MDPI

Article

Combining Parallel Stochastic Methods and Mixed Termination Rules in Optimization

Vasileios Charilogis[1], Ioannis G. Tsoulos[2,*], Anna Maria Gianni[3]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilog@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; itsoulos@uoi.gr
[3] Department of Informatics and Telecommunications, University of Ioannina, Greece; am.gianni@uoi.gr
* Correspondence: itsoulos@uoi.gr;
† Current address: Department of Informatics and Telecommunications, University of Ioannina, Greece.
‡ These authors contributed equally to this work.

Abstract: Parallel optimization enables faster and more efficient problem-solving by reducing computational resource consumption and time. By simultaneously combining multiple methods, such as evolutionary algorithms and swarm-based optimization, effective exploration of the search space and achievement of optimal solutions in shorter time frames are realized. In this study, a combination of termination criteria is proposed, utilizing three different criteria to end the algorithmic process. These criteria include measuring the difference between optimal values in successive iterations, calculating the mean value of the cost function in each iteration, and the so-called "DoubleBox" criterion, which is based on the relative variance of the best value of the objective cost function over a specific number of iterations. The problem is addressed through the parallel execution of three different optimization methods (PSO, Differential Evolution, Multistart). Each method operates independently on separate computational units with the goal of faster discovery of the optimal solution and more efficient use of computational resources. The optimal solution identified in each iteration is transferred to the other computational units. The proposed enhancements were tested on a series of well-known optimization problems from relevant literature, demonstrating significant improvements in convergence speed and solution quality compared to traditional approaches.

Keywords: Global optimization; Parallel techniques; Termination rules; Evolutionary techniques

Citation: Charilogis V.; Tsoulos I.G.; Gianni A.M; Combining Parallel Stochastic Methods and Mixed Termination Rules in Optimization. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

The problem of finding the global minimum of a multidimensional function occurs in various scientific areas and has wide applications. In this the programmer seeks the absolute minimum of a function subject to some assumptions or constraints. The objective function is defined as $f : S \to R, S \subset R^n$ and the mathematical formulation of the global optimization problem is as follows:

$$x^* = \arg\min_{x \in S} f(x). \tag{1}$$

where the set S is as follows:

$$S = [a_1, b_1] \otimes [a_2, b_2] \otimes \ldots [a_n, b_n]$$

Such problems frequently arise in sciences like physics, where genetic algorithms prove effective in locating positions in magnetic plasmas and in developing optimization tools [1]. Additionally, the combined use of various optimization techniques enhances performance and stability in complex problems [2]. Furthermore, the hybrid approach that combines artificial physics techniques with particle swarm optimization has proven to be particularly effective in solving energy allocation problems [3]. In the field of chemistry, minimizing potential energy functions is crucial for understanding the ground states of molecular clusters

and proteins [4]. Optimizing these functions has been used for accurate protein structure prediction, with results closely matching crystal structures [5]. Moreover, a new computational framework combining variance-based methods with artificial neural networks has significantly improved accuracy and reduced computational cost in sensitivity analysis [6]. Even in the field of economics, optimization plays a crucial role. The particle swarm optimization (PSO) method is shown to be highly effective in solving the economic dispatch problem in power systems, providing higher quality solutions with greater efficiency compared to genetic algorithms [7]. Additionally, a multi-objective optimization approach for economic load dispatch and emission reduction in hydroelectric and thermal plants achieves optimal solutions that align with the user's preferences and goals [8]. Finally, optimization is also applied in biological process models [9] and medical data classification, achieving high accuracy in diagnosing diseases and analyzing genomes [10]. A series of methods have been proposed in the recent literature to handle problems of Equation 1. Methods used to handle problems of Equation 1 are usually divided into two categories: deterministic and stochastic methods. In the first category, the most common method is the interval method [11,12], where the set S is divided through a series of steps into subregions and some subregions that do not contain the global solution can be removed using some pre-defined criteria. The second category contains stochastic techniques that do not guarantee finding the total minimum, they are easier to program since they do not rely on some assumptions about the objective function. In addition, they also constitute the vast majority of global optimization methods. Among them, there are methods based on considerations derived from Physics, such as the Simulated Annealing method[14], the Henry's Gas Solubility Optimization (HGSO) [15], the Gravitational Search Algorithm (GSA) [16], the Small World Optimization Algorithm (SWOA) [17], etc. Also, a series of evolutionary based techniques have also been suggested for Global Optimization problems, such as Genetic Algorithms [18], the Differential Evolution method [19,20], Particle Swarm Optimization (PSO) [21,22], Ant Colony Optimization (ACO) [23], Bat Algorithm (BA) [24], Whale Optimization Algorithm (WOA) [25], Grasshopper Optimization Algorithm (GOA) [26], etc.

However, the above optimization methods require significant computational power and time. Therefore, parallel processing of these methods is essential. Recently, various methods have been proposed to leverage parallel processing. Specifically, one study details the parallel implementation of the Particle Swarm Optimization (PSO) algorithm, which improves performance for load-balanced problems but shows reduced efficiency for load-imbalanced ones. The study suggests using larger particle populations and implementing asynchronous processing to enhance performance [28]. Another review provides a comprehensive overview of derivative-free optimization methods, useful for problems where objective and constraint functions come only from a black-box or simulator interface. The review focuses on recent developments and categorizes methods based on assumptions about functions and their characteristics [29]. Lastly, the PDoublePop software is presented, which implements parallel genetic algorithms with advanced features, such as an enhanced stopping rule and advanced mutation schemes. The software allows for coding the objective function in C++ or Fortran77 and has been tested on well-known benchmark functions, showing promising results [30]. Moreover, there are methods that utilize GPU architectures for solving complex optimization problems. In the case of the Traveling Salesman Problem (TSP), a parallel GPU implementation has been developed to achieve high performance on large scales, solving problems up to 6,000 cities with great efficiency [31]. Additionally, the use of the multi-start model in local search algorithms has proven to be particularly effective when combined with GPUs, reducing computation time and improving solution quality for large and time-intensive problems [32]. Finally, a parallel algorithm using Peano curves for dimension reduction has demonstrated significant improvements in speed and efficiency when run on GPUs, compared to CPU-only implementations, for multidimensional problems with multiple local minima [33]. Parallel optimization represents a significant approach in the field of optimization problem-solving

and is applied across a broad range of applications, including the optimization of machine learning model parameters. In the case of GraphLab, a platform is presented that enhances existing abstract methods, such as MapReduce, for parallel execution of machine learning algorithms with high performance and accuracy [34]. Additionally, the new MARSAOP method for hyperparameter optimization uses multidimensional spirals as surrogates and dynamic coordinate search to achieve high-quality solutions with limited computational cost [35]. Lastly, the study proposes a processing time estimation system using machine learning models to adapt to complex distributions and improve production scheduling, achieving significant reduction in completion time [36].

The use of parallel optimization has brought significant advancements in solving complex control and design problems [37]. In one case, the parallel application of the Genetic Algorithm with a Fair Competitive Strategy (HFCGA) has been used for the design of an optimized cascade controller for ball and beam systems. This approach allows for the automation of controller parameter tuning, improving performance compared to traditional methods, with parallel computations helping to avoid premature convergence to suboptimal solutions [38]. In another case, the advanced Teaching-Learning-Based Optimization (TLBO) method has been applied to optimize the design of propulsion control systems with hydrogen peroxide. This method, which includes improvements in the teaching and learning phases as well as in the search process, has proven highly effective in addressing uncertainties in real-world design problems, enhancing both accuracy and convergence speed of the optimization process [39].

Similarly, parallel optimization has significant applications in energy and resource management. In the case of pump and cooling system management, a two-stage method based on an innovative multidimensional optimization algorithm has been proposed, which reduces overall energy consumption and discrepancies between pump and cooling system flow rates, achieving significant improvements in energy efficiency [40]. In energy network management, a coordinated scheduled optimization method has been proposed, which simultaneously considers various energy sources and uncertainty conditions. This method, using the Competitive Swarm Optimization (CSO) algorithm and adjusted based on chaos theory, has proven 50% faster than other methods and effective in managing uncertainties and energy [41]. In topological optimization, parallel processing through CPUs and GPUs has drastically improved speed and energy consumption for complex problems, achieving up to 25 times faster processing and reducing energy consumption by up to 93% [42]. Lastly, in building energy retrofitting, a multi-objective platform was used for evaluating and optimizing upgrade strategies, demonstrating that removing subsidies and providing incentives for energy upgrades yield promising results [43].

In conclusion, parallel optimization has provided significant solutions to problems related to sustainable development and enhancing sustainability. In the field of increasing the resilience of energy systems, the development of strategies for managing severe risks, such as extreme weather conditions and supplier disruptions, has been improved through parallel optimization, enabling safer and more cost-effective energy system operations [44]. In addressing groundwater contamination, the use of parallel optimization has highlighted the potential for faster and more effective design of remediation methods, significantly reducing computational budgets and ensuring better results compared to other methods [45]. Finally, in the field of sustainable production, metaheuristic optimization algorithms have proven useful for improving production scheduling, promoting resource efficiency, and reducing environmental impacts, while contributing to the achievement of sustainable development goals [46].

By harnessing multiple computational resources, parallel optimization allows for the simultaneous execution of multiple algorithms, leading to faster convergence and improved performance. These computational resources can communicate with each other to exchange information and synchronize processes, thereby contributing to faster convergence towards common solutions. Additionally, leveraging multiple resources enables more effective handling of exceptions and errors, while increased computational power for conducting

more trials or utilizing more complex models leads to enhanced performance [47]. Of course, this process requires the development of suitable algorithms and techniques to effectively manage and exploit available resources. Each parallel optimization algorithm requires a coherent strategy for workload distribution among the available resources, as well as an efficient method for collecting and evaluating results.

Various researchers have developed parallel techniques, such as the development and evaluation of five different parallel Simulated Annealing (SA) algorithms for solving global optimization problems. The paper compares the performance of these algorithms across an extensive set of tests, focusing on various synchronization and information exchange approaches, and highlights their particularly noteworthy performance on functions with large search spaces where other methods have failed [48]. Additionally, other research focuses on the parallel application of the Particle Swarm Optimization (PSO) method, analyzing the performance of the parallel PSO algorithm on two categories of problems: analytical problems with low computational cost and industrial problems with high computational cost [49]. This research demonstrates that the parallel PSO method performs exceptionally well on problems with evenly distributed load, while in cases of uneven load, the use of asynchronous approaches and larger particle populations proves more effective. Another study develops a parallel implementation of a stochastic Radial Basis Function (RBF) algorithm for global optimization, which does not require derivatives and is suitable for computationally intensive functions. The paper compares the performance of the proposed method with other parallel optimization methods, noting that the RBF method achieves good results with one, four, or eight processors across various optimization problems [50].Parallel techniques are also applied in real-time image processing, which is critical for applications such as video surveillance, diagnostic medicine, and autonomous vehicles. A related article examines parallel architectures and algorithms, identifying effective applications, evaluating challenges, and limitations of their practical implementation, aiming to develop more efficient solutions [53]. Another study reviews parallel computing strategies for computational fluid dynamics (CFD),focusing on tools like OpenMP, MPI, and CUDA to reduce computational time [54]. Finally, a significant study explores parallel programming technologies for processing genetic sequences, presenting three main parallel computing models and analyzing applications such as sequence alignment, single nucleotide polymorphism calling, sequence preprocessing, and pattern detection [55].

