Particle Swarm Optimization-Based Variables Decomposition Method for Global Optimization

Khelil Kassoul[1], Samir Brahim Belhaouari [2] and Naoufel Cheikhrouhou[1,3]

[1] Geneva School of Business Administration, University of Applied Sciences Western Switzerland, HES-SO, 1227 Geneva, Switzerland
khelil.kassoul@hesge.ch
[2] Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Qatar
sbelhaouari@hbku.edu.qa
[3] IFM Business School, 1205 Geneva, Switzerland
naoufel.cheikhrouhou@hesge.ch

Abstract. The Particle Swarm Optimization (PSO) algorithm is a well-known nature-inspired technique used to tackle complex optimization problems, widely used by researchers and practitioners due to its simplicity and effectiveness. This paper introduces an improved version of PSO, called Particle Swarm Optimization-based Variables Decomposition Method (PSO-VDM), which utilizes a decomposition technique and a semi-random initialization strategy to divide the problem into subproblems, enhancing exploration and exploitation of the search space. To evaluate the proposed algorithm, a comparison with seven other well-known algorithms is conducted across 13 benchmark problems. The search performance of the algorithms is analyzed using both the test of Wilcoxon signed-rank and Friedman rank. The results of the comparisons and statistical analyses demonstrate that the strategies employed in the PSO-VDM algorithm make a significant contribution to the search process. These comparisons indicate that the PSO-VDM algorithm outperforms other state-of-the-art optimization algorithms in terms of solution quality, highlighting its potential to effectively tackle challenging optimization problems.

Keywords: Particle swarm optimization, single optimization, decomposition method.

1 Introduction

Particle Swarm Optimization (PSO) is a popular nature-inspired optimization algorithm that is used to solve complex optimization problems. It is based on the behavior of a group of particles that move and interact with each other in a multidimensional search space to find the optimal solution. The particles move in the search space based on their position and velocity, which are updated at each iteration of the algorithm. The particles are influenced by their own best position, as well as the best position of the group, which is determined by the particle with the highest fitness value [1].

Due to its simplicity and versatility in tackling different types of optimization problems including multi-objective optimization [2], constraints optimization [3], and global optimization [4], PSO has gained popularity as an optimization method and it has been used in several fields, such as engineering design [5], financial forecasting [6], and image processing [7]. However, it should be noted that its convergence is not always guaranteed [1]. PSO is a prevalent nature-inspired optimization method, alongside other popular techniques like Genetic Algorithms (GA) [8], Ant Colony Optimization (ACO) [9], Firefly Algorithm (FA) [10], and Bison Algorithm [11].

This paper aims to enhance the optimization capabilities of Particle Swarm Optimization (PSO) by proposing a new variant, referred to as Particle Swarm Optimization-based Variables Decomposition Method (PSO-VDM), which focuses on solving single-objective optimization problems. The PSO-VDM method incorporates a decomposition technique and a semi-random initialization of variables to facilitate more effective exploration and exploitation of the search space. By dividing the problem into smaller subproblems, the method is able to efficiently explore the search space, thereby improving the global optimization performance of PSO.

The paper is organized as follows: Section 2 presents an overview of PSO. Section 3 introduces the proposed PSO-VDM algorithm, Section 4 provides the experimental results obtained from testing the proposed algorithm, and finally, Section 5 concludes the paper and suggests possible directions for future research.

2 Particle swarm optimization

Inspired by the collective behavior of birds or fishes, PSO is a heuristic global optimization algorithm [12]. PSO employs a swarm of particles, with each particle representing a potential solution within the search space. By simulating the social behavior of these particles, PSO is able to effectively explore and optimize the solution space in search of the best possible solution. The particles undergo two types of learning, cognitive learning and social learning, to improve their performance. In cognitive learning, they rely on their own experience and remember their best solution visited so far. This is referred to as their personal best position. In social learning, they learn from other particles in the swarm and remember the best solution found by any particle. This is called the global best position. This dual learning approach enables the particles to continuously optimize their performance in the swarm [1].

