

Hybridization of Metaheuristics with Data Mining for Continuous Optimization: A Cooperative Parallel Approach

Victor José B. A. Martínez

Universidade Federal do Maranhão

Av. dos Portugueses São Luís/MA

victor.martinez@discente.ufma.br

Rick Eick Vieira dos Santos

Universidade Federal do Maranhão

Av. dos Portugueses São Luís/MA

rick.eick@discente.ufma.br

Raphael Gomes Santos

Universidade Federal Fluminense

Rua Miguel de Frias, Niterói/RJ

raphaelgomes@id.uff.br

Alexandre Plastino

Universidade Federal Fluminense

Rua Miguel de Frias, Niterói/RJ

plastino@ic.uff.br

Alexandre C. M. Oliveira

Universidade Federal do Maranhão

Av. dos Portugueses São Luís/MA

alexandre.cesar@ufma.br

ABSTRACT

The hybridization of metaheuristics via data-driven techniques has been successfully applied to combinatorial optimization. However, the use of these hybrid methods in continuous optimization problems is still scarce in the literature. Furthermore, data-driven algorithms may be ineffective when managing large volumes of data in high-dimensional spaces. In this context, this work proposes a hybrid parallel metaheuristic, *CDM-LSHADE*, for continuous optimization resulting from the combination of mining techniques and cooperative parallelism. Computational experiments were performed to validate the algorithm by comparing it with evolutionary and data-driven approaches. It was found that the proposed algorithm presented, on average, better solutions in 59.48% of the cases, while maintaining competitive execution times. Future work will focus on the integration of new clustering approaches, as well as exploring more strategies for sending and inserting data to the mining set.

KEYWORDS. Evolutionary Algorithms, Data Mining, Parallelism.

OC – Continuous Optimization, MH – Metaheuristics

RESUMO

A hibridização de meta-heurísticas via técnicas dirigidas por dados tem sido aplicadas com sucesso para problemas de otimização combinatória. Entretanto, a utilização desses métodos híbridos em problemas de otimização contínua ainda é escassa na literatura. Além disso, algoritmos guiados por dados podem ser ineficazes ao gerenciar grande volume de dados em espaços de alta dimensão. Nesse contexto, o presente trabalho propõe uma meta-heurística híbrida paralela, *CDM-LSHADE*, para otimização contínua resultante da combinação de técnicas de mineração e paralelismo cooperativo. Experimentos Computacionais foram realizados para validação do algoritmo comparando-o com abordagens evolutivas e dirigidas por dados. Verificou-se que o algoritmo proposto apresentou, em média, soluções melhores em 59.48% dos casos, mantendo ainda tempos de execução competitivos. Trabalhos futuros irão focar na integração de novas abordagem de agrupamento e explorar mais estratégia para o envio e inserção de dados ao conjunto de mineração.

PALAVRAS CHAVE. Algoritmos Evolucionários, Mineração de Dados, Paralelismo.

OC – Otimização Contínua, MH – Metaheurísticas

1. Introduction

Optimization is a central area of Operations Research, which employs a wide range of mathematical, statistical, and computational methods to support decision-making in complex systems. Its primary goal is to find solutions that either maximize or minimize specific objectives. The process of optimization plays a crucial role in improving efficiency, reducing costs, and enhancing performance across various domains. In fact, many problems across various scientific and engineering fields can be modeled as numerical optimization problems, such as structural design, financial modeling, and control system configuration [Pardalos and Resende, 2002].

These problems typically involve optimizing an objective function over a continuous or discrete search space. To solve them, there are two main approaches: Exact and Heuristic. Exact methods (or classical methods), such as gradient descent, Newton's method, and convex programming, rely on the analytical properties of the objective function, assuming specific mathematical properties such as continuity, convexity, and differentiability. However, many problems encountered in real-world applications are non-convex or non-differentiable, and may exhibit discontinuities or even noise, limiting the applicability of exact methods [Nocedal and Wright, 2006].

In this context, the heuristic approach, more flexible and primarily based on evaluating the objective function, offers practical alternatives for exploring complex and high-dimensional search spaces while making minimal assumptions in real-world problems. Metaheuristics, for example, are powerful tools within the heuristic approach to guide the search for optimal solutions. An example of such metaheuristics, mimicking a natural processes, is Evolutionary Algorithms, which are particularly effective in complex, multimodal, and black-box optimization problems, where the internal structure of the objective function remains unknown [Eiben and Smith, 2015].

Among the Evolutionary Algorithms, the Differential Evolution (DE) algorithm stands out [Storn and Price, 1997]. Its operation revolves around evolving a population of candidate solutions through mutation, recombination (crossover), and selection mechanisms, exploiting the differences between individuals in the population. Its performance, ease of implementation, and minimal need for parameter tuning have made DE a standard reference in continuous optimization. Furthermore, DE variants such as LSHADE (Linear Population Size Reduction Success History-based Adaptive Differential Evolution) have achieved state-of-the-art results in benchmark competitions, such as the IEEE Congress on Evolutionary Computation (CEC) [Tanabe and Fukunaga, 2013], where LSHADE and its descendants consistently rank among the top-performing algorithms.

Aiming to further enhance the quality of solutions provided by metaheuristic methods, including DE and its variants, there has been growing research on algorithm hybridization. That is, the incorporation of complementary techniques to improve performance, robustness, and adaptability across diverse problem landscapes. Among these, the integration with Data Mining (DM) techniques has gained attention, aiming at learning from past experiences to enable greater responsiveness in strategy adaptation and pattern detection in the search space.

In this way, data generated during the optimization process — such as good-quality solutions, search trajectories, and convergence history — can be interpreted using DM methods, including clustering, classification, regression, predictive analysis, or association rule mining. This enables the identification of promising search regions and/or the adjustment of control parameters. Such an approach has already demonstrated promising results in both combinatorial and continuous problem contexts. For instance, knowledge extracted through DM has been used to dynamically adapt mutation operators, population sizes, and even restart strategies [Talbi, 2021].

Moreover, Data-Driven Metaheuristics can also leverage parallelism techniques, enabling the simultaneous execution of multiple operations such as solution evaluation, operator application, or subpopulation management. Recent works on parallel metaheuristic algorithms for high-dimensional optimization also focus on overcoming challenges such as stagnation in local optima and slow convergence by introducing cooperative strategies and parallel computing to improve search efficiency and solution quality [Cai et al., 2024].