Genetic Algorithms (GAs) are methods that can be easily parallelized, and several researchers have thoroughly examined them in the literature. For example, a universal parallel execution method for Genetic Algorithms, known as IIP (Parallel, Independent, and Identical Processing), is presented. This method achieves acceleration with the use of m processors. The technique calculates execution speed and compares results on small-size problems and the non-parametric Inverse Fractal Problem [51]. Similarly, another paper presents a distributed mechanism for improving resource protection in a digital ecosystem. This mechanism can be used not only for secure and reliable transactions but also to enhance collaboration among digital ecosystem community members to secure the environment, and also employs Public Key Infrastructure to provide strong protection for access workflows [52]. Genetic Algorithms have also been used as a strategy for optimizing large and complex problems, specifically in wavelength selection for multi-component analysis. The study examines the effectiveness of the genetic algorithm in finding acceptable solutions in a reasonable time and notes that the algorithm incorporates prior information to improve performance, based on mathematically grounded frameworks such as the schema theorem [56]. Additionally, another paper develops two genetic algorithm-based algorithms to improve the lifetime and energy consumption in mobile wireless sensor networks (MWSNs). The first algorithm is an improvement of the Unequal Clustering Genetic Algorithm, while the second algorithm combines the K-means Clustering Algorithm with Genetic Algorithms. These algorithms aim to better adapt to dynamic changes in network topology, thereby enhancing network lifetime and energy consumption [57]. Lastly, the use of Genetic Algorithms has promising applications across various medical specialties,

including radiology, radiotherapy, oncology, pediatrics, cardiology, endocrinology, surgery, obstetrics and gynecology, pulmonology, infectious diseases, orthopedics, rehabilitation medicine, neurology, pharmacotherapy, and health care management. A related paper reviews and introduces applications of Genetic Algorithms in disease screening, diagnosis, treatment planning, pharmacovigilance, prognosis, and health care management, enabling physicians to envision potential applications of this metaheuristic method in their medical careers [58].

In this paper, a new optimization method is proposed which is a mixture of existing global optimization techniques running in parallel on a number of available computing units. Each technique is executed independently of the others and periodically the optimal values retrieved from them are distributed to the rest of the computing units using the propagation techniques presented here. In addition, for the most efficient termination of the overall algorithm, intelligent termination techniques based on stochastic observations are used, which are suitably modified to adapt to the parallel computing environment. The contributions of this work as summarized as follows:

1. Periodic local search was integrated into the Differential Evolution (DE) and Particle Swarm Optimization (PSO) methods.
2. A mechanism for disseminating the optimal solution to all PUs was added in each iteration of the methods.
3. The overall algorithm terminates based on the proposed termination criterion.

The following sections are organized as follows: In section 2, the three main algorithms participating in the overall algorithm are described. In section 3 the algorithm for parallelizing the three methods is described, along with the proposed mechanism for propagating the optimal solution to the remaining methods. In section 4, experimental models and experimental results are described. Finally, in section 5, the conclusions from the application of the current work are discussed.

2. Adopted Algorithms

The methods that may executed in each processing unit are fully described in this section.

2.1. The method PSO

PSO is an optimization method inspired by the behavior of swarms in nature. In PSO, a set of particles moves in the search space seeking the optimal solution. Each particle has a current position and velocity, and it moves based on its past performance and that of its neighboring particles. PSO continuously adjusts the movement of particles aiming for convergence to the optimal solution [59–61]. This method can be programmed easily and the set of the parameters to be set is limited. Hence, it has been used in a series of practical problems, such as problems that arise in physics [62,63], chemistry [64,65], medicine [66,67], economics [68] etc. The main steps of the PSO method are presented in the Algorithm 1.

Algorithm 1 The main steps of the PSO method.

1. **Initialization**.
   (a) **Set** $k = 0$ (iteration counter).
   (b) **Set** the number of particles N_p.
   (c) **Set** the maximum number of iterations allowed $k_{\max}$
   (d) **Set** the local search rate $p_l \in [0, 1]$.
   (e) **Initialize** randomly the positions of the m particles $x_1, x_2, ..., x_m$, with $x_i \in S \subset R^n$
   (f) **Initialize** randomly the velocities of the m particles $u_1, u_2, ..., u_m$, with $u_i \in S \subset R^n$
   (g) **For** $i = 1..N_p$ do $p_i = x_i$. The p_i vector are the best located values for every particle i.
   (h) **Set** $p_{\text{best}} = \arg\min_{i \in 1..m} f(x_i)$
2. **Termination Check**. Check for termination. If termination criteria are met then stop.
3. **For** $i = 1..N_p$ **Do**
   (a) **Update** the velocity:

$$u_i = \omega u_i + r_1 c_1 (p_i - x_i) + r_2 c_2 \left(p_{\text{best}} - x_i\right)$$

   The parameters $r_1,\ r_2$ are random numbers with $r_1 \in [0, 1]$ and $r_2 \in [0, 1]$.
   The constant number $c_1,\ c_2$ are in the range $[1, 2]$.
   The variable ω is called inertia, with $\omega \in [0, 1]$.
   (b) **Update** the position

$$x_i = x_i + u_i$$

   (c) **Set** $r \in [0, 1]$ a random number. If $r \leq p_m$ then $x_i = \text{LS}(x_i)$, where $\text{LS}(x)$ is a local search procedure.
   (d) **Evaluate** the fitness of the particle i, $f(x_i)$
   (e) **If** $f(x_i) \leq f(p_i)$ then $p_i = x_i$
4. **End For**
5. **Set** $p_{\text{best}} = \arg\min_{i \in 1..m} f(x_i)$
6. **Set** $k = k + 1$.
7. **Goto** Step 2

2.2. The method Differential evolution

The DE method relies on differential operators and is particularly effective in optimization problems that involve searching through a continuous search space. By employing differential operators, DE generates new solutions that are then evaluated and adjusted until the optimal solution is achieved [69,70]. The method was used in a series of problems, such as electromagnetics [71], energy consumption problems [72], job shop scheduling [73], image segmentation [74] etc. The Differential Evolution operates through the following steps: Initially, initialization takes place with a random population of solutions in the search space. Then, each solution is evaluated based on the objective function. Subsequently, a process is iterated involving the generation of new solutions through modifications, evaluation of these new solutions, and selection of the best ones for the next generation. The algorithm terminates when a termination criterion is met, such as achieving sufficient improvement in performance or exhausting the number of iterations. The main steps of the DE method are presented in Algorithm 2.

Algorithm 2 The main steps of the DE method.

1. **INPUT:**
   (a) The population size $N_d \geq 4$. The members of this population are also called agents.
   (b) The crossover probability CR $\in [0,1]$.
   (c) The differential weight $F \in [0,2]$.
2. **OUTPUT**:
   (a) The agent x_{best} with the lowest function value $f(x_{\text{best}})$.
3. **Initialize** all agents in S.
4. **While** termination criteria are not met **do**
   (a) **For** $i = 1 \ldots N_d$ **do**
      i. **Select** as x the agent i.
      ii. **Select** randomly three agents a, b, c with the property $a \neq b$, $b \neq c$, $c \neq a$.
      iii. **Select** a random position $R \in \{1, \ldots, n\}$
      iv. **Create** the vector $y = [y_1, y_2, \ldots, y_n]$ with the following procedure
      v. **For** $j = 1, \ldots, n$ **do**
         A. **Set** $r_i \in [0,1]$ a random number.
         B. **If** $r_j < \text{CR}$ **or** $j = R$ **then** $y_j = a_j + F \times (b_j - c_j)$ **else** $y_j = x_j$.
      vi. If $y \in S$ AND $f(y) \leq f(x)$ then $x = y$.
      vii. **EndFor**
   (b) **EndFor**
5. **End While**

2.3. The method Multistart

In contrast, the Multistart method follows a different approach than previous methods. Instead of focusing on a single initial point, it repeats the search from various initial points in the search space. This allows for a broader exploration of the search space and increases the chances of finding the optimal solution**.** The Multistart method is particularly useful when optimization algorithms may get trapped in local minima [75–77]. The approach of multiple starts belongs to the simplest techniques for global optimization. At the outset of the process, an initial distribution of points is made in Euclidean space. Subsequently, local optimization begins simultaneously from these points. The discovered minima are compared, and the best one is retained as the global minimum. Local optimizations rely on the Broyden Fletcher Goldfarb Shanno (BFGS) method [78]. The main steps of the multistart method are presented in Algorithm 3.

Algorithm 3 The main steps of the Multistart method.

1. **Initialization** step.
   (a) **Set** N_m as the total number of samples.
   (b) **Set** (x^*, y^*) as the global minimum. Initialize y^* to a very large value.
2. **Sampling** step.
   (a) **For** $i = 1 \ldots N_m$ **Do**
      i. **Sample** a point $x_i \in S$
      ii. $y_i = \text{LS}(x_i)$. Where LS(x) is a local search procedure.
      iii. **If** $y_i \leq y^*$ then $x^* = x_i, y^* = y_i$
   (b) **EndFor**

3. The proposed method

In the present work, a mixture of termination rules is proposed as a novel technique, which can be used without partitioning in any stochastic global optimization technique. Also, a new parallel global optimization technique is proposed that utilizes the new termination rule.

The existing method that was presented in section 2, employs Particle Swarm Optimization (PSO), Differential Evolution (DE), and Multistart techniques, each with its own independent termination criteria. In contrast, the proposed method that will be presented in this section, integrates an innovative combination of termination rules, including Best-Fitness, Mean-Fitness, and DoubleBox, enhancing the termination process of the search. By incorporating these advanced termination strategies into a parallel algorithm, the new method achieves faster convergence and improved solution quality. This approach combines the efficiency of parallel execution with the flexibility of advanced termination criteria, representing a significant advancement over traditional methods. In the following subsections, the new termination rule is fully described, followed by the new optimization method.

3.1. The new termination rule

The proposed termination rule is a mixture of several stochastic termination rules proposed in the recent literature. The first algorithm will be called *best - fitness* and in this termination technique, at each iteration k, the difference between the current best value $f_{min}^{(k)}$ and the previous best value $f_{min}^{(k+1)}$ is calculated [83], i.e., the absolute difference:

$$\left| f_{\min}^{(k)} - f_{\min}^{(k-1)} \right| \tag{2}$$

If this difference is zero for a series of predefined consecutive iterations N_k, then the method terminates. In the second termination rule, which will be called *mean - fitness*, the average function value for each iteration is computed. If this value remains relatively unchanged over a certain number of consecutive iterations, it indicates that the method may not be making significant progress towards discovering a new global minimum. Therefore, termination is warranted [20]. Hence, in every iteration k, we compute:

$$\delta^{(k)} = \left| \sum_{i=1}^{\mathrm{NP}} \left| f_i^{(k)} \right| - \sum_{i=1}^{\mathrm{NP}} \left| f_i^{(k-1)} \right| \right| \tag{3}$$

The value NP denotes the number of particles or chromosomes that participate in the algorithm. The termination rule is defined as follows: terminate if $\delta^{(k)} \leq \epsilon$ for a predefined number N_k of iterations. The third stopping rule was the so - called DoubleBox stopping rule, that was initially proposed in the work of Tsoulos [79]. In this criterion, the search process is terminated once sufficient coverage of the search space has been achieved. The estimation of the coverage is based on the asymptotic approximation of the relative proportion of points that lead to local minima. Since the exact coverage cannot be directly calculated, sampling is conducted over a larger region. The search is stopped when the variance of the sample distribution falls below a predefined threshold, which is adjusted based on the most recent discovery of a new local minimum. According to this criterion, the algorithm terminates when one of the following conditions is met:

- The iteration k count exceeds a predefined limit of iterations k_{max}
- The relative variance $\sigma^{(k)}$ falls below half of the variance $\sigma^{(k_{last})}$ of the last iteration $k_{(last)}$ where a new optimal functional value was found.

$$k \geq N_k \, or \, \sigma^{(k)} \leq \frac{\sigma^{(klast)}}{2} \tag{4}$$

This proposed termination criterion is the combination of the aforementioned similarity types. Optimization termination is achieved when any homogeneity condition is satisfied, resulting in improved speed. Hence by using equations 2, 3 and 4 following termination rule is derived:

$$\left| f_{\min}^{(k)} - f_{\min}^{(k-1)} \right| or$$

$$\delta^{(k)} = \left| \sum_{i=1}^{\mathrm{NP}} \left| f_i^{(k)} \right| - \sum_{i=1}^{\mathrm{NP}} \left| f_i^{(k-1)} \right| \right| or$$

$$\sigma^{(iteration)} \leq \frac{\sigma^{(klast)}}{2} or$$

$$k \geq N_k$$

3.2. The proposed algorithm

In the scientific literature, parallelization typically involves distributing the population among parallel computing units to save computational power and time [83,84]. In the present study, we concurrently compute the optimal solutions of three different stochastic optimization methods originating from different classification categories. The proposed algorithm is shown in Algorithm 4.