PSO is an iterative algorithm that begins with a swarm of randomly initialized solutions and searches for optima by updating iterations. Each particle in the swarm has several parameters, including its velocity, its personal best position, and its current position [13]. Each particle's position ($x_i(t)$) and velocity ($v_i(t)$) in a d-dimensional search space are denoted by d-dimensional vectors. These vectors serve as guides for the particle's movement based on both individual and collective knowledge. By leveraging these vectors, the particles can navigate the search space and optimize their performance as part of the swarm. In the original version of PSO, the position and velocity of each particle are updated at each iteration *t* using the following equations:

$$v_i(t+1) = v_i(t) + r_1 c_1\big(p_i(t) - x_i(t)\big) + r_2 c_2\left(p_g(t) - x_i(t)\right) \quad (1)$$

$$x_i(t+1) = v_i(t+1) + x_i(t) \quad (2)$$

Where, at iteration *t*, $v_i(t)$ denotes the velocity of the i^{th} particle and is updated using information from two sources: the particle's personal best position, denoted by $p_i(t)$, and the global best position of the swarm, denoted by $p_g(t)$. The update also involves two random values, r_1 and r_2, which follow a uniform distribution in the range of [0,1]. Additionally, the algorithm uses two acceleration constants, known as cognitive and social parameters, denoted by c_1 and c_2, respectively. The original version of PSO typically employs fixed values of 2.0 for these parameters.

Shi and Eberhart [14] propose an improvement to the original PSO algorithm by introducing the concept of an inertia weight denoted as (w). The purpose of this weight is to balance the exploration of the entire search space with the exploitation of promising regions. Empirical results demonstrate that when the value of "w" falls within the range of [0.8, 1.2], particles tend to search for global solutions, while values that are too small cause particles to converge too quickly towards local optima.

Clerc [15] proposes a way to promote the convergence of the PSO algorithm by introducing a parameter called the constriction factor (χ). By introducing this factor, the impact of a particle's velocity is reduced as the search progresses, which maintains a balance between exploitation and exploration. This approach enhances the search process by allowing the algorithm to focus on the current solution space region while still exploring diverse regions of the solution space.

PSO has undergone various modifications and improvements since its introduction, such as the addition of an inertia weight factor [14] as mentioned above and the proposition of new strategies [16, 17]. These changes are made because the original PSO algorithm had some limitations, including premature convergence and its restriction to real optimization problems. In response, researchers have introduced and modified different parameters and aspects of the algorithm. Furthermore, the combination of PSO with other optimization algorithms has been proposed, leading to the development of many different variants of PSO that can be applied to solve various types of problems across different fields. The various modifications and applications of PSO demonstrate its versatility and potential to solve complex optimization problems.

3 Methodology

In this section, we present the developed method that aims to achieve two main objectives. The first objective is to simplify the problem by decomposing it into smaller subproblems. To accomplish this, we divide the problem into subproblems of *n* variables and identify their optimal/ near optimal values within the swarm population. This process is repeated until the full dimension of the problem is reached. The second objective is to prevent premature convergence, which is accomplished by using a jumping technique, as proposed in [18], based on an exponential function.

3.1 Decomposition method

To simplify the problem, the method first identifies the optimal/near optimal values of *n* variables $(x_1, x_2, \dots, x_n)$ among the entire size of the problem. Then, the algorithm evaluates *n* additional variables $(x_{n+1}, x_{n+2}, \dots, x_m)$, where $m = 2n$, to determine their optimal/near optimal values. This process reduces the problem to one with *m* variables $(x_1, x_2, \dots, x_m)$ as illustrated in Figure 1, and the same procedure is repeated until the full dimension of the problem is reached. Furthermore, unlike a complete random initialization of variables, our approach involves a partially random initialization technique during the update process. The variables are grouped into subproblems, allowing for better coverage of the entire search space and ultimately enhancing the effectiveness of the optimization process.

