An improved parallel Particle Swarm Optimization

Vasileios Charilogis, Ioannis G. Tsoulos*, Alexandros Tzallas

July 25, 2023

Department of Informatics and Telecommunications, University of Ioannina

Abstract

In the area of global optimization, a variety of techniques have been developed to find the global minimum. These techniques, in most cases, require a significant amount of computational resources and time to complete and therefore there is a need to develop parallel techniques. In addition, the wide spread of parallel architectures in recent years greatly facilitates the implementation of such techniques. Among the most widely used global optimization techniques is the Particle Swarm Optimization technique. In this work, a series of modifications are proposed in the direction of efficient parallelization for Particle Swarm Optimization. These modifications include an innovative velocity calculation mechanism that has also been successfully used in the serial version of the method, mechanisms for propagating the best particles between parallel computing units, but also a process termination mechanism, which has been properly configured for efficient execution in parallel computing environments. The proposed technique was applied to a multitude of computational problems from the relevant literature and the results were more than promising, since it was found that increasing the computational threads can significantly reduce the required number of function calls to find the global minimum. The proposed technique is at rate of 50-70% of the required number of function calls compared to other optimization techniques. This reduction is visible even if 1-2 parallel processing units are used. In addition, with the increase in parallel processing units, a drastic reduction in the number of calls is observed and therefore a reduction in the required computing time, which can reach up to 70%.

Keywords: Optimization, Parallel methods, Evolutionary techniques, Stochastic methods, Termination rules.

1 Introduction

The global optimization problem is usually defined as:

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

with S:

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

where the function f is assumed to be continuous and differentiable. There is a wide range of problems in the relevant literature that can be treated as global optimization problems. In the physical, for example, Yang et al proposed a multiobjective genetic algorithm on an accelerator lattice[1], Iuliano proposed global optimization techniques for benchmark aerodynamic cases and Duan et al used global optimization techniques for Conceptual Rainfall-Runoff Models[3]. In the area of chemistry, Heils and Johnston provided a review on the usage of global optimization techniques on clusters[4], Shin et al proposed a software that utilizes global optimization methods for protein - ligand docking[5] and Liwo et al proposed a global optimization algorithm for protein structure prediction by minimizing the potential energy function[6]. In addition, in the area of economic sciences, Gaing proposed a Particle Swarm Optimization method to solve the economic dispatch[7] and Maranas et al proposed global optimization methods for tackling long - term financial planning problems[8]. Furthermore, in medicine, Lee proposed large - scale optimization - based classification models applied in medical problems[9] and Cherruault reviewed some global optimization methods that were applied to medicine problems[10].

There are several ways to categorize global optimization techniques, but most researchers suggest dividing them into two broad categories: deterministic and stochastic techniques. In the area of deterministic techniques, the one with the greatest spread among researchers is the Interval method[11, 12], where the initial interval of values S of the objective function is continuously divided into smaller parts. The segments that may not contain the total minimum are discarded and eventually a very narrow interval of values should be left which will contain the total minimum of the objective function. Also, in the area of deterministic methods, many related works with various applications, such as the work of Maranas and Floudas that proposed a deterministic method for molecular structure determination [13], the TRUST method for global optimization [14], the method of Evtushenko and Posypkin for global box - constrained optimization [15], a method that uses smooth diagonal auxiliary functions to approximate the global minimum [16] etc. Also, Kunde et al [17] suggested the usage of deterministic global optimization methods for chemical problems. Furthermore, recently, Sergeyev et al [18] proposed a systematic comparison between deterministic and stochastic methods used to handle the global optimization problem. Recently, Yassami and Ashtari proposed a hybrid optimization method to handle optimization problems [19].

On the other hand, in the category of stochastic techniques, where the greatest research effort is made, techniques are presented that look for the total mini-

mum using stochastic methods, which, of course, does not guarantee finding the total minimum. In this vast area of research endeavors one encounters methods such as controlled random search methods[20, 21, 22] which can be considered as a direct search method, simulated annealing methods [23, 24, 25], differential evolution methods [26, 27], particle swarm optimization methods [28, 29, 30], Ant Colony Optimization [31, 32], Genetic Algorithms [33, 34, 35], etc. In addition, since in recent years there has been an extremely wide spread of parallel computing units, many studies have appeared that utilize the modern parallel processing units [36, 37, 38] to tackle the global optimization problem. Some research that one can study regarding metaheuristic algorithms is presented in some recent papers [39, 40, 41]. This paper suggests a number of directions for efficient parallelization of particle swarm optimization (PSO) techniques.

The PSO method is inspired by the observations of Eberhart and Kennedy in the 1990s. The two observed in detail the behavior of birds as they move through space looking for food and then formulated a technique that falls under the domain of global optimization, using the experience they gained from their observations. In the new technique they formulated, the atoms or particles move in the research space of the problem in search of the minimum of the objective function. These particles are directly related to two basic characteristics: the current position of each particle and the velocity at which it moves in search of the global minimum. The current position is denoted as $\overrightarrow{x}$ and the velocity is referred to as $\overrightarrow{u}$. In addition, for each particle, the best position in which it has been found in the past (the one with the smallest value for the objective function) is kept in memory, but also, for the total population, its best position is kept. The method moves the particles to search for the global minimum through an iterative process, in which the positions of the particles at each iteration are derived from each particle's current position, the best position it has found in previous iterations, and the overall best position of the population.