Algorithm 4 The proposed overall algorithm

1. **Set** as N_I the total number of parallel processing units.
2. **Set** as N_k the total number of allowed iterations.
3. **Set** $k = 0$ the iteration number.
4. **For** $j = 1, .., N_I$ do in parallel
   (a) **Execute** an iteration of any stochastic algorithm mentioned in section 2.
   (b) **Find** the best element from all optimization methods and **propagate** it to the rest of processing units.
5. **End For**
6. **Update** $k = k + 1$
7. **Check** the proposed termination rule. If the termination rule is valid, then goto step 7a else goto step 4.
   (a) **Terminate** and report the best value from all processing units.

The proposed algorithm is graphically outlined in Figure 1. The methods that are used in each processing unit may include Particle Swarm Optimization, the Multistart method and Differential Evolution (DE). These techniques have been adopted since they have been widely used in the relevant literature and provide the possibility to be parallelized relatively easily. Initially the algorithm distributes the three methodologies across N_I threads, where $N_I > 3$. The first thread executes the DE method, the next executes PSO, and the third runs the Multistart method. As shown in the Figure 1, at each iteration, every computational unit that uses this specific optimization method calculates the best solution and replaces the worst solutions of the other computational units. This process is repeated to ensure an equivalent distribution of the methods across all threads, optimizing the utilization of available computational resources. When the number of threads is equal to or a multiple of the number of methods, the distribution is 100% equivalent. In other cases, the degree of equivalence increases with the number of threads.

As presented in this subsection, the proposed algorithm exploits the parallel execution of three distinct optimization methods, with its efficiency in termination procedures incorporated through the criteria described in Subsection 3.1. The algorithm uses three termination criteria: the difference between the best values in successive iterations, the

stability of the mean fitness value, and the relative variance of the cost function values. Each criterion contributes to the evaluation of the algorithm's progress, dynamically switching between these criteria based on the current state of the search. Specifically, when any of the criteria meets its predefined conditions, the algorithm terminates the search, thus adapting the process to the continuous search conditions. This strategy ensures the flexibility and efficiency of the algorithm, minimizing computational resources and time wastage.

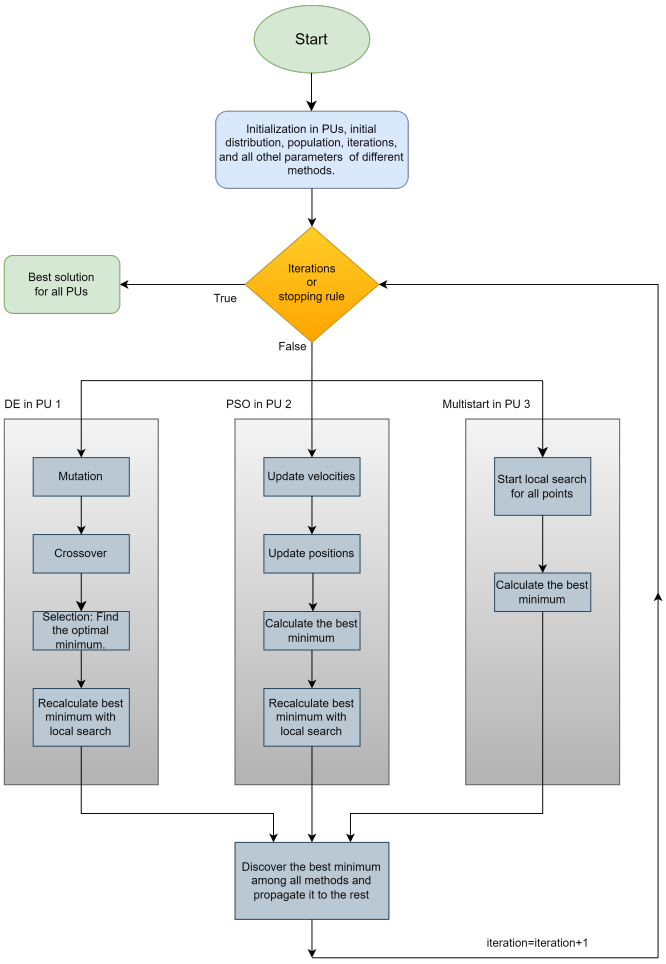

Figure 1. Flowchart of the overall process. The DE acronym stands for the Differential Evolution method, the PU acronym represents the Parallel Unit that executes the algorithm.

4. Experiments

All experiments conducted were repeated 30 times to ensure the reliability of the algorithm producing the results. In experimental research, the number of repetitions required to ensure reliable results depends on the nature of the experiment, the variability of the data, and the precision requirements. There is no universal answer that applies to all experiments, but general guidelines and best practices have been established in the literature. In many scientific studies, at least 30 repetitions are recommended to achieve satisfactory statistical power. This is based on statistical theories such as the Law of Large Numbers and theorems for estimating sample distributions. The initial sample distribution is the same across all optimization methods, and the other parameters have been kept constant, as shown in Table 1. Specifically, in the parallel algorithm, the total number of initial samples remains as constant as possible, regardless of the number of threads used. The parallelization was achieved using the OpenMP library [88], while the implementation of the method was done in ANSI C++ within the optimization package OPTIMUS, available at https://github.com/itsoulos/OPTIMUS. The experiments were executed on a Debian Linux system that runs on an AMD Ryzen 5950X processor, with 128GB of RAM. The

parameters used here are either suggested in the original publications of the methods (e.g. Differential Evolution, Particle Swarm Optimization) or have been chosen so that there is a compromise between the execution time of the method and its efficiency.

Table 1. The following parameters were considered for conducting the experiments

Parameter	Value	Explanation
N_p	120	Total particles for PSO
N_d	120	Total elements for DE
N_m	120	Total elements for Multistart
N_k	200	Maximum number of iterations
N_s	15	Similarity max count
F	0.8	Differential weight for DE
CR	0.9	Crossover Probability for DE
P_m	0.005	Local search rate for DE and PSO
C_1, C_2	0.5	Parameters of PSO

4.1. Test functions

The test functions [85,86] presented below exhibit varying levels of difficulty in solving them; hence, a periodic local optimization mechanism has been incorporated. Periodic local optimization plays a crucial role in increasing the success rate in locating the minimum of functions. This addition appears to lead to a success rate approaching 100% for all functions, regardless of their characteristics such as dimensionality, minima, scalability, and symmetry. A study by Z.-M. Gao and colleagues [87] specifically examines the issue of symmetry and asymmetry in the test functions. The test functions used in the conducted experiments are shown in Table 2. Generally, the region of attraction $A(x^*)$ of a local minimum x^* is defined as:

$$A(x^*) = \{x : \ x \in S, \ L(x) = x^*\} \tag{5}$$

where $L(x)$ is a local search procedure, such as BFGS. Global optimization methods, usually find points which are in the region of attraction $A(x^*)$ of a local minimum x^* but not necessarily the local minimum itself. For this reason, at the end of their execution, a local minimization method is applied in order to ensure the exact finding of a local minimum. Of course, this does not imply that the minimum that will be located will be the total minimum of the function, since a function can contain tens or even hundreds of local minima.

Table 2. The test functions used in the conducted experiments.

NAME	FORMULA	DIMENSION
Cigar10	$f(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$	10
BF1	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	2
BF2	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	2
Branin	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	2
CM	$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	4
Discus	$f(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$	10
Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2-\pi)^2 - (x_1-\pi)^2\right)$	2
Exp	$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \le x_i \le 1$	$n = 4, 16$
Griewank2	$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}$	2
Griewank10	$\mathrm{f}(x) = 1 + \frac{1}{200}\sum_{i=1}^{10} x_i^2 - \prod_{i=1}^{10}\frac{\cos(x_i)}{\sqrt{(i)}}$	10
Gkls[80]	$f(x) = \mathrm{Gkls}(x, n, w)$	$n = 2, 3\ w = 50, 100$
Hansen	$f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]$	2
Hartman3	$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	3
Hartman6	$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	6
Elp	$f(x) = \sum_{i=1}^{n}(10^6)^{\frac{i-1}{n-1}} x_i^2$	10
Potential[81]	$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$	$n = 9, 15, 21, 30$
Rastrigin	$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$	2
Rosenbrcok	$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \le x_i \le 30$	$n = 4, 8, 16$
Shekel5	$f(x) = -\sum_{i=1}^{5}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4
Shekel7	$f(x) = -\sum_{i=1}^{7}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4
Shekel10	$f(x) = -\sum_{i=1}^{10}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	4
Sinusoidal[82]	$f(x) = -(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))), \quad 0 \le x_i \le \pi$	$n = 4, 8$
Test2N	$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i$	$n = 4, 9$
Test30N	$\frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i-1)^2(1+\sin^2(3\pi x_{i+1}))\right) + (x_n - 1)^2(1 + \sin^2(2\pi x_n))$	$n = 3, 4$

4.2. Experimental results

All tables presented here demonstrate the average function calls obtained for each test problem. The average function can be considered as a measure of the speed of each method, since each function has a different complexity and therefore, the average execution time would not be a good indication of the computational time required. In the first set of experiments we examined if the presence of the suggested stopping rule, which is the combination of a series of stopping rules, affects the optimization techniques without parallelization. During parallel execution, the total sum of calls is computed across all computing units nodes, ensuring efficient load distribution and resource utilization. However, the algorithm terminates at the computing unit that identifies the optimal solution, signaling the achievement of the desired convergence criterion. In Table 3, we observe the performance of all termination rules for the DE method. From these, both statistical and quantitative comparisons arise, depicted in Figure 2.