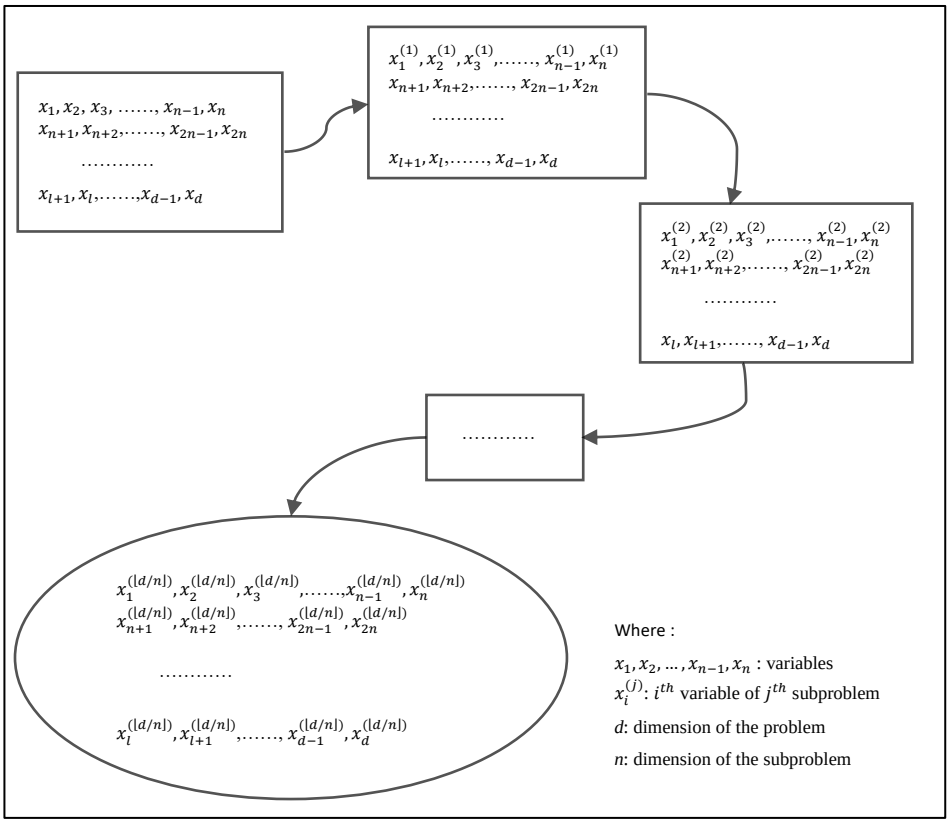

Fig. 1. Process of the proposed method PSO-VDM

3.2 Jumping strategy

To prevent particles from being stuck in local minima traps, the jumping strategy is incorporated into the velocity equation. This strategy utilizes an exponential function denoted as $\xi(t)$. Equation (3) depict the formula of the exponential function which is

defined at every iteration *t*. When a particle's velocity is close to zero, it implies that the particle is stuck in a local optimum. To avoid such scenarios, the particle can make a leap using $\xi(t)$ function. Each particle's velocity is updated using equation (4), where $c(t)$ is the exponential coefficient, which depends on the iteration *t*, and is different from the approach used in [18]. Additionally, the equation includes other parameters such as *w*, the inertia weight, V_r, a random vector, c_1, and c_2, all of which are defined at the start of the run.

$$\xi(t) = e^{1/(\|p_i(t)-x_i(t)\|+\epsilon)}, \ \epsilon \text{ is a small positive value.} \tag{3}$$

$$v_i(t+1) = w\, v_i(t) + c(t)\xi\, V_r + r_1 c_1 (p_i(t) - x_i(t)) + r_2 c_2 \big(p_g(t) - x_i(t)\big) \tag{4}$$

3.3 Dynamic parameters

In order to maintain a balance between exploration and exploitation among the swarm population, a linearly decreasing exponential coefficient $c(t)$ is employed. This approach allows for larger jumps at the beginning of the search to facilitate exploration of the entire search space, while smaller jumps are used later on to refine solutions in specific regions. To update the coefficient $c(t)$ at each iteration t, equation (5) is used as follows:

$$c(t+1) = r^t \tag{5}$$

where *r* is a damping coefficient $\in$]0,1].

The proposed algorithm integrates a dynamic inertia weight strategy, where $w(t)$ value linearly decreases with each iteration based on equation (6). This approach reduces the particles' inclination towards continuous global search, while increasing their propensity towards local search.

$$w(t+1) = w(t)^t \tag{6}$$

In summary, the proposed PSO-VDM algorithm achieves efficient local and global search by decomposing the problem into smaller subproblems and preventing premature convergence using a jumping strategy based on an exponential function. The balance between exploitation and exploration is achieved by utilizing a linearly decreasing dynamic exponential coefficient $c(t)$. To balance the inclination of particles towards local and global search, PSO-VDM also integrates a dynamic inertia weight. The pseudo-code of the PSO-VDM algorithm is presented in Algorithm 1.