Although many studies highlight improvements in both solution quality and computational time as advantages of parallelizing metaheuristics [Zhuo et al., 2023; Almeida et al., 2022], enhancing the quality of the final solution is, in some cases, more desirable, as long as the computation time remains viable. As such, the efficiency of the parallelization process and the resulting speedup are not the main concern. Therefore, cooperative strategies can be employed to improve the exploration and exploitation of the search space through information exchange, as shown in the works of Bujok and Kolenovský [2022] and Cai et al. [2024].

In this context, this paper proposes a cooperative multi-agent evolutionary framework based on a hybridized variant of the *LSHADE* algorithm, called *DM-LSHADE*, which employs Data Mining techniques, as further explained in Section 3. This novel cooperative algorithm, called Cooperative *DM-LSHADE* (*CDM-LSHADE*), is capable of mining and exploiting high-quality patterns from elite solutions to guide the search for optimal solutions in continuous optimization problems. The proposal implements cooperative strategies within Data-Driven Differential Evolution, featuring a parallel multi-agent architecture with an information exchange protocol.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature, providing the foundation for understanding this work. Section 3 provides background and presents the *DM-LSHADE*, our base algorithm. Section 4 describes the proposed cooperative variant of *DM-LSHADE*: *CDM-LSHADE*. In Section 5, the results and discussion of computational experiments comparing *CDM-LSHADE* with *DM-LSHADE* and *LSHADE* are presented. Finally, Section 6 summarizes key findings and highlights potential directions for future work.

2. Related Works

In general, metaheuristics can be classified into trajectory-based or population-based algorithms. The former explores the solution space using a single solution that is iteratively improved, while the latter maintains and evolves a set of candidate solutions simultaneously. This evolutionary approach has shown particular success in combinatorial optimization problems such as routing, scheduling, and layout design, where evolutionary learning-based strategies are able to exploit problem-specific structures [Rahman et al., 2021]. Meanwhile, applications in continuous optimization contexts are steadily gaining attention, with recent studies exploring enhanced mutation and crossover mechanisms to improve search diversity and convergence rates [Ghasemi et al., 2023; Salgotra and Gandomi, 2024].

Moreover, resource-aware strategies, such as iterative division, sampling, and as population size adaptation, are also being investigated to enhance both local and global search while reducing computational costs without compromising solution quality. Furthermore, evolutionary algorithms can also benefit from online learning techniques aimed at mitigating premature convergence [Salgotra and Gandomi, 2024], as well as from the use of Gaussian mixture models to improve the balance between exploration and exploitation [Fonseca et al., 2023]. These advances represent a promising direction toward intelligent optimization systems that are robust, general-purpose, and capable of self-improvement and adaptation across problem domains.

Metaheuristics can also be hybridized by introducing problem-specific knowledge or by combining different algorithms to optimize performance. In particular, data mining has proven to be a powerful tool when integrated with metaheuristics: trajectory-based and population-based metaheuristics alike can benefit from data-driven approaches, where the extraction of relevant patterns can guide the search process and enhance adaptability to diverse problem spaces. Hybrid algorithms such as the *DM-GRASP* [Ribeiro et al., 2006] and *C-MDE* [Schoen and Tigli, 2021] exemplify the vantages of data mining hybridization in trajectory-based and population-based metaheuristics.

The inclusion of data mining techniques in the specific context of Differential Evolution (DE) hybridization aims to address DE's limitations by incorporating both adaptive mechanisms and search operations inspired by data-driven insights. For example, the use of a clustering-based mutation operator in a DE algorithm facilitates population segmentation, enabling promising regions to be explored more intensively based on statistical indicators [Mousavirad et al., 2021]. This approach seeks to guide the mutation process toward more favorable areas of the search space, thereby enhancing exploration effectiveness and preventing stagnation in local optima.

Additionally, metaheuristics can also incorporate cooperative methods to balance exploration and exploitation, as demonstrated in Cai et al. [2024]. In that work, a cooperative metaheuristic algorithm (*CMA*), inspired by heterosis theory, is proposed to address the susceptibility to local optima in large search spaces and slow convergence, by subdividing the population into three subpopulations that evolve cooperatively. Similar approaches include the integration of a collaborative mutation strategy into *LSHADE* [Zhao et al., 2020], and the development of a Cooperative Co-evolutionary *LSHADE* for large-scale optimization (*CCLSHADE*) [Sharawi and El-Abd, 2017].

Recently, a new data mining-based variant of *LSHADE*, called *DM-LSHADE*, was proposed by Santos et al. [2025]. This hybrid data-driven approach extracts patterns from already generated solutions and uses them to guide the search in subsequent generations. For this purpose, an elite set is formed, and clustering is applied to approximate the most frequent values of each decision variable in the continuous domain, yielding a representative pattern. These patterns are then merged into the population and used to generate new candidate solutions during the standard operations of *LSHADE*, enhancing the algorithm's ability to continuously evolve by focusing its search on the most promising regions of the solution space.

Although *DM-LSHADE* has achieved competitive results on the Congress on Evolutionary Computation's benchmark functions, it still suffers from computationally expensive data mining steps and remains susceptible to stagnation in local optima and slow convergence, particularly in high-dimensional problems. To address these challenges, this work builds upon *DM-LSHADE* and proposes a cooperative co-evolutionary algorithm, called *CDM-LSHADE*, which employs message-passing parallel communication for pattern exchange and external data mining, thereby reducing computational overhead and enabling improved solution quality. The specifics mechanisms of *DM-LSHADE* are discussed in the following section, while *CDM-LSHADE* is presented in Section 4.

3. Background

Our proposed cooperative algorithm, *CDM-LSHADE*, is built upon *DM-LSHADE*, a hybrid variant of *LSHADE* that incorporates data mining techniques to enhance performance. *LSHADE*, in turn, improves the *SHADE* algorithm by introducing a linear population size reduction strategy. Finally, the *SHADE* algorithm, which is built upon the foundational Differential Evolution (*DE*), introduces self-adaptive mechanisms that enable the dynamic adjustment of control parameters based on past successes. In this section, each of the algorithms mentioned above will be presented, providing the necessary foundation to understand this work.