Table 3. Average function calls for the Differential evolution method, using different termination rules

Problem	BEST	MEAN	DOUBLEBOX	ALL
BF1	6761	10511	10369	6114
BF2	7693	9222	11524	7560
BRANIN	4639	10982	4907	4289
CAMEL	6116	13290	9905	5940
CIGAR10	7647	4348	21116	3111
CM4	7913	4175	11851	4079
DISCUS10	5191	2042	13614	2034
EASOM	1917	15103	1721	1721
ELP10	6046	24675	12519	6031
EXP4	5216	18151	6385	5040
EXP16	5588	27387	7552	5339
GKLS250	5227	2641	8379	2641
GKLS350	5624	3298	17437	3206
GRIEWANK2	8027	10458	14756	6915
GRIEWANK10	9664	37839	17312	9539
POTENTIAL3	5278	24823	13871	5256
PONTENTIAL5	8225	32439	20828	7742
PONTENTIAL6	8467	34946	19169	8197
PONTENTIAL10	10330	41300	26308	10643
HANSEN	5219	23050	14245	4263
HARTMAN3	4613	17966	7333	4566
HARTMAN6	5734	17109	9354	5550
RASTRIGIN	5912	17240	8987	5999
ROSENBROCK8	8503	34429	20980	8051
ROSENBROCK16	10052	41335	26156	8307
SHEKEL5	6064	25800	19877	5933
SHEKEL7	5597	26168	20274	5677
SHEKEL10	6037	26439	22960	5100
SINU4	5751	24620	12511	4776
SINU8	6340	25477	18741	4980
TEST2N4	5848	24947	11640	5727
TEST2N9	7303	26571	16165	6288
TEST30N3	3169	8395	8134	2869
TEST30N4	2841	10381	7853	2714
Total	**214552**	**677557**	**474733**	**186197**

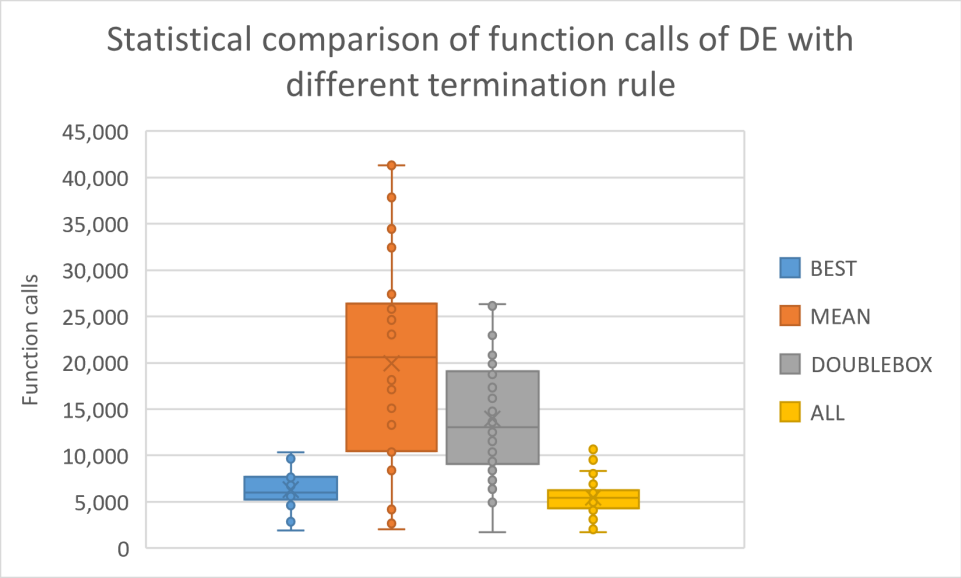

Figure 2. Statistical comparison of function calls with different termination rule of differential evolution

As it is evident, the combination of the stopping rules reduces the required number of function calls that are required to obtain the global minimum by the DE method. The proposed termination scheme outperforms by a percentage that varies from 13% to 73% the other termination rules used here. In addition, in order to ascertain which of the three termination rules has the greatest effect on the termination with the mixed rule, the rule responsible for the termination of the DE optimization technique was recorded for all functions of the test set. The results from this recording are shown in Table 4. The two techniques that seem to be the most decisive for terminating each function are Best and the Doublebox method, and in fact the Best method appears to significantly outperform DoubleBox termination scheme on average.

Table 4. Percentage of the rule which is responsible for termination in the mixed stopping rule.

Problem	Mean	Best	Doublebox
BF1	0.00%	36.67%	63.33%
BF2	6.67%	33.33%	60.00%
BRANIN	0.00%	6.67%	93.33%
CAMEL	10.00%	76.67%	13.33%
CIGAR10	0.00%	20.00%	80.00%
CM4	0.00%	90.00%	10.00%
DISCUS10	0.00%	6.67%	93.33%
EASOM	0.00%	0.00%	100.00%
ELP10	0.00%	0.00%	100.00%
EXP4	0.00%	86.67%	13.33%
EXP16	0.00%	76.67%	23.33%
GKLS250	6.67%	73.33%	20.00%
GKLS350	20.00%	80.00%	0.00%
GRIEWANK2	3.33%	26.67%	70.00%
GRIEWANK10	0.00%	10.00%	90.00%
POTENTIAL3	3.33%	66.67%	30.00%
PONTENTIAL5	10.00%	86.67%	3.33%
PONTENTIAL6	0.00%	90.00%	10.00%
PONTENTIAL10	13.33%	86.67%	0.00%
HANSEN	16.67%	50.00%	33.33%
HARTMAN3	0.00%	70.00%	30.00%
HARTMAN6	3.33%	76.67%	20.00%
RASTRIGIN	0.00%	83.33%	16.67%
ROSENBROCK8	0.00%	13.33%	86.67%
ROSENBROCK16	0.00%	13.33%	86.67%
SHEKEL5	3.33%	93.33%	3.33%
SHEKEL7	10.00%	90.00%	0.00%
SHEKEL10	6.67%	93.33%	0.00%
SINU4	6.67%	80.00%	13.33%
SINU8	0.00%	100.00%	0.00%
TEST2N4	0.00%	76.67%	23.33%
TEST2N9	0.00%	100.00%	0.00%
TEST30N3	0.00%	33.33%	66.67%
TEST30N4	0.00%	40.00%	60.00%
AVERAGE	**3.53%**	**57.84%**	**38.63%**

Subsequently, the same experiment was conducted for the PSO method and the results are shown in Table 5 and the statistical comparison is outlined graphically in Figure 3.

Table 5. Average function calls for the PSO method, with different termination rules

Problem	BEST	MEAN	DOUBLEBOX	ALL
BF1	3418	3127	3122	3005
BF2	3265	2995	2919	2889
BRANIN	2601	2549	2431	2417
CAMEL	2801	2675	2547	2547
CIGAR10	3987	3674	3638	3638
CM4	3863	3299	3144	3144
DISCUS10	2611	2409	2405	2405
EASOM	2420	2232	2232	2232
ELP10	1705	1741	1586	1586
EXP4	2751	2558	2558	2558
EXP16	2898	2676	2676	2676
GKLS250	2796	2553	2422	2422
GKLS350	2947	2645	3833	2558
GRIEWANK2	4405	2839	4282	2820
GRIEWANK10	5327	4561	4797	4248
POTENTIAL3	3289	3231	3222	3170
PONTENTIAL5	4926	4881	7282	4730
PONTENTIAL6	6699	5537	7534	5077
PONTENTIAL10	10582	6737	29865	6539
HANSEN	3788	2687	3062	2587
HARTMAN3	2807	2743	2550	2550
HARTMAN6	3081	2915	2809	2809
RASTRIGIN	3558	2990	2829	2829
ROSENBROCK8	4279	4263	4036	3969
ROSENBROCK16	5761	5432	5170	5170
SHEKEL5	3132	2823	12476	2816
SHEKEL7	3064	2855	11520	2856
SHEKEL10	3128	2942	14786	2866
SINU4	3047	2740	2673	2657
SINU8	3177	2824	4113	2828
TEST2N4	3037	2839	2681	2681
TEST2N9	4188	3106	3105	3067
TEST30N3	2943	3130	2762	2762
TEST30N4	3202	3185	2915	2915
Total	**125483**	**110393**	**169982**	**106023**

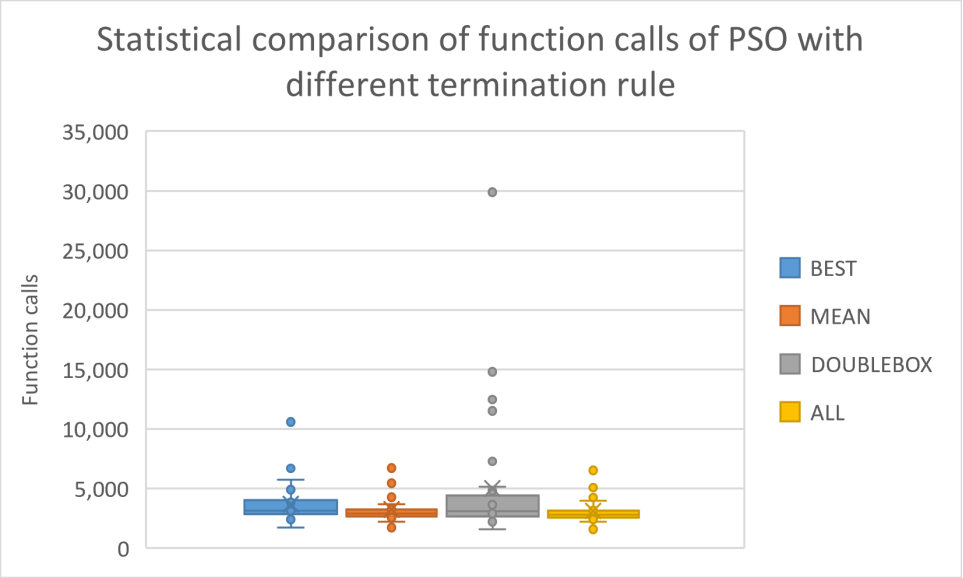

Figure 3. Statistical comparison of function calls with different termination rule of PSO method

Once more, the proposed termination scheme outperforms the other stopping rules in almost every test function. Likewise, the same experiment was carried out for the Multistart method and the results are depicted in Table 6 and the statistical comparison is shown in Figure 4.