Algorithm 1. Steps of the PSO-VDM algorithm

```
Initialization
1: Set the initial parameters
2: Assign random initial values to the particles' positions and velocities
3: Assign the first best position values to the particle's current position
4: Evaluate the fitness values of the particles to determine the global best position

Update of particle position and velocity
5: while the full dimension of the problem is not reached do
6: Consider only n variables of the problem.
7: Apply equations (4) and (2) to update the particle's velocity and position, respectively
8: Apply position and velocity limits

Update of best global and personal positions
9: Update the global and best positions
10: Apply equation (5) to update the exponential coefficient
11: Apply equation (6) to update the inertia weight coefficient
12: Consider n additional variables of the problem
13: end while

Convergence procedure
14: Iterate through steps 5 to 13 until the maximum number of iterations is reached
```

4 Experimental data and analysis

4.1 Experimental settings

In this section, we evaluate the performance of the PSO-VDM algorithm by evaluating its effectiveness on 13 classical functions commonly used by researchers. Furthermore, we compare our results with those of other benchmark algorithms to validate the efficiency of our approach. Tables 1-2 list these benchmark functions and provide information such as the dimension of the function (*D*), the boundary of the function's search space (Column 2), and the optimum value f_{min}. Despite their simplicity, these benchmark functions are valuable for evaluating the effectiveness of the PSO-VDM algorithm. The scalable problems are divided into two categories: unimodal functions ($f_1 - f_7$)(Table1) and multimodal functions ($f_8 - f_{13}$)(Table 2). The algorithm's exploitation search and convergence are well-suited for unimodal functions that have only

one global optimum. On the other hand, multimodal functions, with multiple local optima, are useful for the exploration search.

Table 1. Description of unimodal functions.

D	Search space	f_{min}	f	Name
30	$[-100,100]$	0	$f_1(x) = \sum_{i=1}^{d} x_i^2$	Sphere Function
	$[-10,10]$	0	$f_2(x) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	Schwefel's Problem 2.22
	$[-100,100]$	0	$f_3(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j\right)^2$	Schwefel's Problem 1.2
	$[-100,100]$	0	$f_4(x) = max\lvert x_i \rvert, 0 \le i \le d$	Schwefel's Problem 2.21
	$[-30,30]$	0	$f_5(x) = \sum_{i=1}^{d} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	Rosenbrock Function
	$[-100,100]$	0	$f_6(x) = \sum_{i=1}^{d} (\lfloor x_i + 0.5 \rfloor)^2$	Step Function
	$[-1.28,1.28]$	0	$f_7(x) = \sum_{i=1}^{d} i x_i^4 + rand[0.1)$	Noise Function 'Quartic'

Table 2. Description of multimodal functions.

D	Search space	f_{min}	f	Name
30	$[-500,500]$	−1.25e+04	$f_8(x) = 418.9829 \cdot d - \sum_{i=1}^{d} x_i sin\left(\sqrt{x_i}\right)$	Schwefel's Function
	$[-5.12,5.12]$	0	$f_9(x) = \sum_{i=1}^{d} (x_i^2 - 10cos(2\pi x_i) + 10)$	Rastrigin's Function
	$[-32,32]$	0	$f_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} cos2\pi x_i\right) + 20 + e$	Ackley's Function
	$[-600,600]$	0	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Griewank Function
	$[-50,50]$	0	$f_{12}(x) = \frac{\pi}{d}\{10sin^2(\pi y_i) + \sum_{i=1}^{d-1}(y_i - 1)^2 [1 + 10sin^2(\pi y_{i+1})] + (y_d - 1)^2\} + \sum_{i=1}^{d} u(x_i, 10,100,4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < a \end{cases}$	Generalized Penalized Function 1
	$[-50,50]$	0	$f_{13=}0.1\{sin^2(3\pi x_i) + \sum_{i=1}^{d}(x_i - 1)^2 [1 + sin^2(3\pi x_i + 1)] + (x_d - 1)^2[1 + sin^2(2\pi x_i)]\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$	Generalized Penalized Function 2

4.2 Benchmark problems

The performance of PSO-VDM is evaluated against seven state-of-the-art algorithms: Chaotic Grey Wolf Optimizer (Chaotic GWO) [19], Adaptive Differential Evolution (JADE) [20], Improved Moth-Flame Optimization (IMFO) algorithm [21], Whale Optimization Algorithm (WOA) [22], Improved Equilibrium Optimization Algorithm (IEOA) [23], Cellular Genetic Algorithm (cGA) [24], and Clerc's PSO (with parameters $w = \chi \approx 0.7298, c_1 = 2\ and\ c_2 = 2$). In all experiments conducted, the termination criterion is based on a maximum number of function evaluations known as

