer from the curse of dimensionality, allowing the algorithms to evaluate all decision variables together. Since the position update of an entire decision vector for LSOPs is often not optimal in regards to performance due to the curse of dimensionality, the problem space is often separated into smaller sub-spaces, thus certain related variables may be separated. As such, it is necessary for algorithms optimizing LSOPs to address both the curse of dimensionality as well as variable dependency.

Different approaches to LSOPs have been introduced, including both problem decomposition and non-decomposition based methods [5]. The most notable of the two is the divide-and-conquer approach which uses cooperative coevolution (CC) algorithms to decompose the search space into smaller sub-spaces. The first example of decomposition for high dimensional problems was the cooperative coevolutionary genetic algorithm (CCGA) proposed by Potter and De Jong [6]. The CC framework has since been adapted to various meta-heuristics such as genetic algorithms (GAs), differential evolution (DE), and particle swarm optimization (PSO). One such family of algorithms resulting from this work is known as cooperative particle swarm optimization (CPSO) proposed by Van den Bergh and Engelbrecht [10], which applies the divide-and-conquer approach to PSO. The paper proposed the generalized CPSO-S$_k$ algorithm which divides an n-dimensional problem amongst k subswarms, along with a hybrid CPSO-H$_k$ algorithm which alternates between running as a PSO and CPSO-S$_k$ with information exchange between the two. While proving to be an efficient way to combat the curse of dimensionality for separable problems, the static groupings of CPSO-S$_k$ and CPSO-H$_k$ saw worse performance for problems with higher degrees of variable dependence. As a result, variants of CPSO have been developed to improve performance under the presence of variable dependencies, such as the random grouping CCPSO algorithm proposed by Yang *et. al* [4], which works to randomly permutate dimensions every iteration amongst the sub-swarms.

Two new variants of CPSO, namely the decomposition CPSO (DCPSO) and the merging CPSO (MCPSO) have been introduced in the companion paper [2]. These algorithms address the curse of dimensionality problem by using the same divide-and-conquer approach as seen in CPSO-S$_k$, as well as variable dependencies by iteratively restructuring the sizes of sub-swarms, allowing for various groups of variables to be optimized together. While the companion paper has introduced these algorithms, this paper conducts an empirical analysis of DCPSO and MCPSO in comparison with the original CPSO-S$_k$ and CPSO-H$_k$ algorithms, as well as a random grouping variant known as CCPSO proposed by Yao and Li [4], which attempts to address variable dependency by randomly regrouping dimensions between a static number of sub-swarms.

Section 2 outlines the experimental details used to compare the performance of the various CPSO algorithms. A discussion of the results given in section 3.

2 EXPERIMENTAL SETUP

This section describes the empirical process followed to analyse the performance of the proposed CPSO algorithms. The performance measures monitored for each of the algorithms are given in section 2.1. Section 2.2 provides an overview of the statistical tests used to verify results. The actual test problems are highlighted in section 2.3. The parameters of each algorithm are detailed in section 2.4.

2.1 Performance Measures

In order to quantify the performance of each algorithm, the quality of the global best particle at the end of the optimization process was used. This measure reflects the accuracy of the algorithms.

2.2 Statistical Methods

Each algorithm was executed for 30 independent runs on each benchmark problem. A Mann-Whitney U-test was performed on every pair of algorithms for each problem at a 95% confidence level in order to establish whether or not the differences were statistically significant or not. The tests outcomes were used to compute the total number of wins and losses for each algorithm over different problem classes. The algorithms were then ranked based on the difference between their wins and losses.

2.3 Benchmark Problems

The problems used in this paper are a combination of unimodal, multimodal, separable, non-separable, and rotated problems, listed in table 1. Separable problems have no dependencies among their decision variables, allowing optimization of each decision variable in isolation. On the other hand, non-separable problems have at least two decision variables that are dependent on one another. The difficulty in solving non-separable problems increases in the number of dependencies and whether these dependencies are linear or non-linear. Variable dependencies can be introduced by rotating the decision space, thereby turning each of the separable problems into non-separable problems [3] [8]. In addition to separability, the benchmark functions differ in modality: Both unimodal and multimodal functions are included. Problem dimensionality varied between 30, 500, and 1000 dimensions in order to test performance for both high and low dimensional problems.