3.1. Diferencial Evolution (DE)

In 1995, Storn and Price [1995] introduced Differential Evolution, an evolutionary meta-heuristic designed to solve continuous global optimization problems. The algorithm operates through three control parameters — Population Size, Mutation Factor, and Crossover Probability — and three main mechanisms: mutation, crossover, and selection. Mutation is done by adding a weighted difference between two population vectors to a third, creating a new vector. Crossover combines this vector with the original individual to form a trial solution. Then, selection compares the trial solution with the target and chooses the better of the two for the next generation. The effectiveness of DE, along with its simplicity, robustness, ease of implementation, and flexibility, has led to its widespread use in real-world optimization tasks, also leading to the proposal of many variants.

3.2. Success-History Based Adaptive DE (SHADE)

One of the well-known variants of DE is the *Success-History based Adaptive Differential Evolution (SHADE)* [Tanabe and Fukunaga, 2013]. This variant uses a historical memory of past successful values of the Mutation Factor (F) and Crossover Probability (CR) to adaptively update parameter values during the optimization process. That is, when a new trial solution achieves better fitness than its parent, the corresponding values of F and CR are stored in memory and later used to guide parameter generation in subsequent iterations. This self-adaptive strategy enables *SHADE* to balance exploration and exploitation effectively throughout the search process.

3.3. Linearly Population Size Reduction SHADE (LSHADE)

In the year following the proposition of the *SHADE* algorithm, Tanabe and Fukunaga [2014] also introduced a variant called *Linear Population Size Reduction SHADE (LSHADE)*. This version further extends *SHADE* with Linear Population Size Reduction (LPSR), which continually decreases the population size according to a linear function. This strategy allows for broad exploration of the search space in the early stages, followed by more focused exploitation in later stages. Such dynamic adjustments not only help reduce computational overhead but also improve convergence speed and solution quality throughout the optimization process.

3.4. Data Mining LSHADE (DM-LSHADE)

More recently, Santos et al. [2025] presented a new hybridized data mining-based variant of *LSHADE*, called *DM-LSHADE*. This hybrid data-driven approach extracts patterns from already generated solutions and uses them to guide the search in subsequent generations. To this end, an elite set $\mathbf{E}$ is formed from the best solutions found since the beginning of the search and iteratively updated with the best solutions from each generation, serving as a memory of promising regions in the search history. Then, at regular generation intervals ΔG , the elite set is used to extract patterns that are treated as new candidate solutions and used to generate new individuals.

To extract patterns, the *K-means* clustering algorithm is used to identify representatives with low dispersion, specifically the centroids of elite groups with a high concentration of closely located solutions. In this way, the solutions within a cluster are considered similar and its centroid represents a pattern corresponding to a promising region. To exploit the information encoded in these patterns, they are inserted into the population and used to generate new solutions through the standard operations of *LSHADE*. The pseudocode of *DM-LSHADE* is shown in the Algorithm 1.

The main changes of *DM-LSHADE* compared to *LSHADE* are in lines 3-7, where the population is sorted by fitness, the best individuals are added to the elite set, and patterns are extracted by a mining process and injected into the population. This concise implementation allows the algorithm to serve as a solid foundation for our cooperative variant. Moreover, *DM-LSHADE* will also be used as a non-cooperative strategy in the computational experiments, aiming to provide a comparative baseline for evaluating the proposed solution, as described in Section 5.

Algorithm 1 DM-LSHADE — Adapted from [Santos et al., 2025]

```

1: Population  $\leftarrow$  INITIALIZEPOPULATION()
2: while stopping criteria are not met do
3:   Elite  $\leftarrow$  UPDATEELITE(Population)
4:   if is a hybrid generation then  $\triangleright$  Data Mining
5:     Patterns  $\leftarrow$  EXTRACTPATTERNS(Elite)
6:     INSERTPATTERNS(Patterns, Population)
7:   end if
8:   for  $i \leftarrow 1$  to Population.size do
9:      $\mathbf{x}_{i,G+1} = \text{GENERATENEWINDIVIDUAL}(\mathbf{x}_{i,G})$   $\triangleright$  Analogous to LSHADE
10:  end for
11: end while

```

4. Proposal

Although *DM-LSHADE* has demonstrated competitive performance on the Congress on Evolutionary Computation benchmark functions, it remains limited by the high computational cost of its data mining step and is prone to stagnation in local optima and slow convergence, especially in high-dimensional settings. To address these challenges, this work proposes a cooperative co-evolutionary algorithm, called *CDM-LSHADE*, which employs message-passing parallel communication for pattern exchange and external data mining.

Our hypothesis is that a cooperative and co-evolutionary parallel variant of *DM-LSHADE*, designed to leverage distributed computing resources and structured process collaboration, can effectively mitigate the major limitations currently affecting its performance by:

1. Introducing a dedicated master process responsible for pattern extraction to offload the high computational cost of data mining from the agents, thereby preventing data mining overhead from interfering with the evolutionary optimization tasks.
2. Mitigating slow convergence and stagnation in local optima through cooperative parallel strategies that enable search processes to share regions of high potential for exploitation, thereby reducing stagnation and accelerating convergence.

4.1. Cooperative Data Mining LSHADE (*CDM-LSHADE*)

The overall functioning of *CDM-LSHADE* consists of splitting the tasks of Data Mining and Evolutionary Optimization, originally performed serially by *DM-LSHADE*, between two types of processes. The first is the Master Process (or Miner), which is mainly responsible for pattern mining, communication, and managing the other processes. These, in turn, are called Agents Processes (also Workers or Evolutionary Optimizers), and are responsible for the evolutionary optimization process. Put simply, the master process handles the data mining (DM) component of *DM-LSHADE*, whereas the agents processes are responsible for the *LSHADE*-based optimization.

In order for the master process to perform pattern mining, given that it is not allowed to execute any optimization activities, it must establish consistent communication with the other processes. Thus, the master must receive the elite individuals from all agents processes. Once the master has collected elites from all agents, ensuring diversity, as no worker is excluded from the mining process, it applies a clustering algorithm (in this case *K-means*) to identify patterns. Once the mining stage is completed, these patterns are broadcast back to the agents. This process is repeated throughout the algorithm's execution, until the master receives a termination message from all agents, at which point it finalizes its execution and returns the best solution found.