Max_FES. To ensure a fair comparison, we set the population size *N* and Max_FES to 30 and 30000, respectively. In this paper, the proposed algorithm's parameter settings are presented in Table 3, which includes the parameters utilized in updating the velocity equation, such as the exponential coefficient, cognitive and social parameters, inertia weight, and damping coefficient. Additionally, the subproblem's number of variables, denoted as *n*, is set to 5. Tables 4-5 present the comparison results for each experiment, including the standard deviation (S.D.) and the average value (Avg.). The best-performing algorithm's results are highlighted in bold for easy identification.

Table 3. Parameter settings of the proposed algorithm.

Parameter Settings	
$c = 2$	Exponential coefficient
$c_1 = 2$	Cognitive parameter
$c_2 = 2$	Social parameter
$w = 0.9$	Inertia weight
$r = 0.05$	Damping coefficient
$n = 5$	Number of variables of a subproblem

Table 4. Comparison results for unimodal functions.

f	Results	PSO-VDM	Chaotic GWO	IEOA	JADE	IMFO	WOA	cGA	PSO
f_1	Avg.	**0.00e+00**	6.83e−32	2.78e−21	1.8e–60	5.14e−53	1.41e−30	1.22e−01	6.29e+02
	S.D.	0.00e+00	3.70e−31	0.00e+00	8.40e–60	2.80e−52	4.91e−30	3.80e−02	3.45e+02
f_2	Avg.	**0.00e+00**	1.83e−19	1.95e−26	1.80e–25	1.35e−27	1.06e−21	8.33e−02	9.31e+01
	S.D.	0.00e+00	1.00e−18	2.03e−26	8.80e–25	6.78e−27	2.39e−21	1.60e−02	3.84e+01
f_3	Avg.	**0.00e+00**	5.79e−08	2.52e−13	5.70e–61	2.35e−14	5.39e−07	2.33e+03	2.22e+03
	S.D.	0.00e+00	1.60e−07	7.04e−13	2.70e–60	4.04e−14	2.93e−06	7.80e+02	1.17e+03
f_4	Avg.	**0.00e+00**	4.67e−08	1.03e−13	8.20e–24	1.59e−14	7.26e−02	2.86e+00	1.50e+01
	S.D.	0.00e+00	2.40e−07	1.87e−13	4.00e–23	2.75e−14	3.98e−01	5.40e−01	3.75e+00
f_5	Avg.	**8.77e−06**	2.14e+01	2.54e+01	8.00e–02	7.98e−01	2.79e+01	2.01e+02	5.04e+04
	S.D.	3.50e−05	7.20e−01	2.18e-01	5.60e–01	1.62e+00	7.64e−01	4.40e+02	5.24e+04
f_6	Avg.	**0.00e+00**	**0.00e+00**	1.62e−05	2.90e+00	**0.00e+00**	3.12e+00	**0.00e+00**	9.46e+02
	S.D.	0.00e+00	0.00e+00	2.00e−05	1.20e+00	0.00e+00	5.32e−01	0.00e+00	5.00e+02
f_7	Avg.	**0.00e+00**	**0.00e+00**	7.71e-04	6.40e–04	4.35e−03	1.43e−03	2.91e−02	2.64e−02
	S.D.	0.00e+00	0.00e+00	5.05e−04	2.50e–04	2.81e−03	1.15e−03	8.90e−03	2.56e−02

According to Tables 4-5, the results demonstrate that PSO-VDM consistently outperforms the other algorithms on unimodal functions $f_1 - f_7$, showcasing its superior exploitation potential compared to the other methods. Furthermore, for multimodal functions $f_8 - f_{13}$, PSO-VDM achieves the best results on $f_9 - f_{11}$, secures the second-best result on f_8, and yields slightly inferior results on $f_{12} - f_{13}$ compared to the top-performing algorithm (IMFO) and (JADE). These results indicate that PSO-VDM has a strong ability to explore the search space thoroughly and effectively.