Table 6. Multistart: Function calls with different termination rules

Problem	BEST	MEAN	DOUBLEBOX	ALL
BF1	53873	479374	50762	50762
BF2	37670	239220	35711	35711
BRANIN	11063	11046	10521	10521
CAMEL	16083	153559	15255	15255
CIGAR10	14937	14937	14697	14697
CM4	62236	62192	58581	58581
DISCUS10	6296	6296	6056	6056
EASOM	5412	55130	51132	5412
ELP10	14757	14981	14517	14517
EXP4	10022	54174	9774	9774
EXP16	10528	54680	10280	10280
GKLS250	6202	47948	5908	5908
GKLS350	7560	42145	7087	7087
GRIEWANK2	19179	188429	17877	17877
GRIEWANK10	65206	567153	61818	61818
POTENTIAL3	17773	126632	17161	17161
PONTENTIAL5	31347	226873	30249	30249
PONTENTIAL6	35457	247144	33883	33883
PONTENTIAL10	44957	283230	43618	43618
HANSEN	18568	201543	17541	17541
HARTMAN3	17395	162934	16562	16562
HARTMAN6	22010	179073	21015	21015
RASTRIGIN	24696	275610	23015	23015
ROSENBROCK8	17850	17850	17610	17610
ROSENBROCK16	25300	25300	25060	25060
SHEKEL5	17725	151648	16972	16972
SHEKEL7	17894	154373	17127	17127
SHEKEL10	17931	159825	17135	17135
SINU4	16429	155634	15647	15647
SINU8	19519	170167	18674	18674
TEST2N4	16486	154092	15806	15806
TEST2N9	22035	160651	20716	20716
TEST30N3	16934	161628	16145	16145
TEST30N4	16686	159753	15907	15907
Total	**758016**	**5165224**	**769819**	**724099**

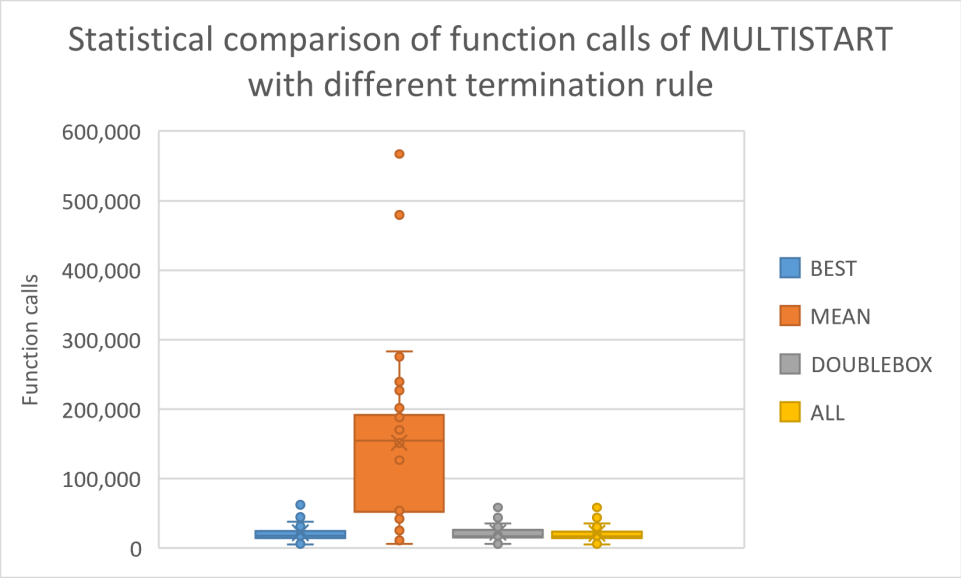

Figure 4. Statistical comparison of function calls with different termination rule of multistart

Observing the tables with the corresponding statistical and quantitative comparisons, it is evident that the calls to the objective function for the proposed termination rule are fewer than any other rule in any method. Also, the proposed termination rule was also applied to a recent global optimization method named Grey Wolf Optimizer [89] and the results are shown in Table 7 and the corresponding statistical comparison in Figure 5.

Table 7. Experiments using the variety of termination rules and the GWO optimization method.

Problem	BEST	MEAN	DOUBLEBOX	ALL
BF1	3080	4675	6719	3080
BF2	3083	4747	6923	3083
BRANIN	3217	3217	5105	2661
CAMEL	2750	3411	3780	2724
CIGAR10	14718	23003	24134	14718
CM4	3187	5638	8802	3187
DISCUS10	12946	22194	24134	12946
EASOM	2701	2965	10201	2135
ELP10	6440	22449	24123	6440
EXP4	2796	5198	7830	2796
EXP16	4991	10842	22025	4991
GKLS250	3072	3072	4840	2940
GKLS350	4041	4041	8620	4041
GRIEWANK2	3155	4818	7258	3155
GRIEWANK10	8568	10196	22841	8568
POTENTIAL3	3766	3766	13063	3701
PONTENTIAL5	4082	4082	19826	3645
PONTENTIAL6	3754	3754	19477	3380
PONTENTIAL10	3502	3502	13129	2538
HANSEN	3407	3407	8345	2795
HARTMAN3	3251	3251	7587	2911
HARTMAN6	3639	3639	13342	3408
RASTRIGIN	2878	4499	6463	2806
ROSENBROCK8	5859	22384	24123	5859
ROSENBROCK16	8528	23513	24132	8528
SHEKEL5	3381	3381	16781	2778
SHEKEL7	3600	3600	17989	3324
SHEKEL10	3258	3258	19227	2866
SINU4	2923	2923	9767	2231
SINU8	2934	2934	10714	2502
TEST2N4	3623	3623	9619	3603
TEST2N9	4327	4327	21381	4207
TEST30N3	3225	3237	3513	2769
TEST30N4	3324	3324	6124	3144
Total	**152006**	**236870**	**451937**	**144460**

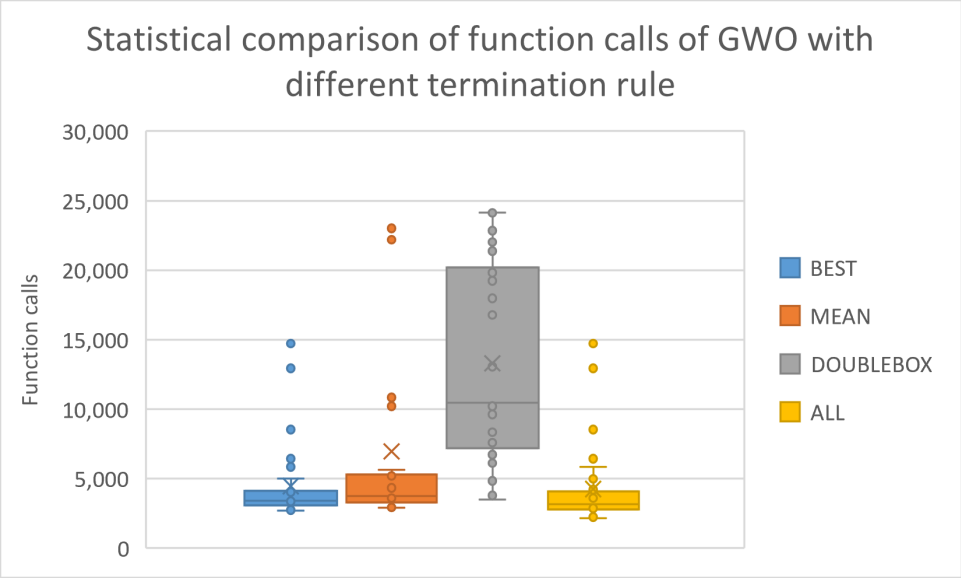

Figure 5. Statistical comparison for the GWO optimizer using different termination rules.

Once again the combinatorial termination technique achieves the lowest number of function calls of all available techniques. Each optimization method and termination criterion function differently and yield varying results. This variation is further influenced by the differing nature of the cost functions involved. When integrated into the overall algorithm, each method, in combination with the termination criterion that best suits it, can converge to optimal solutions more quickly, thereby enhancing the algorithm's overall performance.

The proposed termination rule was also applied to the current parallel optimization technique for the test functions described previously. The experimental results for the so - called *DoubleBox* stopping rule and the given parallel method are shown in Table 8. The columns in the table stand for the following:

1. Column Problem denotes the test function used.
2. Column 1x500 denotes the application of the proposed technique with one processing unit and 500 particles.
3. Column 2x250 stands for the usage of two processing units. In each unit, the number of particles was set to 250.
4. Column 5x100 denotes the incorporation of 5 processing units into the proposed method. In each processing unit the number of particles was set to 100.
5. Column 10x50 stands for the usage of 10 processing units. In each processing unit, the number of particles was set to 50.

Table 8. Experiments using the proposed optimization technique and the Doublebox stopping rule. The experiment was performed on the test functions described previously. Numbers in cells denote average function calls.

Problem	1x500	2x250	5x100	10x50
BF1	36018	28282	18419	10859
BF2	42191	37555	16568	10710
BRANIN	45165	45726	45757	34381
CAMEL	54786	54782	54253	53839
CIGAR10	18387	18300	18307	11950
CM4	60796	60501	60287	60200
DISCUS10	25115	24586	14565	8957
EASOM	40376	40211	40100	40099
ELP10	20193	14745	11211	7691
EXP4	54779	54676	54416	54412
EXP16	52768	52205	52563	52560
GKLS250	51980	51240	51233	51231
GKLS350	48577	48761	48702	4100
GRIEWANK2	56767	56690	51170	26076
GRIEWANK10	20941	20721	14870	11765
POTENTIAL3	49418	49800	49109	49002
PONTENTIAL5	59827	59656	59367	59322
PONTENTIAL6	62720	62398	62294	62280
PONTENTIAL10	72899	72745	72943	72578
HANSEN	45064	45689	45510	45424
HARTMAN3	47307	47112	47111	47100
HARTMAN6	49370	49302	49312	49222
RASTRIGIN	57617	56577	56513	56400
ROSENBROCK8	22997	22903	14139	14001
POSENBROCK16	42117	34856	21392	13722
SHEKEL5	50631	50198	50233	44662
SHEKEL7	51212	51200	51200	45610
SHEKEL10	51741	51607	51587	47761
SINU4	48140	48100	48111	48099
SINU8	48761	48434	48423	48204
TEST2N4	48904	48307	48354	48354
TEST2N9	48838	48555	48600	48335
TEST30N3	51640	51307	51194	51143
TEST30N4	48992	48549	48540	48487
Total	**1587034**	**1556276**	**1476353**	**1338536**

In this experiment, the total number of particles was set to 500, in order to have credibility in the values recorded. The results indicate that the increase in processing units may reduce the required number of function calls. The same series of experiments was also conducted using the so - called *mean fitness* termination rule, that has been described in equation 3 and 2. The obtained results are presented in Table 9 and 10.

Table 9. Experiments using the proposed optimization technique and the mean - fitness stopping rule. Numbers in cells denote average function calls.

Problems	1x500	2x250	5x100	10x50
BF1	49714	41284	41089	19596
BF2	58272	46270	41324	15878
BRANIN	45165	45665	44894	7268
CAMEL	54786	54989	54651	10295
CIGAR10	63026	62206	60277	27217
CM4	60796	60333	40072	12977
DISCUS10	50988	50690	36151	14040
EASOM	40376	34178	21355	7377
ELP10	50397	16787	9334	7809
EXP4	54749	54445	29812	11024
EXP16	52768	51233	24782	10722
GKLS250	51980	50333	37696	9641
GKLS350	48577	48442	28420	9337
GRIEWANK2	56767	40471	30632	12356
GRIEWANK10	68439	58410	53818	30317
POTENTIAL3	49418	49328	33657	10140
PONTENTIAL5	59827	58259	34436	11515
PONTENTIAL6	62720	62715	34158	14417
PONTENTIAL10	72899	72800	64964	16780
HANSEN	45064	45582	32013	9318
HARTMAN3	47307	47983	31033	8899
HARTMAN6	49370	47983	27884	9602
RASTRIGIN	54637	53551	33095	10169
ROSENBROCK8	67885	57314	46961	25383
POSENBROCK16	77207	66172	59098	41461
SHEKEL5	50631	49800	32025	10389
SHEKEL7	51212	50630	29136	11193
SHEKEL10	51741	50321	33462	11200
SINU4	48140	52788	32412	9721
SINU8	48761	55864	32685	10410
TEST2N4	48904	48800	28959	8643
TEST2N9	48838	47001	36281	8085
TEST30N3	51514	50100	39828	11699
TEST30N4	48992	46875	42564	11954
Total	**1841867**	**1729602**	**1258958**	**456832**

And in this series of experiments, the columns of the table retain the same meaning as in Table 8. Furthermore, once again, it is observed that the increase in the number of computing units significantly reduces the required number of function calls to find the global minimum. Additionally, there is a significant reduction in the number of function calls required to be compared to the previous termination rule. Finally, the proposed termination rule is utilized in the parallel optimization technique and the experimental results are outlined in Table 11.