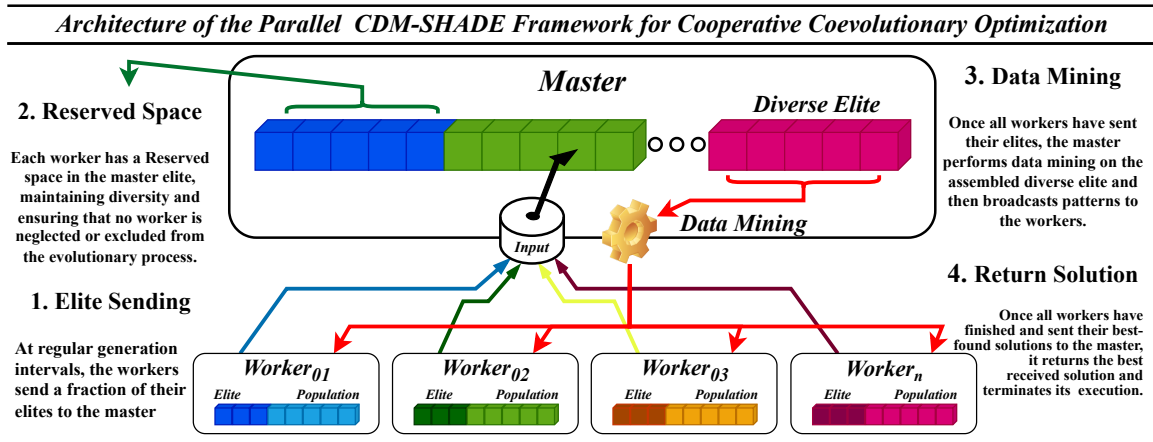


Figure 1: Visualization of the Master-slave modeling of the *CDM-LSHADE*

On the other hand, the agents processes, although primarily dedicated to evolutionary optimization, also require communication with the master to implement a data-driven approach. In this context, the agents not only periodically send elite individuals, but also receive patterns sent by the master and incorporate them into their populations. To address this, due to the asynchrony introduced by the delay between sending elites and receiving patterns, it is essential to check for incoming messages from the master at least once per generation. This ensures that the discovered patterns are integrated as early as possible, enabling more directed and efficient exploration.

This process, illustrated in Figure 1, adopts the *Multi-Process Same Data (MPSD)* architectural paradigm, in which all processes operate over a logically shared search space. That is, although each process independently evolves its own subpopulation, they periodically cooperate through shared knowledge extracted from a common elite set. This strategy balances exploration and exploitation across processes, enabling more informed local decisions based on global insights. Naturally, to achieve the proposed model, it is necessary to employ a parallel programming library.

For this work, the *Message Passing Interface (MPI)* was chosen due to its simplicity and widespread adoption in distributed computing environments. However, it is important to note that functional parallelism libraries play a secondary role in this context, serving primarily as middleware to support the execution architecture. Additionally, to avoid unnecessary overhead, non-blocking communication instructions were employed. For this purpose, auxiliary buffers were used to enable buffered message exchanges without relying on blocking operations. The pseudocode of *CDM-LSHADE* is didactically presented in Algorithms 2,3, and 4.

Algorithm 2 CDM-LSHADE: Main

```

1: MPI_INICIALIZE( )
2: ALLOCATEBUFFERS( )
3: INITIALIZEPARAMETERS( )
4: if rank == Master then
5:   MASTER( )
6: else
7:   WORKER( )
8: end if
9: MPI_FINALIZE( )
  
```

Algorithm 3 CDM-LSHADE: Master Process

```

1: while not all workers have finished do
2:   while new Messages do ▷ Receiving Messages
3:     if Solution Menssage then
4:        $\text{Elite}_n \leftarrow \text{new Worker}_n.\text{Elite}$ 
5:        $\text{MiningFlag} \leftarrow \text{True}$ 
6:     else if Finalization Menssage then
7:        $\text{FinalSolutions}[n] \leftarrow \text{Worker}_n.\text{Solution}$ 
8:     end if
9:   end while
10:  if MiningFlag then
11:     $\text{Patterns} \leftarrow \text{PATTERNMINING}()$  ▷ Data Mining
12:     $\text{BROADCAST}(\text{Patterns})$  ▷ Cooperative
13:     $\text{MiningFlag} \leftarrow \text{False}$ 
14:  end if
15: end while
16: return BEST(FinalSolutions)

```

Algorithm 4 CDM-LSHADE: Worker Process

```

1:  $\text{Population} \leftarrow \text{INITIALIZEPOPULATION}()$ 
2: while termination criteria not met do
3:    $\text{Elite} \leftarrow \text{UPDATEELITE}(\text{Population})$ 
4:   while new Messages do ▷ Receiving Messages
5:      $\text{Patterns} \leftarrow \text{RECEIVEPATTERNS}()$ 
6:      $\text{INSERTPATTERNS}(\text{Patterns}, \text{Population})$ 
7:   end while
8:   if  $(++\Delta G) \geq \text{Send Interval}$  then ▷ Cooperative
9:      $\text{SEND}(\text{Elite})$ 
10:     $\Delta G \leftarrow 0$ 
11:   end if
12:   for  $i \leftarrow 1$  to  $\text{Population.size}$  do
13:      $\mathbf{x}_{i,G+1} = \text{GENERATENEWINDIVIDUAL}(\mathbf{x}_{i,G})$  ▷ Analogous to LSHADE
14:   end for
15: end while
16:  $\text{SEND}(\text{Final Solution})$ 

```

5. Computational Experiments

This section presents the results and analysis of the computational experiments to evaluate the proposed cooperative algorithm, comparing it with its baseline algorithms *DM-LSHADE* and *LSHADE*. The benchmark functions are described in Subsection 5.1, followed by Subsection 5.2, which presents the adopted parameter configuration. Subsection 5.3 illustrates the behavior of *CDM-LSHADE* on a single benchmark function, and Subsection 5.4 reports the obtained results along with a comprehensive discussion. All experiments were implemented in **C++** and performed on a 12th Gen Intel® Core™ i7-12700F × 12 cores (8 Performance / 4 Efficient cores) with 32GB RAM. The code is available at <https://github.com/VICTORMARTINEZ2002/CDM-LSHADE>.