Table 5. Comparison results for unimodal functions.

f	Results	PSO-VDM	Chaotic GWO	IEOA	JADE	IMFO	WOA	cGA	PSO
f_8	Avg.	−1.24e+04	−7.01e+03	−1.07e+04	**−1.25e+04**	−3.62e+03	−5.08e+03	−1.21e+04	−6.28e+03
	S.D.	8.81e+02	1.50e+03	4.80e+02	2.30e−05	2.86e+02	6.96e+02	1.90e+02	6.53e+02
f_9	Avg.	**0.00e+00**	1.02e+01	**0.00e+00**	1.00e−04	4.08e+00	**0.00e+00**	8.21e−02	6.48e+01
	S.D.	0.00e+00	9.80e+00	0.00e+00	6.00e−05	1.72e+00	0.00e+00	3.90e−02	2.55e+01
f_{10}	Avg.	**8.88e−16**	1.30e−14	7.16e−15	8.20e−10	4.56e−15	7.40e+00	8.91e−02	7.74e+00
	S.D.	0.00e+00	7.50e−15	1.53e−15	6.90e−10	6.49e−16	9.90e+00	1.80e−02	1.34e+00
f_{11}	Avg.	**0.00e+00**	3.67e−03	**0.00e+00**	9.90e−08	8.01e−02	2.89e−04	1.89e−01	6.35e+00
	S.D.	0.00e+00	6.00e−03	0.00e+00	6.00e−07	3.56e−02	1.59e−03	5.10e−02	3.28e+00
f_{12}	Avg.	1.39e−09	2.53e−03	5.71e−07	4.60e−17	**4.72e−32**	3.40e−01	5.57e−04	1.12e+01
	S.D.	4.67e−09	2.70e−03	6.15e−07	1.90e−16	2.46e−34	2.15e−01	4.20e−04	6.24e+00
f_{13}	Avg.	4.21e−12	5.99e−03	1.31e−02	2.00e−16	**1.35e−32**	1.89e+00	9.05e−03	9.00e+03
	S.D.	4.86e−12	2.80e−03	3.13e−02	6.50e−16	5.57e−48	2.66e−01	5.20e−03	3.43e+04

4.3 Algorithm complexity

In this section, we provide an analysis of the complexity of our PSO-VDM algorithm, as evaluated in the CEC 2020 competition for dimensions D = {5, 10, 15} [25]. The experiments are conducted using Matlab R2020a on a PC with an 8 GB RAM and i5CPU processor with a 1.9 GHz Core (TM). As outlined in [25], the algorithm complexity is determined by evaluating T_0, which is the time taken to execute the following test problem:

$z = 0.55$
for $i = 1{:}\,1000000$
$z = z + z;\;\; z = z/2;\;\; z = z * z;\;\; z = sqrt(z);\;\; z = \log(z);$
$z = \exp(z);\;\; z = z/(z + 2);$
end

T_1 represents the time required for computing f_1 over 200,000 Function Evaluations (FES), while T_2 represents the time taken to execute PSO-VDM for 200,000 FES of f_1 in D dimensions, and $\widehat{T_2}$ represents the mean of T_2 calculated from five independent runs. The complexity of algorithm is calculated by the expression $\frac{\widehat{T_2}-T_1}{T_0}$. Table 6 summarizes the results obtained, which show that the computational time increases linearly with an increase in the problem's dimension.

Table 6. Algorithm Complexity.

D	T_0	T_1	$\widehat{T_2}$	$\frac{\widehat{T_2}-T_1}{T_0}$
5	0.0394	1,9425	10,4330	215,49
10		2,0828	10,7346	219,58

15		2,8293	11,8506	228,96

4.4 Statistical analysis

Table 6 summarizes the results of the validation studies presented in Tables 4-5, which include all pairwise comparisons involving PSO-VDM and its competitive algorithms. The table uses symbols (+/-/=) to indicate whether PSO-VDM performs better than, worse than, or similar to the compared algorithm. The bolded numbers indicate the functions for which PSO-VDM outperforms the corresponding algorithm.

Table 7 clearly shows that PSO-VDM outperforms the other competitive algorithms in terms of convergence across the two diverse problem categories. It is worth noting, however, that JADE and IFMO exhibit better performance than PSO-VDM on three out of six functions and two out of six functions, respectively, in the multimodal benchmark problems. Nevertheless, our method remains unbeaten in terms of unimodal functions.

Table 7. A summary of the statistical results.