Table 10. Experiments using the proposed optimization technique and the best - fitness stopping rule. Numbers in cells denote average function calls.

Problems	1x500	2x250	5x100	10x50
BF1	9270	6253	6497	6089
BF2	9838	6249	6078	5853
BRANIN	5787	5022	4991	4989
CAMEL	12204	5882	5765	5399
CIGAR10	7631	6482	6350	6420
CM4	16222	5037	7422	6366
DISCUS10	5997	7364	4906	5037
EASOM	4701	4752	4653	4668
ELP10	6230	5223	5407	5314
EXP4	10277	5675	5638	5526
EXP16	9577	5575	5471	5470
GKLS250	13328	5864	5323	4993
GKLS350	10131	4998	5442	4919
GRIEWANK2	8504	7345	6108	5876
GRIEWANK10	8535	7704	7771	7126
POTENTIAL3	11338	5726	5471	5699
PONTENTIAL5	12414	7192	6977	7070
PONTENTIAL6	14760	8480	7210	7511
PONTENTIAL10	17729	12218	10217	9439
HANSEN	8379	5945	5023	4972
HARTMAN3	10674	5613	5703	5163
HARTMAN6	10841	5541	5485	5536
RASTRIGIN	13657	6636	6129	5399
ROSENBROCK8	9456	6979	7165	7049
POSENBROCK16	9853	8620	8110	8379
SHEKEL5	10959	5437	5684	5616
SHEKEL7	10651	5715	5739	5486
SHEKEL10	1533	6064	5772	5516
SINU4	10291	6249	5426	5188
SINU8	9993	6847	6080	5582
TEST2N4	10242	6131	5837	5480
TEST2N9	12998	7402	6455	5702
TEST30N3	6228	5712	5723	5321
TEST30N4	5540	5437	5490	5484
Total	335768	217369	207518	199637

Table 11. Experiments using the proposed optimization technique and proposed stopping rule. Numbers in cells denote average function calls.

Problems	1x500	2x250	5x100	10x50
BF1	10839	6543	6375	5933
BF2	9838	6317	5978	5716
BRANIN	5787	5003	5076	4936
CAMEL	12204	5889	5703	5434
CIGAR10	7631	6575	6582	6500
CM4	16222	8935	6837	6392
DISCUS10	5997	5039	5001	5252
EASOM	4701	4791	4772	4751
ELP10	6230	5363	5471	5205
EXP4	10277	5542	5620	5533
EXP16	9577	5518	5544	5532
GKLS250	13328	5993	5205	5011
GKLS350	10131	7373	5094	4869
GRIEWANK2	8504	8284	6371	5761
GRIEWANK10	8535	7585	7229	6922
POTENTIAL3	11338	5666	5830	5695
PONTENTIAL5	12414	7310	7126	6996
PONTENTIAL6	14760	8450	7477	7561
PONTENTIAL10	17729	11928	10431	9338
HANSEN	8379	6862	4883	4970
HARTMAN3	10674	5395	5219	5078
HARTMAN6	10841	5486	5649	5348
RASTRIGIN	13657	8283	5850	5292
ROSENBROCK8	9456	6751	7232	6979
POSENBROCK16	9853	7993	8377	8086
SHEKEL5	10959	5749	5810	5884
SHEKEL7	10651	5828	5843	5698
SHEKEL10	1533	5863	6112	5588
SINU4	10291	6427	5371	5251
SINU8	9993	7139	6274	5583
TEST2N4	10242	7516	5573	5129
TEST2N9	12998	10137	6112	5566
TEST30N3	6228	5512	5402	5485
TEST30N4	5540	5749	5750	5350
Total	**337337**	**228794**	**207179**	**198624**

And in this case, it is observed that the increase of parallel processing units significantly reduces the required number of function calls, as in the two previous termination rules. However, the proposed termination rule dramatically reduces the number of required function calls and consequently the computation time compared to previous termination rules. This reduction in fact intensifies as the number of computing nodes increases. Also, for the same set of experiments, a statistical test (Wilcoxon test) was conducted and the it is graphically presented in Figure 6.

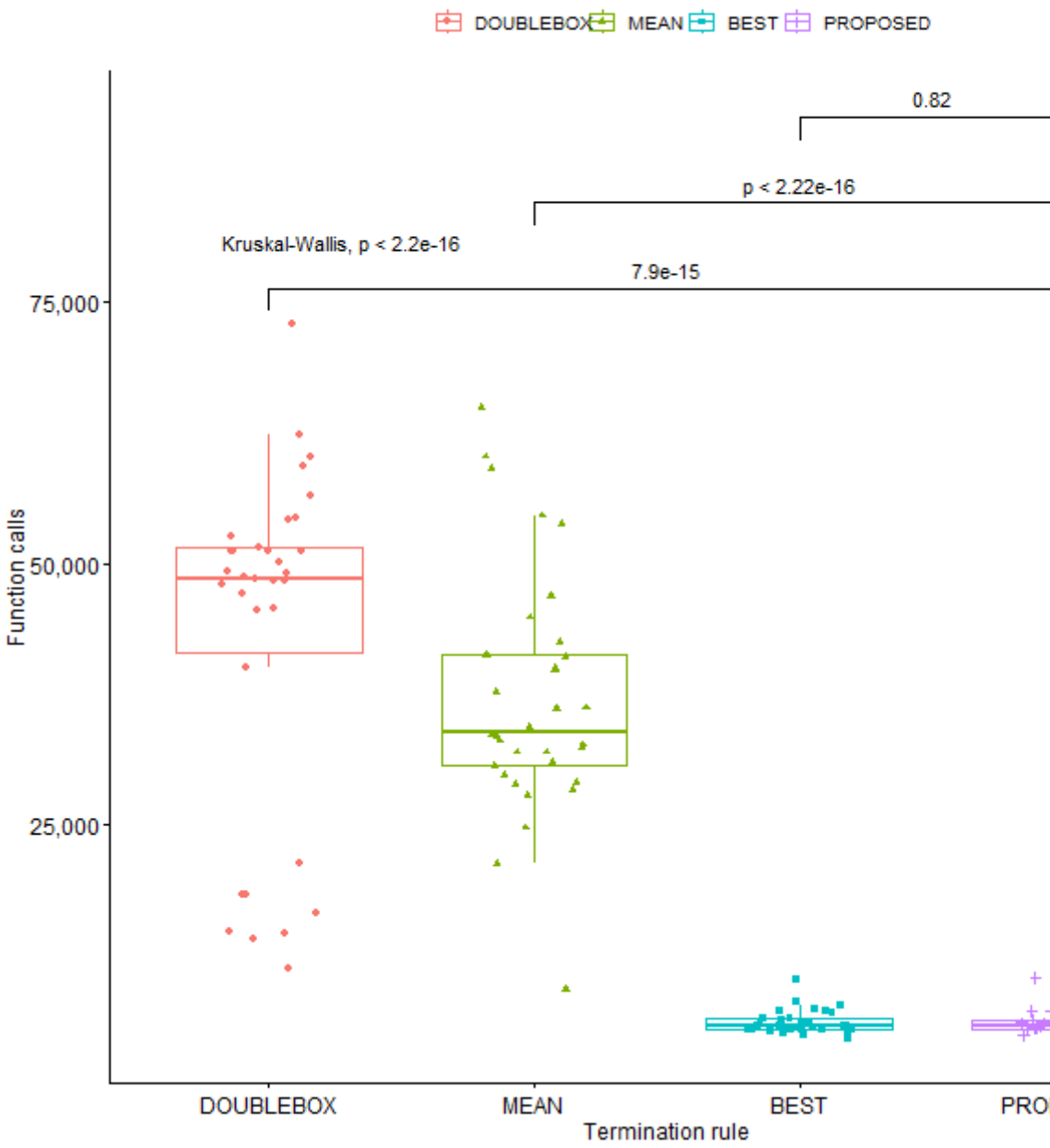

Figure 6. Wilcoxon rank-sum test results for the comparison of different termination rules as used in the proposed parallel optimization technique and for 5 processing units.

A p-value of less than 0.05 (two-tailed) was used to determine statistical significance and is indicated in bold.

An additional experiment was performed to assess the capabilities of the parallel technique using the CM function for various values of the dimension. This function is defined as:

$$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10} \sum_{i=1}^{n} \cos(5\pi x_i)$$

The proposed method was tested on this function for different values of the dimension n and for a variety of processing units and the results are graphically illustrated in Figure 7.

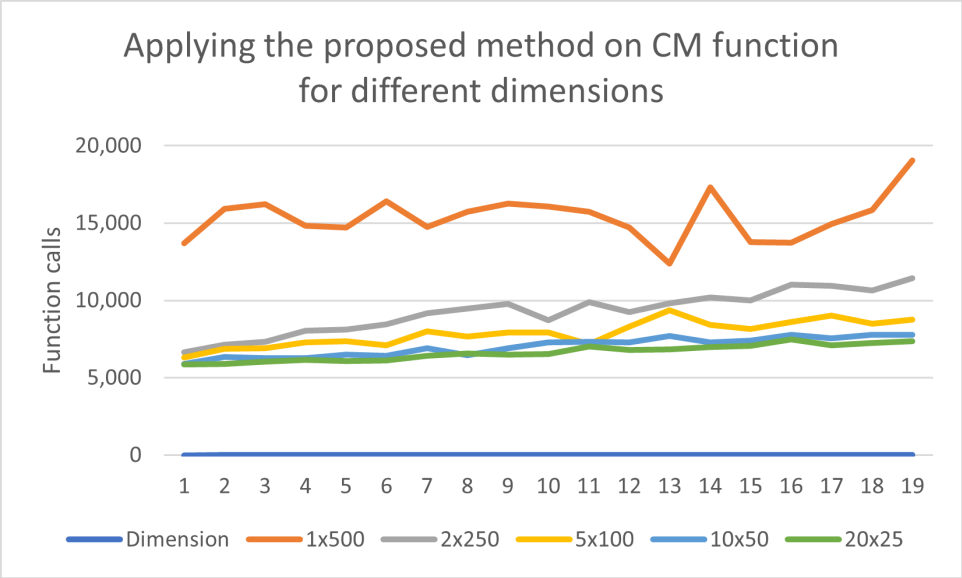

Figure 7. Experiments using the proposed paralell technique and the CM function. The dimension of the function n varies from 1 to 20 and the number of processing units changes from 1 to 20.

From the experimental results it is obvious that the required number of function calls is drastically reduced with the increase of computing units from 1 to 2 or to 5.

Also, the proposed method was applied on the GKLS function for different versions of the function. The dimension of the function was in the range [2..5]. In each dimension, 10 different versions of the functions were used and average execution times and function calls were recorded. Also, the proposed method was applied on the GKLS function using different number of processing units. The average function calls from this experiment are show graphically in Figure 8 and the average execution times in Figure 9.