5.1. Benchmark Functions

For a comparison evaluation of *CDM-LSHADE*, the functions from the CEC 2014 Competition on Single Objective Bound-Constrained Numerical Optimization benchmark suite were used [Liang et al., 2013]. This suite consists of 30 test functions with a search space of $[-100, 100]^D$. Functions F1 to F3 are unimodal, while F4 to F16 are basic multimodal functions. F17 to F22 are hybrid functions, where variables are randomly divided into subcomponents, each with a different basic function. Finally, F23 to F30 are composite functions, combining multiple test problems into a complex landscape.

5.2. Parameters Configuration

Parameter calibration in parallel code can present significant challenges, especially when it comes to adjusting variables related to the number of processors. The behavior of the code can vary substantially depending on the process allocation, for example, configurations that are optimal for 2 processes may perform poorly with 8 processes. In this context, to avoid the need for specific calibration for each processor configuration, it was decided to inherit the parameters from the baseline algorithm *DM-LSHADE* [Santos et al., 2025], allowing for a simplified global configuration through the reuse of previously adjusted settings. *CDM-LSHADE*'s exclusive parameters were configured to balance the significance of individual transmission and the reduction of overhead. The parameters configuration is shown in Table 1.

Parameter	Description	Type	Value Setting
Elite Rate	Fraction of population selected as Elites	Ordinal	0.1
Cluster Rate	Patterns to be Extracted (K-means)	Real	0.1468
Hybrid Generation	Hybridization interval (Generations)	Integer	168
Send Rate	Fraction of individuals sent to Master	Ordinal	0.24

Table 1: Control parameters used in *CDM-LSHADE*.

5.3. CDM-LSHADE Behavior

Before presenting the full results, it is important to analyze the algorithm's behavior under varying numbers of processes and compare it with the baseline algorithms. This analysis provides valuable insights into the algorithm's performance, scalability, and efficiency. To this end, 25 independent executions were performed for *LSHADE*, *DM-LSHADE* and each *CDM-LSHADE* configuration, varying the number of processors from 2 to 10, allowing for a assessment of how increased computational resources affect solution quality and overall execution time. Figure 2 presents the average solution quality obtained across the runs for the shifted Schwefel's function with dimensions 20 and 100, respectively.

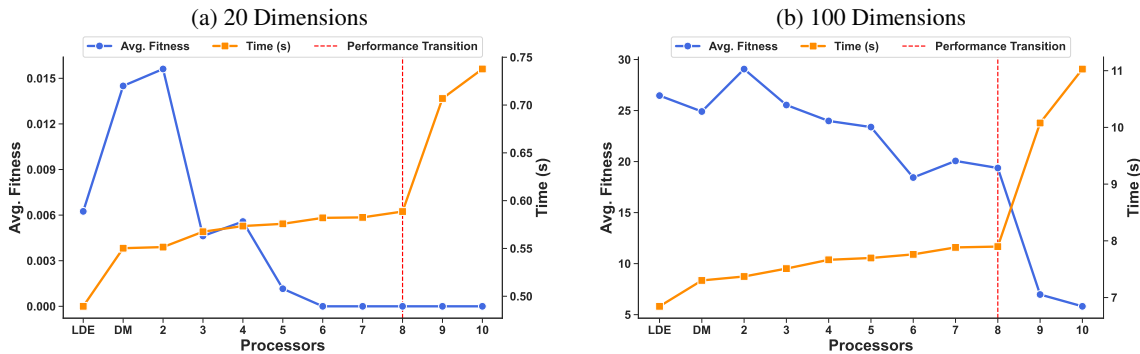


Figure 2: Comparison of the average solution quality and execution time for *LSHADE* (LDE), *DM-LSHADE* (DM), and *CDM-LSHADE* on the shifted Schwefel's function in 20 and 100 dimensions.

From Figure 2, for both 20 and 100 dimensions, it is possible to observe the reduction in fitness as more processes are incorporated into *CDM-LSHADE*, suggesting that the cooperative strategy is advantageous. Specifically, for the 20-dimensional case, a significant improvement in fitness is seen from 2 to 3 processes and from 4 to 6 processes, after which the algorithm was able to reach the optimal solution. Additionally, for the 100-dimensional execution, a noticeable improvement in fitness is observed once the landmark of 8 processes is surpassed, at the cost of a considerable increase in execution time.

This increase in execution time can be attributed to the processor's architecture (12 cores). Once all the 8 performance cores are fully utilized, additional processes must be assigned to efficiency cores, which leads to a decrease in execution speed. This slowdown results in longer execution times, allowing the processes to receive more patterns from the master, which remains active until all processes complete. In hardware with more performance cores (or with cores of the same type), it is expected that such a slowdown would be delayed or, in some cases, not occur at all. In these scenarios, the algorithm would likely experience a more consistent performance, as seen in executions prior to the landmark.

5.4. Validation

To validate the effectiveness of the proposed algorithm, experiments were conducted using the benchmark suite from CEC 2014 (Subsection 5.1). For each function, a total of 25 independent runs were performed for *LSHADE*, *DM-LSHADE*, and *CDM-LSHADE* across dimensions [10, 20, 30, 50, 100], with the stopping criterion set to $(10,000 \times \text{Dimension})$ function evaluations. For *CDM-LSHADE*, the number of processes used ranged from 2 to 10. The algorithm is considered to have reached the optimum if the gap between the best-found solution and the optimum is equal to or smaller than 10^{-8} .

Table 2 illustrates this comparison for the *CDM-LSHADE* (6 processors) in a 100-dimensional space, providing information on best cost, average cost and standard deviation of cost. Additionally, it also compares time performance with the reference algorithms and reports the number of victories in each category. Furthermore, to assess whether there is a statistically significant difference between the algorithms, pairwise Wilcoxon signed-rank tests were applied with a 95% confidence level on each function.