Table 1: Test Functions

Test Function, f(x)	Domain	Class
Absolute Value, f_1	[-100 .. 100]	Unimodal Separable
Ackley, f_2	[-30 .. 30]	Multimodal Non-separable Rotated
Egg Holder, f_3	[-512 .. 512]	Multimodal Non-separable
Elliptic, f_4	[-100 .. 100]	Unimodal Separable Rotated
Generalized Griewank, f_5	[-600 .. 600]	Multimodal Non-separable Rotated
Hyperellipsoid, f_6	[-5.12 .. 5.12]	Unimodal Separable
Michalewicz, f_7	[0 .. π]	Multimodal Separable
Norwegian, f_8	[-1.1 .. 1.1]	Multimodal Non-separable
Quadric, f_9	[-100 .. 100]	Unimodal Non-separable
Quartic, f_{10}	[-1.28 .. 1.28]	Unimodal Separable
Generalized Rastrigin, f_{11}	[-5.12 .. 5.12]	Multimodal Separable Rotated
Generalized Rosenbrock, f_{12}	[-30 .. 30]	Multimodal Non-separable Rotated
Salomon, f_{13}	[-100 .. 100]	Multimodal Non-separable
Schaffer 6, f_{14}	[-100 .. 100]	Multimodal Non-separable
Schwefel 1.2, f_{15}	[-100 .. 100]	Unimodal Non-separable Rotated
Schwefel 2.21, f_{16}	[-100 .. 100]	Unimodal Separable
Spherical, f_{17}	[-5.12 .. 5.12]	Unimodal Separable
Step, f_{18}	[-100 .. 100]	Multimodal Separable
Vincent, f_{19}	[0.25 .. 10]	Multimodal Separable
Weierstrass, f_{20}	[-0.5 .. 0.5]	Multimodal Separable

2.4 Control Parameters

Each algorithm was executed on each benchmark problem for 10000 fitness function evaluations. The swarm parameters of the algorithms are listed in table 2.

2.4.1 CPSO-S$_k$ Parameters. The parameters in table 2 used by CPSO-S$_k$ were chosen according to Van den Bergh and Engelbrecht

[1], which shows that these parameters lead to convergence and performed empirically well. $\mathbf{r}_1$ and $\mathbf{r}_2$ were kept random in order to maintain diversity in the population [7].

2.4.2 CPSO-H$_k$ Parameters. Both the CPSO-S$_k$ part and the PSO part of the hybrid CPSO-H$_k$ algorithm utilize the same parameters as given in table 2, aside from the k-value for PSO, which only utilizes a single swarm of 10 particles. Information exchange is limited to only half the particles of each algorithm, as recommended by Van den Bergh and Engelbrecht [10]. This exchange is performed at the end of each iteration for both halves of the algorithm.

2.4.3 CCPSO Parameters. The random variant shares the same parameters as mentioned in table 2, although the decision variables that each sub-swarm optimizes are randomly regrouped amongst the sub-swarms. At the end of every iteration of the CPSO-S$_k$ part of CCPSO, a new k-dimensional vanilla PSO algorithm is initialized and ran for three iterations which shares the ω, c_1, c_2, and number of particles given in table 2.

2.4.4 DCPSO Parameters. Due to the fact that each swarm eventually splits into additional sub-swarms, the k-value of DCPSO naturally changes, as does the total number of particles in the problem space, because each sub-swarm will have 10 particles. The values used for ω, c_1, and c_2 can be found in table 2. The number of splits that occur in each sub-swarm, n_r, is set to two. The DCPSO is initialized as an n-dimensional PSO algorithm, and restructures evenly over the specified maximum number of function evaluations, until DCPSO eventually becomes a CPSO-S.