Problem category	PSO-VDM	Chaotic GWO	IEOA	JADE	IMFO	WOA	cGA	PSO
Unimodal	+/−/=	**5**/0/2	**7**/0/0	**7**/0/0	**6**/0/1	**7**/0/0	**6**/0/1	**7**/0/0
Multimodal	+/−/=	**6**/0/0	**4**/0/2	3/3/0	**4**/2/0	**5**/0/1	**6**/0/0	**6**/0/0

To evaluate the validity of the results and compare the performance of PSO-VDM against other competitive algorithms, the test of Wilcoxon signed is employed with a significance level of 5%. Table 8 displays the z-value and p-value for the comparison of PSO-VDM with other optimizers. A p-value greater than 0.05 indicates that there is no significant difference in the results between the two algorithms. PSO-VDM outperforms the other algorithms significantly, as indicated by the p-values of only two comparisons (PSO-VDM vs. JADE and PSO-VDM vs. IFMO) being greater than 0.05 (marked in bold in Table 8).

Among all the approaches and for all types of functions, PSO-VDM is ranked first in the Friedman mean rank. The test of Friedman is employed with a 5% significance level, and the p-value obtained (i.e., 2.1268e−9 as shown in Table 8) indicates that the results of PSO-VDM are significantly better than those of other algorithms. Thus, PSO-VDM is an effective optimization technique with a strong ability to balance the exploration and exploitation of the search space.

Table 8. A summary of the statistical results.

Method	**Rank**	**Mean rank of Friedman**	**z-value** (PSO-VDM versus)	**p-value** (PSO-VDM versus)
PSO-VDM	**1**	1,77	-	-
Chaotic GWO	5	4,50	-2,588733	0,009633
IEOA	4	3,96	-2,934058	0,003346
JADE	2	2,92	-1,432656	**0,151956**
IFMO	3	3,54	-1,712199	**0,086860**
WOA	6	5,69	-3,059412	0,002218
cGA	7	5,92	-3,109912	0,001871
PSO	8	7,69	-3,179797	0,001474
p-value		2,1268e−9	**Wilcoxon signed test**	

5 Conclusion

In this paper, a new optimization algorithm called Particle Swarm Optimization-based-Variables Decomposition Method (PSO-VDM) is introduced. The objective of the PSO-VDM algorithm is to improve the overall optimization performance of the PSO by utilizing a decomposition technique to divide the problem into smaller sub-problems with a semi-random initialization of variables. The algorithm's performance is evaluated using 13 test functions, and the results prove that the PSO-VDM algorithm is competitive with other popular heuristics such as GWO, JADE, and GA.

The PSO-VDM algorithm demonstrates superior exploitation capabilities for uni-modal functions and strong exploration abilities for multimodal functions. The statistical analysis confirms that the proposed algorithm has a good balance between exploiting and exploring the search space.

One potential research perspective is to investigate the limitations of PSO-VDM regarding solution quality and computation time. To this end, it would be valuable to assess the effectiveness of the proposed approach when applied to large-scale optimization problems. Another potential research direction involves the hybridization of PSO-VDM with other metaheuristics for enhancing its performance.

References

1. Bansal, J.C., Singh, P.K., Pal, N.R. eds: Evolutionary and Swarm Intelligence Algorithms. Springer International Publishing, Cham (2019)
2. Zhao, W., Wang, H., Geng, J., Hu, W., Zhang, Z., Zhang, G.: Multi-Objective Weather Routing Algorithm for Ships Based on Hybrid Particle Swarm Optimization. J. Ocean Univ. China. 21, 28–38 (2022). https://doi.org/10.1007/s11802-022-4709-8
3. Duan, B., Guo, C., Liu, H.: A hybrid genetic-particle swarm optimization algorithm for multi-constraint optimization problems. Soft Comput. 26, 11695–11711 (2022). https://doi.org/10.1007/s00500-022-07489-8
4. Kassoul, K., Belhaouari, S.B., Cheikhrouhou, N.: Dynamic Cognitive-Social Particle Swarm Optimization. In: 2021 7th International Conference on Automation, Robotics and