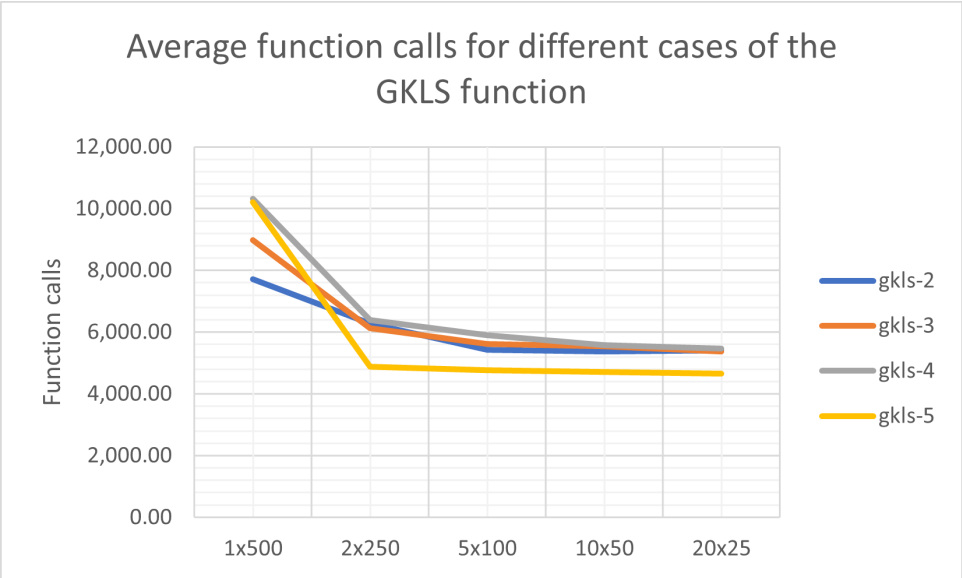

Figure 8. Average function calls for the proposed method and different dimensions of the GKLS function.

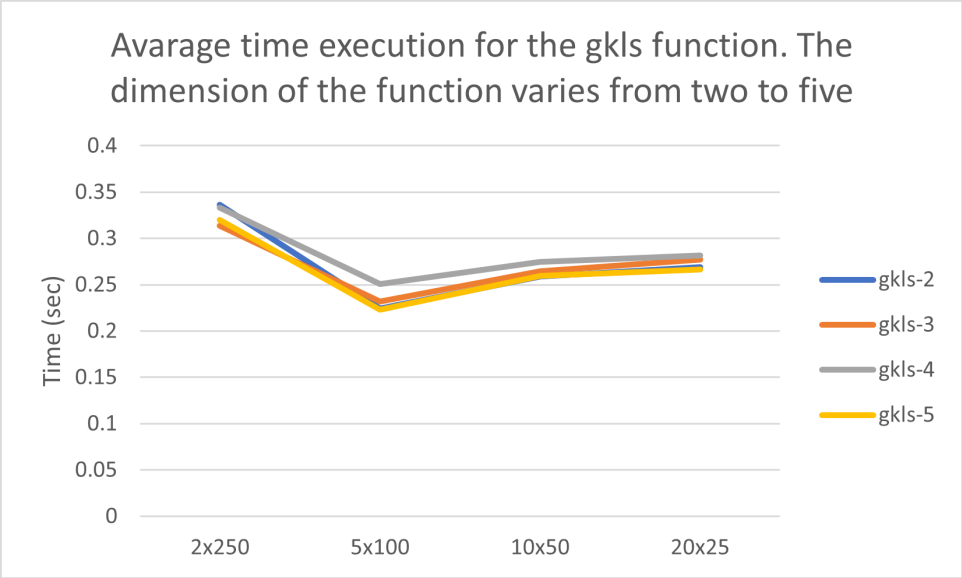

Figure 9. Average execution time for the GKLS function and the proposed method.

The graphs above show once again that adding processing units to the proposed technique can reduce not only the execution time, as expected, but also the required number of function calls. In the specific scheme, the execution time increases slightly for a certain number of execution units and above, as this time also includes the time spent by the execution engine for the synchronization of the processing units as well as for the exchange of information between them.

Furthermore, in Figure 10, the behavior of speedup and efficiency is depicted as a function of the number of PUs. The execution of the algorithm with a single processing unit, where the speedup is 1, serves as the reference point. As the number of PUs increases, a corresponding increase in speedup is observed, though it is often less than the ideal (equal to the number of PUs) due to communication delays and overhead. Conversely, efficiency decreases with the increase in PUs, a common phenomenon in parallel computing, caused by non-linear scaling and uneven workload distribution.

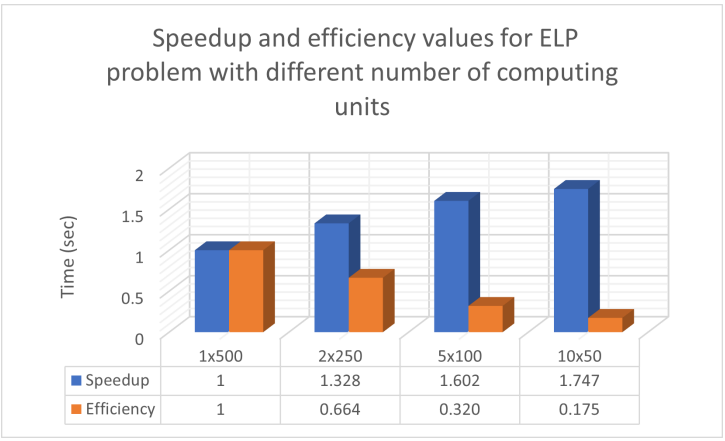

Figure 10. Speedup and efficiency of the parallel approach.

5. Conclusions

In this paper, the use of mixed termination rules for global optimization methods was thoroughly presented, and a new global optimization method that takes full advantage of parallel computing structures was afterward proposed. The use of mixed termination rules leads a series of computational global optimization techniques to find the global minimum of the objective function faster, as was also shown by the experimental results.

The termination rules exploited are based on asymptotic criteria and are general enough to be applicable to any global optimization problem and stochastic global optimization technique.

Furthermore, the new stopping rule was utilized in a novel stochastic global optimization technique that involves some basic global optimization techniques. This method is designed to be executed in parallel computation environments. At each step of the method, a different global optimization method is also executed on each parallel computing unit, and these different techniques exchange optimal solutions with each other. In current work, the global optimization techniques of Differential Evolution, Multistart and PSO were used in the processing units, but the method can be generalized to use other stochastic optimization techniques, such as genetic algorithms or simulated annealing. Also, the overall method terminates with the combined termination rule proposed here, and the experimental results performed on a series of well - known test problems from the relevant literature seem to be very promising.

However, the present work is limited to a specific set of optimization methods and termination criteria. Therefore, future extensions of this work may include the integration of additional and more advanced termination criteria, as well as the implementation of alternative global optimization methods in the processing units, such as genetic algorithms or variations of simulated annealing.

Author Contributions: V.C., I.G.T and A.M.G conceived the idea and methodology and supervised the technical part regarding the software. V.C. conducted the experiments. A.M.G. performed the statistical analysis. I.G.T. and all other authors prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research has been financed by the European Union : Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan , under the call RESEARCH – CREATE – INNOVATE, project name “iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code:TAEDK-06195).

Conflicts of Interest: The authors have no conflicts of interest to declare.

References

1. M. Honda, Application of genetic algorithms to modelings of fusion plasma physics, Computer Physics Communications **231**, pp. 94-106, 2018.
2. X.L. Luo, J. Feng, H.H. Zhang, A genetic algorithm for astroparticle physics studies, Computer Physics Communications **250**, 106818, 2020.
3. T.M. Aljohani, A.F. Ebrahim, O. Mohammed, Single and Multiobjective Optimal Reactive Power Dispatch Based on Hybrid Artificial Physics–Particle Swarm Optimization, Energies **12**, 2333, 2019.
4. P.M. Pardalos, D. Shalloway, G. Xue, Optimization methods for computing global minima of nonconvex potential energy functions, Journal of Global Optimization **4**, pp. 117-133, 1994.
5. A. Liwo, J. Lee, D.R. Ripoll, J. Pillardy, H. A. Scheraga, Protein structure prediction by global optimization of a potential energy function, Biophysics **96**, pp. 5482-5485, 1999.
6. J. An, G. He, F. Qin, R. Li, Z. Huang, A new framework of global sensitivity analysis for the chemical kinetic model using PSO-BPNN, Computers & Chemical Engineering **112**, pp. 154-164, 2018.
7. Z.L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on **18** Power Systems, pp. 1187-1195, 2003.
8. M. Basu, A simulated annealing-based goal-attainment method for economic emission load dispatch of fixed head hydrothermal power systems, International Journal of Electrical Power & Energy Systems **27**, pp. 147-153, 2005.
9. Y. Cherruault, Global optimization in biology and medicine, Mathematical and Computer Modelling **20**, pp. 119-132, 1994.
10. E.K. Lee, Large-Scale Optimization-Based Classification Models in Medicine and Biology, Annals of Biomedical Engineering **35**, pp 1095-1109, 2007.

11. M.A. Wolfe, Interval methods for global optimization, Applied Mathematics and Computation **75**, pp. 179-206, 1996.
12. T. Csendes, D. Ratz, Subdivision Direction Selection in Interval Methods for Global Optimization, SIAM J. Numer. Anal. **34**, pp. 922–938, 1997.
13. M. Kawachi, N. Ando, Genetic Algorithms in Search, Optimization & Machine Learning, Artificial Intelligence **7**, pp.: 168, 1989.
14. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science **220**, pp.: 671–680, 1983.
15. F.A. Hashim, E.H. Houssein, M.S. Mabrouk, W. Al-Atabany, S. Mirjalili, Henry gas solubility optimization: A novel physics based algorithm. Future Generation Computer Systems **101**, pp.: 646–667, 2019.
16. E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: A gravitational search algorithm, Information Sciences **179**, pp. 2232–2248, 2009.
17. Du, H., Wu, X., Zhuang, J. (2006). Small-World Optimization Algorithm for Function Optimization. In Proceedings of the International Conference on Natural Computation, Xi'an, China, 24–28 September 2006, pp. 264–273. Doi: 10.1007/11881223_33
18. D.E. Goldberg, J.H. Holland, Genetic Algorithms and Machine Learning, Machine Learning **3**, pp.: 95–99, 1988.
19. S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation **15**, pp. 4–31, 2011.
20. V. Charilogis , I.G. Tsoulos, A. Tzallas, E. Karvounis, Modifications for the Differential Evolution Algorithm, Symmetry **14**, 447, 2022.
21. J. Kennedy, R. Eberhart, Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth,WA, Australia, 27 November–1 December 1995, Vol.: 4, pp. 1942–1948. Doi: 10.1109/ICNN.1995.488968.
22. V. Charilogis, I.G. Tsoulos, Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions, Information **13**, 217, 2022.
23. M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **26**, pp. 29–41, 1996.
24. X.S. Yang, A.H. Gandomi, Bat algorithm: A novel approach for global engineering optimization. Engineering Computations **29**, pp. 464–483, 2012.
25. S. Mirjalili, A. Lewis, The whale optimization algorithm. Advances in Engineering Softwar **95**, pp. 51–67, 2016.
26. S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: Theory and application. Advances in Engineering Software **105**, pp. 30–47, 2017.
27. R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-Aided Design **43**, pp. 303–315, 2011.
28. S.C. Chu, J.F. Roddick, J.S. Pan, A parallel particle swarm optimization algorithm with communication strategies, J. Inf. Sci. Eng **21**, pp. 809-818, 2005.
29. J. Larson and S.M. Wild, Asynchronously parallel optimization solver for finding multiple minima, Mathematical Programming Computation **10**, pp. 303-332, 2018.
30. I.G. Tsoulos, A. Tzallas, D. Tsalikakis, PDoublePop: An implementation of parallel genetic algorithm for function optimization, Computer Physics Communications **209**, pp. 183-189, 2016.
31. R. Kamil, S. Reiji, An Efficient GPU Implementation of a Multi-Start TSP Solver for Large Problem Instances, Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1441-1442, 2012.
32. T. Van Luong., N. Melab, E.G. Talbi, GPU-Based Multi-start Local Search Algorithms. In: Coello C.A.C. (eds) Learning and Intelligent Optimization. LION 2011. Lecture Notes in Computer Science, vol 6683, 2011.
33. K. Barkalov, V. Gergel, Parallel global optimization on GPU, J Glob Optim **66**, pp. 3–20, 2016.
34. Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, J. Hellerstein, GraphLab: A New Framework For Parallel Machine Learning. Source: arXiv, Computer Science, 2010.
35. L. Yangyang, G. Liu, G. Lu, L. Jiao, N. Marturi, R. Shang, Hyper-Parameter Optimization Using MARS Surrogate for Machine-Learning Algorithms. IEEE Transactions on Emerging Topics in Computational Intelligence **99**, pp. 1-11, 2019.
36. H. Yamashiro, H. Nonaka, Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem. Operations Research Perspectives **8**, 2021.
37. H.S. Kim, L. Tsai, Design Optimization of a Cartesian Parallel Manipulator. Journal of Mechanical Design **125**, pp. 43-51, 2003.
38. S. Oh, H. Jong Jang, W. Pedrycz, The design of a fuzzy cascade controller for ball and beam system: A study in optimization with the use of parallel genetic algorithms. ScienceDirect: Engineering Applications of Artificial Intelligence **22**, pp. 261-271, 2009.
39. M. Fatehi, A. Toloei, E. Zio, S.T.A. Niaki, B. Kesh, Robust optimization of the design of monopropellant propulsion control systems using an advanced teaching-learning-based optimization method. Engineering Applications of Artificial Intelligence **126**, 2023.
40. J. Cai, H. Yang, T. Lai, K. Xu,Parallel pump and chiller system optimization method for minimizing energy consumption based on a novel multi-objective gorilla troops optimizer, Journal Of Building Engineering **76**, 2023.
41. Y. Yu, L. Shahabi, Optimal infrastructure in microgrids with diverse uncertainties based on demand response, renewable energy sources and two-stage parallel optimization algorithm, Engineering Application of Artificial Intelligence **123**, 2023.
42. F.J. Ramirez-Gil, C.M. Pere-Madrid, E.C. Nelli Silva, W. Montealerge-Rubio, Sustainable Parallel computing for the topology optimization method: Performance metrics and energy consumption analysis in multiphysics problems, Computing: Informatics and Systems **30**, 2021.