2.4.5 MCPSO Parameters. The MCPSO parameters are listed in table 2 and follow closely to those seen in DCPSO, aside from the initialization of the algorithm, which begins as CPSO-S. n_r in the context of merging refers to how many sub-swarms are merged during the restructure iteration, and is also set to two. This algorithm also restructures evenly over the maximum number of function evaluations, ending its last fitness evaluations as a PSO algorithm containing one n-dimensional swarm.

Table 2: Swarm Parameters

Parameter	Value
# of sub-swarms (k)	6
# of particles per sub-swarm	10
ω	0.729844
c_1	1.496180
c_2	1.496180

3 EXPERIMENTAL RESULTS DISCUSSION

This section analyses the performance of the five algorithms on the problems discussed in section 2.3. The sections that follow respectively considers the 30, 500, and 1000 dimensional problems.

3.1 30 Dimensions

Table 3 shows mixed results for 30-dimensional problems. The CPSO-S$_k$ ranked the best for separable unimodal problems. The CCPSO algorithm was found to be the best performing algorithm for rotated unimodal problems. DCPSO ranked first for non-separable unimodal, separable multimodal, and rotated multimodal problems. MCPSO tied for first alongside DCPSO for separable multimodal problems, along with being the best performing algorithm for non-separable multimodal problems. On average, DCPSO had the highest rank across all problem classes. For these lower dimensional problems, CPSO-S$_k$ ranked as the best algorithm for separable unimodal and multimodal problem classes. Problems with variable dependencies were best optimized by DCPSO.

Table 3: Paired Mann-Whitney U Test for problems with 30 dimensions (S: Separable, NS: Non-separable, R: Rotated).

Algorithm	Result	Problem Classes					
		Unimodal			Multimodal		
		S	NS	R	S	NS	R
CPSO-S$_k$	Wins	17	1	2	5	8	2
	Losses	4	0	4	10	10	4
	Difference	13	1	-2	-5	-2	-2
	Rank	1	2	3	2	3	4
	S Rank	2					
	NS Rank	3					
CPSO-H$_k$	Wins	14	0	0	1	2	3
	Losses	8	1	3	15	20	4
	Difference	6	-1	-3	-14	-18	-1
	Rank	2	4	4	3	5	3
	S Rank	2.5					
	NS Rank	4					
CCPSO	Wins	5	0	7	5	6	9
	Losses	18	4	0	10	18	7
	Difference	-13	-4	7	-5	-12	2
	Rank	5	5	1	2	4	2
	S Rank	3.5					
	NS Rank	3					
DCPSO	Wins	8	4	3	14	18	7
	Losses	13	0	2	2	5	2
	Difference	-5	4	1	12	13	5
	Rank	4	1	2	1	2	1
	S Rank	2.5					
	NS Rank	1.5					
MCPSO	Wins	11	2	1	14	20	2
	Losses	11	2	4	2	3	6
	Difference	0	0	-3	12	17	-4
	Rank	3	3	4	1	1	5
	S Rank	2					
	NS Rank	3.25					

3.2 500 Dimensions

Results for the larger 500-dimensional problems listed in table 4 begin to show better performance for algorithms which attempt to address variable dependency. The random approach by CCPSO was the best ranking algorithm for non-separable and rotated unimodal functions, while DCPSO performed the best for rotated multimodal problems. MCPSO maintained the highest win-loss ratio for separable unimodal, separable multimodal, and non-separable multimodal problems. For all 500-dimensional problem classes, DCPSO and MCPSO tied as the best performing algorithms in regards to their average rank across all tested problem types. MCPSO ranked the best for both of the separable problem classes. When tackling problems with variable dependency, it was noted that CCPSO and DCPSO tied in their average rank across these non-separable problems.

Table 4: Paired Mann-Whitney U Test for unrotated problems with 500 dimensions (S: Separable, NS: Non-separable, R: Rotated).