Function (100 Dim.)	LSHADE				DM-LSHADE				CDM-LSHADE (6 Processors)				Time Comparison	
	Best Cost	Avg. Cost	Std. Cost	Avg. Time(s)	Best Cost	Avg. Cost	Std. Cost	Avg. Time(s)	Best Cost	Avg. Cost	Std. Cost	Avg. Time(s)	LSHADE Time Gap(%)	DM-LSHADE Time Gap(%)
1	9.420e+04	1.626e+05	5.248e+04	2.025e+01	7.481e+04	1.633e+05	5.510e+04	2.085e+01	7.243e+04	<u>1.242e+05</u> †	3.117e+04	2.191e+01	7.58	4.82
2	0.000e+00	0.000e+00	0.000e+00	1.966e+01	0.000e+00	0.000e+00	0.000e+00	2.013e+01	0.000e+00	0.000e+00	0.000e+00	2.127e+01	7.55	5.36
3	0.000e+00	0.000e+00	0.000e+00	1.953e+01	0.000e+00	0.000e+00	0.000e+00	1.997e+01	0.000e+00	0.000e+00	0.000e+00	2.112e+01	7.50	5.43
4	1.435e+02	1.795e+02	3.165e+01	1.985e+01	1.376e+02	1.853e+02	2.955e+01	2.037e+01	1.435e+02	1.820e+02	3.076e+01	2.147e+01	7.53	5.13
5	2.049e+01	2.056e+01	3.981e-02	2.063e+01	2.029e+01	2.043e+01	9.289e-02	2.100e+01	2.005e+01	<u>2.027e+01</u> †	1.127e-01	2.218e+01	7.01	5.31
6	3.038e+00	6.841e+00	2.217e+00	1.882e+02	2.338e+00	6.208e+00	2.170e+00	1.888e+02	1.195e+00	5.753e+00	2.367e+00	1.989e+02	5.39	5.10
7	0.000e+00	0.000e+00	0.000e+00	2.021e+01	0.000e+00	0.000e+00	0.000e+00	2.068e+01	0.000e+00	0.000e+00	0.000e+00	2.192e+01	7.78	5.65
8	1.048e-03	1.301e-02	1.041e-02	3.605e+00	4.928e-03	6.412e-02	9.193e-02	4.014e+00	3.885e-05	<u>5.282e-03</u> †	3.044e-03	4.295e+00	16.06	6.53
9	4.975e+00	2.834e+01	7.798e+00	2.086e+01	5.970e+00	2.010e+01	7.464e+00	2.130e+01	9.950e+00	<u>1.953e+01</u> †	4.680e+00	2.257e+01	7.54	5.63
10	1.998e+01	2.647e+01	5.635e+00	6.845e+00	1.441e+01	2.491e+01	6.678e+00	7.302e+00	7.871e+00	<u>1.844e+01</u> †	5.393e+00	7.761e+00	11.80	5.92
11	9.695e+03	1.087e+04	4.997e+02	2.456e+01	8.791e+03	9.997e+03	7.000e+02	2.481e+01	7.882e+03	<u>9.278e+03</u> †	8.006e+02	2.631e+01	6.64	5.68
12	3.573e-01	4.320e-01	3.888e-02	6.093e+01	2.253e-01	3.508e-01	6.599e-02	6.352e+01	2.269e-01	<u>3.111e-01</u> †	6.487e-02	6.470e+01	5.82	1.82
13	2.120e-01	2.413e-01	1.627e-02	1.927e+01	1.899e-01	2.391e-01	2.520e-02	1.973e+01	1.795e-01	<u>2.183e-01</u> †	2.027e-02	2.093e+01	7.92	5.72
14	8.464e-02	1.061e-01	9.991e-03	1.942e+01	8.828e-02	1.070e-01	8.583e-03	1.971e+01	8.577e-02	9.963e-02	8.214e-03	2.082e+01	6.73	5.36
15	1.554e+01	1.669e+01	9.092e-01	2.125e+01	1.261e+01	1.531e+01	1.511e+00	2.170e+01	1.255e+01	<u>1.447e+01</u> †	1.084e+00	2.292e+01	7.31	5.36
16	3.818e+01	3.917e+01	5.166e-01	2.142e+01	3.621e+01	3.834e+01	7.092e-01	2.184e+01	3.629e+01	3.782e+01	6.858e-01	2.306e+01	7.14	5.30
17	3.306e+03	4.220e+03	4.606e+02	2.151e+01	3.004e+03	4.374e+03	7.585e+02	2.206e+01	1.875e+03	<u>4.161e+03</u> †	9.418e+02	2.325e+01	7.47	5.13
18	2.095e+02	2.243e+02	1.268e+01	2.025e+01	1.900e+02	2.225e+02	1.800e+01	2.080e+01	1.873e+02	2.205e+02	1.609e+01	2.189e+01	7.51	4.99
19	9.100e+01	9.462e+01	2.326e+00	5.508e+01	9.183e+01	9.486e+01	1.903e+00	5.568e+01	9.256e+01	9.571e+01	2.213e+00	5.871e+01	6.19	5.15
20	8.261e+01	1.410e+02	4.409e+01	2.054e+01	7.523e+01	1.256e+02	3.544e+01	2.109e+01	7.211e+01	<u>1.171e+02</u> †	3.072e+01	2.223e+01	7.63	5.14
21	1.680e+03	2.426e+03	5.504e+02	2.111e+01	1.043e+03	2.451e+03	5.742e+02	2.170e+01	1.546e+03	2.326e+03	4.186e+02	2.287e+01	7.69	5.13
22	4.231e+02	9.747e+02	2.128e+02	2.584e+01	4.673e+02	9.295e+02	2.429e+02	2.627e+01	5.343e+02	8.647e+02	1.921e+02	2.777e+01	6.95	5.39
23	3.482e+02	3.482e+02	5.832e-14	7.951e+01	3.482e+02	3.482e+02	5.789e-14	7.997e+01	2.000e+02	<u>2.000e+02</u> †	0.000e+00	8.446e+01	5.86	5.31
24	3.877e+02	3.936e+02	3.023e+00	4.482e+01	3.886e+02	3.937e+02	2.350e+00	4.538e+01	2.000e+02	<u>2.000e+02</u> †	0.000e+00	4.820e+01	7.00	5.84
25	2.000e+02	2.000e+02	0.000e+00	6.336e+01	2.000e+02	2.000e+02	0.000e+00	6.385e+01	2.000e+02	2.000e+02	0.000e+00	6.733e+01	5.90	5.16
26	2.000e+02	2.000e+02	0.000e+00	2.774e+02	2.000e+02	2.000e+02	0.000e+00	2.783e+02	2.000e+02	2.000e+02	0.000e+00	2.918e+02	4.93	4.61
27	3.111e+02	3.384e+02	2.568e+01	2.676e+02	3.045e+02	3.484e+02	2.466e+01	2.682e+02	2.000e+02	<u>2.000e+02</u> †	0.000e+00	2.851e+02	6.12	5.91
28	2.177e+03	2.313e+03	5.608e+01	1.027e+02	2.255e+03	2.317e+03	3.594e+01	1.036e+02	2.000e+02	<u>2.000e+02</u> †	0.000e+00	1.089e+02	5.68	4.91
29	7.521e+02	9.156e+02	1.958e+02	9.822e+01	7.151e+02	8.793e+02	1.292e+02	9.887e+01	2.000e+02	<u>3.253e+02</u> †	2.459e+02	1.041e+02	5.69	5.07
30	7.742e+03	9.067e+03	9.418e+02	6.684e+01	7.551e+03	9.290e+03	8.562e+02	6.736e+01	2.000e+02	7.828e+03	3.234e+03	7.102e+01	5.89	5.15
# Wins (Sig.)	4	2 (0)	8	30	4	0 (0)	5	0	17	23 (16/10)	12	0	7.26	5.23