43. M. Tavakolan, F. Mostafazadeh, S.J. Eirdmousa, A. Safari, K. Mirzai, A parallel computing simulation-based multi-objective optimization framework for economic analysis of building energy retrofit: A case study in Iran, Journal of Building Engineering **45**, 2022.
44. G. Lin, (2020). Parallel optimization n based operational planning to enhance the resilience of large-scale power systems. Mississippi State University, Scholars Junction. Online: https://scholarsjunction.msstate.edu/cgi/viewcontent.cgi?article=4435&context=td
45. M. Pang, C.A. Shoemaker, Comparison of parallel optimization algorithms on computationally expensive groundwater remediation designs. Sience of the Total Environment **857**, 2023.
46. A. Ezugwu, A general Framework for Utilizing Metaheuristic Optimization for Sustainable Unrelated Parallel Machine Scheduling: A concise overview. Arxiv, Computer Science, Neural and Evolutionary Computing. Doi: https://doi.org/10.48550/arXiv.2311.12802
47. Y. Censor, S. Zenios, Parallel Optimization: Theory, Algorithms and Applications. Publisher: Oxford University Press, USAISBN: ISBN-13: 978-0195100624. DOI: 10.1093/oso/9780195100624.001.0001.
48. E. Onbaşoğlu, L. Özdamar, Parallel simulated annealing algorithms in global optimization. Journal of global optimization **19**, pp. 27-50, 2001.
49. J.F. Schutte, J.A. Reinbolt, B.J. Fregly, R.T. Haftka, A.D. George, Parallel global optimization with the particle swarm algorithm. International journal for numerical methods in engineering **61**, pp. 2296-2315, 2004.
50. R.G. Regis, C.A. Shoemaker, Parallel stochastic global optimization using radial basis functions. INFORMS Journal on Computing **21**, pp. 411-426, 2009.
51. R. Shonkwiler, Parallel genetic algorithms. In ICGA (pp. 199-205), 1993.
52. E. Cantú-Paz, A survey of parallel genetic algorithms, Calculateurs paralleles, reseaux et systems repartis **10**, pp. 141-171, 1998.
53. R.A.G. Lizárraga, Parallel Computing for Real-Time Image Processing. DOI:10.20944/preprints202408.0040.v1
54. A. Afzal, Z. Ansari, A. Rimaz Faizabadi, M. K. Ramis, Parallelization Strategies for Computational Fluid Dynamics Software: State of the Art Review. Archives of Computational Methods in Engineering **24**, pp. 337–363, 2017)
55. Y. Zou, Y. Zhu, Y. Li, F.X. Wu, J. Wang, Parallel computing for genome sequence processing, 2021.
56. C.B. Lucasius, G. Kateman, Genetic algorithms for large-scale optimization in chemometrics: An application. TrAC Trends in Analytical Chemistry. **10**, pp. 254-261, 1991.
57. M. Sangeetha, A. Sabari, Genetic optimization of hybrid clustering algorithm in mobile wireless sensor networks. Sensor Review **38**, 2018.
58. A. Ghaheri, S. Shoar, M. Naderan, S. Shahabuddin Hoseini, The Applications of Genetic Algorithms in Medicine. Journal List Oman Med J. Nov **30**, pp. 406–416, 2015.
59. F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial. Chemometrics and Intelligent Laboratory Systems **149**, pp. 153-165, 2015.
60. E. García-Gonzalo, J.L. Fernández-Martínez, A Brief Historical Review of Particle Swarm Optimization (PSO). Journal of Bioinformatics and Intelligent Control, Volume 1, Number 1, June 2012, pp. 3-16(14). American Scientific Publishers. DOI:https://doi.org/10.1166/jbic.2012.1002
61. M. Jain, V. Saihjpal, N. Singh, S.B. Singh, An Overview of Variants and Advancements of PSO Algorithm. MDPI, Applied Sciences **12**, 8392, 2022.
62. A.A. de Moura Meneses, M. Dornellas, M.R. Schirru, Particle Swarm Optimization applied to the nuclear reload problem of a Pressurized Water Reactor, Progress in Nuclear Energy **51**, pp. 319-326, 2009.
63. R. Shaw, S. Srivastava, Particle swarm optimization: A new tool to invert geophysical data, Geophysics **72**, 2007.
64. C. O. Ourique, E.C. Biscaia, J.C. Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, Computers & Chemical Engineering **26**, pp. 1783-1793, 2002.
65. H. Fang, J. Zhou, Z. Wang et al, Hybrid method integrating machine learning and particle swarm optimization for smart chemical process operations, Front. Chem. Sci. Eng. **16**, pp. 274–287, 2022.
66. M.P. Wachowiak, R. Smolikova, Yufeng Zheng, J.M. Zurada, A.S. Elmaghraby, An approach to multimodal biomedical image registration utilizing particle swarm optimization, IEEE Transactions on Evolutionary Computation **8**, pp. 289-301, 2004.
67. Y. Marinakis. M. Marinaki, G. Dounias, Particle swarm optimization for pap-smear diagnosis, Expert Systems with Applications **35**, pp. 1645-1656, 2008.
68. J.B. Park, Y.W. Jeong, J.R. Shin, K.Y. Lee, An Improved Particle Swarm Optimization for Nonconvex Economic Dispatch Problems, IEEE Transactions on Power Systems **25**, pp. 156-16**216**6, 2010.
69. V. Feoktistov, Differential Evolution. In Search of Solutions. Optimization and Its Applications, Springer. Doi: https://doi.org/10.1007/978-0-387-36896-2
70. M.P. Bilal, H. Zaheer, L. Garcia-Hernandez, A. Abraham, Differential Evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence **90**, 103479, 2020.
71. P. Rocca, G. Oliveri, A. Massa, Differential Evolution as Applied to Electromagnetics, IEEE Antennas and Propagation Magazine. **53**, pp. 38-49, 2011.
72. W.S. Lee, Y.T. Chen, Y. Kao, Optimal chiller loading by differential evolution algorithm for reducing energy consumption, Energy and Buildings **43**, pp. 599-604, 2011.
73. Y. Yuan, H. Xu, Flexible job shop scheduling using hybrid differential evolution algorithms, Computers & Industrial Engineering **65**, pp. 246-260, 2013.

74. L. Xu, H. Jia, C. Lang, X. Peng, K. Sun, A Novel Method for Multilevel Color Image Segmentation Based on Dragonfly Algorithm and Differential Evolution, IEEE Access **7**, pp. 19502-19538, 2019.
75. R. Marti, M.G.C. Resende, C. Ribeiro, Multi-start methods for combinatorial optimization. European Journal of Operational Research Volume **226**, pp. 1-8, 2013.
76. R. Marti, J. Moreno-Vega, A. Duarte, Advanced Multi-start Methods. Handbook of Metaheuristics, pp 265–281, 2010.
77. W. Tu, R.W. Mayne, Studies of multi-start clustering for global optimization, International Journal for Numerical Methods in Engineering, 2002.
78. Y.H. Dai, Convergence properties of the BFGS algoritm. SIAM Journal on Optimization **13**, pp. 693-701, 2002.
79. I.G. Tsoulos, Modifications of real code genetic algorithm for global optimization, Appl. Math. Comput **203**, pp. 598–607, 2008.
80. M. Gaviano, D.E. Ksasov, D. Lera, Y.D. Sergeyev, Software for generation of classes of test functions with known local and global minima for global optimization, ACM Trans. Math. Softw. **29**, pp. 469-480, 2003.
81. J.E. Lennard-Jones, On the Determination of Molecular Fields, Proc. R. Soc. Lond. A **106**, pp. 463–477, 1924.
82. Z.B. Zabinsky, D.L. Graesser, M.E. Tuttle, G.I. Kim, Global optimization of composite laminates using improving hit and run, In: Recent advances in global optimization, pp. 343-368, 1992.
83. V. Charilogis, I.G. Tsoulos, A Parallel Implementation of the Differential Evolution Method, Analytics **2**, pp. 17–30, 2023.
84. V. Charilogis, I.G. Tsoulos, A. Tzallas, An Improved Parallel Particle Swarm Optimization. SN Computer Science **4**, 766, 2023.
85. C.A. Floudas, P.M. Pardalos, C. Adjiman, W. Esposoto, Z. Gümüs, S. Harding, J. Klepeis, C. Meyer, C. Schweiger, Handbook of Test Problems in Local and Global Optimization, Kluwer Academic Publishers, Dordrecht, 1999.
86. M. Montaz Ali, C. Khompatraporn, Z.B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization **31**, pp 635-672, 2005.
87. Z.M. Gao, J. Zhao, Y.R. Hu, H.F. Chen, The Challenge for the Nature - Inspired Global Optimization Algorithms: Non-Symmetric Benchmark Functions. IEEE Access, July 26, 2021.
88. R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.
89. S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer. Advances in engineering software **69**, pp. 46-61, 2014.