Algorithm	Result	Problem Classes					
		Unimodal			Multimodal		
		S	NS	R	S	NS	R
CPSO-S$_k$	Wins	5	2	2	7	7	3
	Losses	19	6	6	13	20	10
	Difference	-14	-4	-4	-6	-13	-7
	Rank	4	4	3	3	4	4
	S Rank	3.5					
	NS Rank	3.75					
CPSO-H$_k$	Wins	1	0	0	2	1	0
	Losses	21	8	8	18	26	15
	Difference	-20	-8	-8	-16	-25	-15
	Rank	5	5	4	5	5	5
	S	5					
	NS Rank	4.75					
CCPSO	Wins	14	8	8	6	14	12
	Losses	10	0	0	14	14	4
	Difference	4	8	8	-8	0	8
	Rank	3	1	1	4	3	2
	S Rank	3.5					
	NS Rank	1.75					
DCPSO	Wins	17	6	5	16	22	13
	Losses	5	2	3	2	5	2
	Difference	12	4	2	14	17	11
	Rank	2	2	2	2	2	1
	S Rank	2					
	NS Rank	1.75					
MCPSO	Wins	20	4	5	17	24	8
	Losses	2	4	3	1	3	5
	Difference	18	0	2	16	21	3
	Rank	1	3	2	1	1	3
	S Rank	1					
	NS Rank	2.25					

3.3 1000 Dimensions

After further increasing the number of dimensions to 1000, the performance of each algorithm noted in table 5 seem to follow a similar trend as seen with 500 dimensions. CCPSO remained the best performing algorithm for non-separable and rotated unimodal problems. DCPSO ranked best for all separable, non-separable and rotated multimodal problems. MCPSO performed the best for separable unimodal problems, however tied with DCPSO for best performing algorithm of the separable multimodal problems. When averaging the ranks over all problem classes, it was ultimately DCPSO that ranked the highest. DCPSO managed to pull ahead of CCPSO as the best performing algorithm across all problems with variable dependencies.

Table 5: Paired Mann-Whitney U Test for unrotated problems with 1000 dimensions (S: Separable, NS: Non-separable, R: Rotated).

Algorithm	Result	Problem Classes					
		Unimodal			Multimodal		
		S	NS	R	S	NS	R
CPSO-S$_k$	Wins	5	1	2	6	7	3
	Losses	19	7	6	14	23	12
	Difference	-14	-6	-4	-8	-16	-9
	Rank	4	5	3	2	4	4
	S Rank	3					
	NS Rank	4					
CPSO-H$_k$	Wins	2	2	0	3	2	0
	Losses	21	5	8	17	26	15
	Difference	-19	-3	-8	-14	-24	-15
	Rank	5	4	4	3	5	5
	S Rank	4					
	NS Rank	4.5					
CCPSO	Wins	14	8	8	6	14	12
	Losses	10	0	0	14	14	4
	Difference	4	8	8	-8	0	8
	Rank	3	1	1	2	3	2
	S Rank	2.5					
	NS Rank	1.75					
DCPSO	Wins	17	5	5	17	23	13
	Losses	5	2	3	2	3	2
	Difference	12	3	2	15	20	11
	Rank	2	2	2	1	1	1
	S Rank	1.5					
	NS Rank	1.5					
MCPSO	Wins	20	3	5	17	22	10
	Losses	3	5	3	2	4	5
	Difference	17	-2	2	15	18	5
	Rank	1	3	2	1	2	3
	S Rank	1					
	NS Rank	2.5					

3.4 Summary of Results

Overall, each of the problem classes tested had a specific best performing algorithm when considering all 30, 500, and 1000 dimensional problems. For the separable unimodal problem class, MCPSO had the best average rank. Non-separable unimodal problems were best optimized by the DCPSO algorithm. CCPSO managed to rank first for rotated unimodal problems across all dimensions. Both the separable and non-separable problem classes were best optimized by MCPSO. The MCPSO algorithm additionally proved to be the best performing algorithm for rotated multimodal problems, which similarly to CCPSO, was not outranked across any of the dimensions tested.