Table 2: Statistical comparison of *CDM-LSHADE* (6 processors) against *LSHADE* and *DM-LSHADE*.

Bold numbers indicate the best average value among the algorithms; † denotes a statistically significant difference from *LSHADE*; and Underline indicates a statistically significant difference from *DM-LSHADE*.

It can be observed that *CDM-LSHADE*, configured here for 6 processors, achieved not only better results in the *Best Cost* category, winning overall in 17 out of 30 functions, but also showed more favorable results in terms of the average cost of the solutions found, securing overall victories in 23 out of 30 functions. Among these, 16 victories showed statistical significance compared to *LSHADE*, and 10 showed statistical significance compared to *DM-LSHADE*. Furthermore, the cooperative strategy also demonstrated more consistent results, with 12 victories in the *Standard Deviation* category. However, regarding execution time, a slight average increase was observed: 7.26% compared to *LSHADE* and 5.23% compared to *DM-LSHADE*.

Such a slight increase in execution time occurs because, although the computational cost of data mining was offloaded to a dedicated processor (Master), the need for communication introduced an additional level of complexity to the algorithm, thus balancing the relief of outsourcing the data mining process. It is also important to note that each agent process performs the same number of evaluations as their serial versions (10,000D), so if the number of processes is greater than 2, the *CDM-LSHADE* overall performs more function evaluations (FE), while still maintaining a viable execution time due to parallelism. The total number of objective function calls can be calculated using Equation 1, shown below.

$$\text{Total FE} = (P - 1) (10.000 \text{ Dim}) \quad (1)$$

5.5. Results Compilation

The compilation of the results of the experiments described in the Validation Subsection (5.4) is presented in Table 3. For didactic and visualization purposes, the table displays only the results for even numbers of processors, although the final analysis considers all experiments, from 2 to 10 processors.

Algorithm / Processes	Dimension 10					Dimension 20					Dimension 30					Dimension 50					Dimension 100					Average Result (%)	
	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10		
Best Cost	CDM-LSHADE	7	9	14	16	15	7	14	18	21	21	9	12	16	14	20	12	14	15	14	21	10	10	17	14	21	46.15
	DM-LSHADE	10	8	3	1	1	12	8	4	1	8	9	8	5	6	2	11	10	9	10	3	12	11	4	8	1	23.19
	LSHADE	3	3	2	2	2	3	0	0	0	2	4	2	1	2	0	2	1	1	1	1	2	3	4	3	3	6.37
Avg. Cost	CDM-LSHADE	9	14	21	24	23	9	14	21	22	20	9	16	22	19	20	11	16	20	19	22	13	13	23	21	23	59.48
	DM-LSHADE	12	8	2	0	1	12	8	1	1	3	11	5	0	3	2	11	8	5	5	3	9	9	0	3	0	16.37
	LSHADE	4	2	2	1	1	4	1	1	0	0	4	2	1	1	1	5	3	2	3	2	3	3	2	1	2	6.45
Std. Cost	CDM-LSHADE	12	14	19	19	19	14	16	15	16	14	11	10	13	12	14	10	18	14	16	15	13	13	12	16	13	48.30
	DM-LSHADE	6	5	3	1	3	5	4	4	4	7	6	7	6	5	4	5	3	4	5	5	4	6	5	6	3	15.33
	LSHADE	6	4	2	4	2	4	3	4	3	2	6	5	3	5	5	11	5	8	5	6	8	6	8	3	9	16.66
Significances	DM-LSHADE	4	6	15	17	16	7	10	15	20	12	7	9	15	12	10	8	7	8	9	18	5	9	10	8	15	36.44
	LSHADE	4	7	17	18	20	8	13	19	21	20	7	11	16	16	16	8	13	15	17	19	5	11	16	13	18	46.66
Time Gap (%)	DM-LSHADE	-0.2	2.8	4.7	6.5	19.3	0.5	4.0	5.3	7.0	27.1	0.8	4.2	5.7	7.2	30.9	0.8	4.1	5.7	7.5	33.8	0.4	3.6	5.2	7.3	35.7	9.36
	LSHADE	6.7	9.6	11.3	13.0	25.1	8.2	11.4	12.6	14.1	32.9	7.2	10.3	11.7	13.2	35.5	5.1	8.3	9.8	11.5	36.7	2.5	5.6	7.3	9.3	37.1	14.40

Table 3: Summary of experimental results across multiple dimensions and processor counts.

Overall, the average cost of the solutions generated by the proposed algorithm was lower in 59.48% of the cases, while the reference algorithms achieved lower solutions only in 16.37% (*DM-LSHADE*) and 6.45% (*LSHADE*) of the executions. Furthermore, the experiments indicated that the proposed algorithm provided statistically superior solutions in 36.44% of the executions when compared to *DM-LSHADE*, and in 46.66% when compared to *LSHADE*. Moreover, *CDM-LSHADE* achieved the best solutions at the end of 25 runs in 46.15% of the trials, whereas *DM-LSHADE* and *LSHADE* achieved them in only 23.19% and 6.37%, respectively. Additionally, *CDM-LSHADE* provided steadier solutions in 48.30% of the runs.