Applications (ICARA). pp. 200–205, 04-06 February (2021). doi: 10.1109/ICARA51699.2021.9376550
5. Sahoo, L., Bhunia, A.K., Pal, P., Bala, S.S.: Tournament constriction coefficient based particle swarm optimization (TPSO-Co) for engineering design optimization problems. Int J Syst Assur Eng Manag. (2022). https://doi.org/10.1007/s13198-022-01824-w
6. Bas, E., Egrioglu, E., Kolemen, E.: Training simple recurrent deep artificial neural network for forecasting using particle swarm optimization. Granul. Comput. 7, 411–420 (2022). https://doi.org/10.1007/s41066-021-00274-2
7. Pashaei, E., Pashaei, E.: A fusion approach based on black hole algorithm and particle swarm optimization for image enhancement. Multimed Tools Appl. 82, 297–325 (2023). https://doi.org/10.1007/s11042-022-13275-3
8. Figueroa-García, J.C., Neruda, R., Hernandez–Pérez, G.: A genetic algorithm for multivariate missing data imputation. Information Sciences. 619, 947–967 (2023). https://doi.org/10.1016/j.ins.2022.11.037
9. Kavitha, R., Jothi, D.K., Saravanan, K., Swain, M.P., Gonzáles, J.L.A., Bhardwaj, R.J., Adomako, E.: Ant Colony Optimization-Enabled CNN Deep Learning Technique for Accurate Detection of Cervical Cancer. BioMed Research International. 2023, e1742891 (2023). https://doi.org/10.1155/2023/1742891
10. Cheng, Z., Song, H., Zheng, D., Zhou, M., Sun, K.: Hybrid firefly algorithm with a new mechanism of gender distinguishing for global optimization. Expert Systems with Applications. 120027 (2023). https://doi.org/10.1016/j.eswa.2023.120027
11. Kazikova, A., Pluhacek, M., Viktorin, A., Senkerik, R.: New Running Technique for the Bison Algorithm. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., and Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. pp. 417–426. Springer International Publishing, Cham (2018). doi:10.1007/978-3-319-91253-0_39
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. pp. 1942–1948. IEEE (1995). doi:10.1109/ICNN.1995.488968
13. Taherkhani, M., Safabakhsh, R.: A novel stability-based adaptive inertia weight for particle swarm optimization. Applied Soft Computing. 38, 281–295 (2016). https://doi.org/10.1016/j.asoc.2015.10.004
14. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). pp. 69–73. IEEE, 04-09 May (1998). doi:10.1109/ICEC.1998.699146
15. Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). pp. 1951–1957. IEEE, 6-9 July (1999). doi: 10.1109/CEC.1999.785513
16. Liu, H., Zhang, X.-W., Tu, L.-P.: A modified particle swarm optimization using adaptive strategy. Expert Systems with Applications. 152, 113353 (2020). https://doi.org/10.1016/j.eswa.2020.113353
17. Rehman, A.U., Islam, A., Belhaouari, S.B.: Multi-cluster jumping particle swarm optimization for fast convergence. IEEE Access. 8, 189382-189394, (2020). doi: 10.1109/ACCESS.2020.3031003
18. Kassoul, K., Zufferey, N., Cheikhrouhou, N., Brahim Belhaouari, S.: Exponential Particle Swarm Optimization for Global Optimization. IEEE Access. 10, 78320–78344 (2022). https://doi.org/10.1109/ACCESS.2022.3193396

19. Lu, C., Gao, L., Li, X., Hu, C., Yan, X., Gong, W.: Chaotic-based grey wolf optimizer for numerical and engineering optimization problems. Memetic Computing. 12, 371–398 (2020). https://doi.org/10.1007/s12293-020-00313-6
20. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Transactions on evolutionary computation. 13, 945–958 (2009). doi:10.1109/TEVC.2009.2014613
21. Pelusi, D., Mascella, R., Tallini, L., Nayak, J., Naik, B., Deng, Y.: An Improved Moth-Flame Optimization algorithm with hybrid search phase. Knowledge-Based Systems. 191, 105277 (2020). https://doi.org/10.1016/j.knosys.2019.105277
22. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Advances in engineering software. 95, 51–67 (2016). https://doi.org/10.1016/j.advengsoft.2016.01.008
23. Shaheen, A.M., Elsayed, A.M., El-Sehiemy, R.A., Abdelaziz, A.Y.: Equilibrium optimization algorithm for network reconfiguration and distributed generation allocation in power systems. Applied Soft Computing. 98, 106867 (2021). https://doi.org/10.1016/j.asoc.2020.106867
24. Alba, E., Dorronsoro, B.: A hybrid cellular genetic algorithm for the capacitated vehicle routing problem. Engineering Evolutionary Intelligent Systems. 379–422 (2008). doi:10.1007/978-3-540-75396-4_14
25. C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali, B. Y. Qu, N. H. Awad and P. Biswas, Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization, Technical Report 201911, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, November 2019.