4 CONCLUSION

Due to the limitations of traditional particle swarm optimization (PSO) and cooperative PSO (CPSO) variants when addressing variable dependencies for high dimensional problems, this paper tested the performance of decomposition CPSO (DCPSO) and merging CPSO (MCPSO) against the previously established CPSO algorithms on a variety of problem classes. The comparative algorithms included the generalized CPSO-S_k algorithm which divides n-dimensional problems into k sub-swarms, the hybrid CPSO-H_k algorithm which runs as both a PSO and CPSO-S_k, and the random approach by CCPSO which works to randomly permutate decision variables amongst the sub-swarms. Results showed that both the DCPSO and MCPSO were able to not only outperform both CPSO-S_k and CPSO-H_k for large-scale optimization problems (LSOPs), but also were able to outrank the random CCPSO variant for certain problem classes. Trade-offs between DCPSO and MCPSO were noted, specifically in regards to higher dimensional problems, where DCPSO ranked the best for non-separable problems with variable dependencies, while MCPSO remained the top performer for separable problems. It is hypothesized that the initial optimization as a CPSO-S algorithm seen in MCPSO was able to perform well due to the lack of variable dependencies for the separable problems. Since DCPSO initially optimizes as an n-dimensional PSO algorithm, it was able to group related variables found in non-separable problems at the start of its iterations, resulting in superior performance overall. This suggests that the number of decision variables in each sub-swarm at the start of iterations is an important factor when considering the problem class being optimized.

For future work, experiments detailing the performance of the decomposition and merging variants should include additional performance measures, allowing for a better understanding of how exactly DCPSO and MCPSO traverse the search space. Additional cooperative algorithms should also be introduced to further establish the best performing algorithm for LSOPs. Due to the success in scalability of the proposed algorithms, the dimensionality of problems may further be increased to even higher dimensions and degrees of variable dependency.

5 ACKNOWLEDGEMENTS

This work is based on the research supported by the National Research Foundation (NRF) of South Africa (Grant Number 46712). The opinions, findings and conclusions or recommendations expressed in this article is that of the author(s) alone, and not that of the NRF. The NRF accepts no liability whatsoever in this regard.

REFERENCES

[1] F Bergh and Andries P Engelbrecht. 2001. Effects of swarm size on cooperative particle swarm optimisers. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 892–899.

[2] Jay Douglas, Andries P Engelbrecht, and Beatrice Ombuki-Berman. 2018. Merging and Decomposition Variants of Cooperative Particle Swarm Optimization: New Algorithms for Large Scale Optimization Problems. *UNDER REVIEW* (2018).

[3] Antony W Iorio and Xiaodong Li. 2006. Rotated Test Problems for Assessing the Performance of Multi-objective Optimization Algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. ACM, Seattle, WA, 683–690.

[4] Xiaodong Li and Xin Yao. 2009. Tackling High Dimensional Nonseparable Optimization Problems by Cooperatively Coevolving Particle Swarms. In *IEEE Congress on Evolutionary Computation, 2009. CEC 2009.* IEEE, Trondheim, Norway, 1546–1553.

[5] Sedigheh Mahdavi, Mohammad Ebrahim Shiri, and Shahryar Rahnamayan. 2015. Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences* 295 (2015), 407–428.

[6] Mitchell A Potter and Kenneth A De Jong. 1994. A Cooperative Coevolutionary Approach to Function Optimization. In *Parallel Problem Solving from Nature – PPSN III*. Springer, 249–257.

[7] K Premalatha and AM Natarajan. 2009. Hybrid PSO and GA for Global Maximization. *Int. J. Open Problems Compt. Math* 2, 4 (2009), 597–608.

[8] Ralf Salomon. 1996. Re-evaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions. A Survey of Some Theoretical and Practical Aspects of Genetic Algorithms. *BioSystems* 39, 3 (1996), 263–278.

[9] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[10] Frans Van den Bergh and Andries P Engelbrecht. 2004. A Cooperative Approach to Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation* 8, 3 (2004), 225–239.

[11] Zhi-Qiang Zeng, Hong-Bin Yu, Hua-Rong Xu, Yan-Qi Xie, and Ji Gao. 2008. Fast training Support Vector Machines using parallel sequential minimal optimization. In *Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on*, Vol. 1. IEEE, 997–1001.