Regarding efficiency, the average execution time increased by up to 10.0% compared to *DM-LSHADE*, due to communication overhead, which counterbalances the benefit of outsourcing the data mining process. However, when considering only the executions prior to the hardware limitation (8 cores), an increase in execution time of less than 5% was observed. In this context, it can be verified that the proposed algorithm, *CDM-LSHADE*, provides higher-quality solutions at a slight increase in execution time, assuming sufficient hardware is available. Thus, it is understood that adopting a parallel cooperative strategy can be seen as advantageous for mitigating slow convergence and preventing stagnation in local optima, while maintaining execution times close to those of the base algorithm.

6. Conclusion

This study proposes a cooperative co-evolutionary hybridized metaheuristic, *CDM-LSHADE*, for continuous optimization, resulting from the combination of mining techniques and parallelism. This novel approach employs non-blocking message-passing parallel communication for cooperative pattern exchange and external data mining. The algorithm is designed to leverage distributed computing resources and structured process collaboration by offloading the high computational cost of data mining to a dedicated process for pattern extraction. It also aims to mitigate slow convergence and stagnation in local optima through cooperative strategies that allow search processes to share regions of high potential.

A detailed evaluation of the algorithm was conducted by comparing it with its baseline algorithms using the functions from CEC 2014. The results showed that the proposed algorithm achieved a better average fitness in 59.48% of the runs. Furthermore, the experiments indicated statistical significance in 36.44% of the cases when compared to *DM-LSHADE*, and 46.66% when compared to *LSHADE*. Additionally, the algorithm produced more consistent solutions in 48.30% of the runs, incurring only a marginal increase in execution time, provided sufficient hardware resources are available. Therefore, adopting a cooperative strategy proves effective in mitigating slow convergence and preventing stagnation in local optima.

Future work will focus on integrating new clustering approaches for better pattern extraction, as well as exploring other strategies for transmission and insertion of individuals into the mining elite. The validation phase will also be extended to include more algorithms and others benchmark functions. Furthermore, we expect to apply the proposed algorithm into real-world applications. This research contributes to the literature on evolutionary algorithms by introducing a novel approach to Cooperative Data-Driven Metaheuristics, advancing the development of parallel and hybrid optimization techniques through improved information sharing, clustering-driven guidance, and dynamic cooperation among processes.

Acknowledgements: This work was partially supported by FAPERJ (E-26/201.139/2022) and CNPq (315750/2021-9).

References

- Almeida, A. L. B., Lima, J. d. C., and Carvalho, M. A. M. (2022). Systematic literature review on parallel trajectory-based metaheuristics. *ACM Computing Surveys*, 55(8):1–34.
- Bujok, P. and Kolenovský, P. (2022). In *Eigen Crossover in Cooperative Model of Evolutionary Algorithms Applied to CEC 2022 Single Objective Numerical Optimisation*, p. 1–8.
- Cai, T., Zhang, S., Ye, Z., Zhou, W., Wang, M., He, Q., Chen, Z., and Bai, W. (2024). Cooperative metaheuristic algorithm for global optimization and engineering problems inspired by heterosis theory. *Scientific Reports*, 14(1).
- Eiben, A. and Smith, J. (2015). From evolutionary computation to the evolution of things. *Nature*, 521:476–82.
- Fonseca, T. H. L., Nassar, S. M., de Oliveira, A. C. M., and Agard, B. (2023). Low-dimensional space modeling-based differential evolution for large-scale global optimization problems. *IEEE Transactions on Evolutionary Computation*, 27(5):1529–1543.
- Ghasemi, M., Zare, M., Trojovský, P., Zahedibialvaei, A., and Trojovská, E. (2023). A hybridizing-enhanced differential evolution for optimization. *PeerJ Computer Science*, 9:e1420.
- Liang, J., Qu, B., and Suganthan, P. (2013). Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization.
- Mousavirad, S. J., Schaefer, G., Korovin, I., Moghadam, M. H., Saadatmand, M., and Pedram, M. (2021). An enhanced differential evolution algorithm using a novel clustering-based mutation operator. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, p. 176–181.
- Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition. ISBN 978-0-387-30303-1.
- Pardalos, P. M. and Resende, M. G. C. (2002). *Handbook of applied optimization*.
- Rahman, M. A., Sokkalingam, R., Mahmood, O., Biswas, K., Abdullah, L., and Abdul Kadir, E. (2021). Nature-inspired metaheuristic techniques for combinatorial optimization problems: Overview and recent advances. *Mathematics*, 9.
- Ribeiro, M., Plastino, A., and Martins, S. (2006). Hybridization of grasp metaheuristic with data mining techniques. *J. Math. Model. Algorithms*, 5:23–41.
- Salgotra, R. and Gandomi, A. H. (2024). A novel multi-hybrid differential evolution algorithm for optimization of frame structures. *Scientific Reports*, 14(1):4877.
- Santos, G. R., Plastino, A., and de Oliveira, A. C. M. Combining a differential evolution heuristic with data mining. Manuscript submitted for review to *Neural Computing and Applications*, 2025.
- Schoen, F. and Tigli, L. (2021). Efficient large scale global optimization through clustering-based population methods. *Computers & Operations Research*, 127:105165. ISSN 0305-0548.
- Sharawi, M. and El-Abd, M. (2017). A cooperative co-evolutionary lshade algorithm for large-scale global optimization. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, p. 1–8.
- Storn, R. and Price, K. (1995). Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 23.
- Talbi, E.-G. (2021). Machine learning into metaheuristics: A survey and taxonomy. *ACM Computing Surveys*, 54:1–32.
- Tanabe, R. and Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. p. 71–78. ISBN 978-1-4799-0453-2.
- Tanabe, R. and Fukunaga, A. (2014). Improving the search performance of shade using linear population size reduction.
- Zhao, F., Zhao, L., Wang, L., and Song, H. (2020). A collaborative lshade algorithm with comprehensive learning mechanism. *Applied Soft Computing*, 96:106609. ISSN 1568-4946.
- Zhuo, Y., Zhang, T., Du, F., and Liu, R. (2023). A parallel particle swarm optimization algorithm based on gpu/cuda. *Applied Soft Computing*, 144:110499. ISSN 1568-4946.