Due to the simplicity of the method and the small number of parameters that should be set, this method has been applied to many difficult problems from all areas of the sciences, such as problems that arise in physics [42, 43], chemistry [44, 45], medicine [46, 47], economics [48] etc. Also, the PSO method was successfully applied recently in many practical problems such as flow shop scheduling [49], successful development of electric vehicle charging strategies [50], emotion recognition [51], robotics [52] etc. A extensive tutorial on PSO methods can be found in the work of Marini and Walczak [53]. Moreover, a recent overview of PSO variants can be found in the work of Jain et al [54]. Since the technique has gained a lot of popularity in recent years, many works have been developed based on it, such as combination with the mutation mechanism [55, 56, 57], methods that propose new techniques to initialize the velocity vector [58], hybrid techniques [59, 60, 61], parallel techniques [62, 63, 64]. Also, since the calculation of the velocity of the particles is a determining factor for the effectiveness of the method, several techniques were formulated for the calculation of the velocity [65, 66, 67]. The method of particle swarm optimization has been successfully combined with other global optimization techniques, such as the work of Bogdanova et al [68] who suggested a combination of Grammatical

Evolution with PSO [69]. Also Pan et al [70] suggested a method that combines the PSO method with a variation of the simulated annealing. Likewise, Mughal et al [71], proposed a hybrid technique that effectively combines PSO and simulated annealing for solar energy problems. In the same way, Lin et al [72] suggested a hybrid method of PSO and differential evolution for some optimization problems. Additionally, a number of techniques have been presented in which local minimization methods are applied to some iterations of Particle Swarm Optimization [73, 74]. Also, Gao et al suggested the usage of a PSO method to reduce energy consumption [75]. Of course, these techniques significantly increase the computational time required by the method, but also significantly improve the performance of the method.

Since the topic of global optimization is very widely applied in many fields but also due to the fact that it requires significant computing resources, techniques that take advantage of parallel architectures have to be developed to reduce the required computing time. For example, Olenšek et al[76] proposed an asynchronous parallel global optimization method based on simulated annealing and differential evolution applied on a series of benchmark global optimization problems. Also, Regis and Shoemaker [77] employed Radial Basis Functions [78] for parallel global optimization. A systematic review of parallel metaheuristics can be found in the work of Alba et al [79]. Also, a review of metaheuristics that are implemented on modern GPU architectures can be found in the work of Essaid et al [80]. The Particle Swarm Optimization method is an excellent candidate method for parallelization, since it is based on a series of computational solutions that partially act independently of each other. For example, Koh et al proposed a parallel asynchronous implementation of the PSO algorithm [81], Tewolde et al [82] proposed a Multi - swarm PSO algorithm where the swarm is divided into a set of subswarms that are executed in parallel. Also, Ouyang et al [83] proposed a parallel implementation of the PSO algorithm using the CUDA programming library to solve the lD heat conduction equation. Furthermore, Campos et al [84] investigated the impact of communication strategies on a parallel PSO implementation used for solving many objective optimization problems. A survey of parallel PSO algorithms can be found in the work of Lalwani et al [85].

In the present work, the initial population of Particle Swarm Optimization is divided into independent populations running on different parallel computing units, which will also be called islands. In addition, a number of techniques are proposed, which aim at efficient communication between the independent islands on the one hand, and at the early termination of the overall method on the other. These techniques include a new method of calculating particle velocity, a new termination rule specifically modified for parallel techniques, and a new way of propagating the best particles among the parallel computing units involved in the overall method. The proposed velocity calculation is proposed by Eberhart and Shi [86] and it is based on random values. Also, the proposed stopping rule is based on the work of Charilogis and Tsoulos [87], giving excellent results when applied on a series of well - known optimization problems. However, the proposed termination technique is modified appropriately for parallel computing

environments.

The main disadvantages of the PSO method are its tendency to converge quickly to a local minimum in some complex or high-dimensional objective functions, and its performance heavily depends on the initial particle distribution, as incorrect values may lead to suboptimal solutions. The proposed method overcomes these issues and aims to improve the algorithm's performance in terms of speed and reduced consumption of computational resources. In general, any optimization method can be parallelized, ensuring the ability to utilize fewer computational resources and saving time in the resolution process.

The rest of this article is organized as follows: in section 2 the proposed method and the new approaches in Particle Swarm Optimization are discussed in detail, in section 3 the used test functions as well the experimental results are fully outlined and finally in section 4 some conclusions and future guidelines are listed.

2 The proposed method

This section will begin with a detailed description of the serial method to be performed on each parallel processing unit as well as the overall parallel method proposed in this work. Subsequently, the basic components of the proposed process, such as the calculation of the velocity, the proposed termination rule and the method of propagating points between the parallel computing units will be thoroughly analyzed.

In every parallel processing unit a PSO algorithm is executed to locate the global minimum of the objective function. The mains steps for this algorithm are outlined in detail in Algorithm 1.

Algorithm 1 The base PSO algorithm executed in one processing unit.

1. **Initialization Step** .
   (a) **Set** iter $= 0$.
   (b) **Set** m as the total number of particles.
   (c) **Set** $\text{iter}_{\max}$ as the maximum number of iterations allowed for the serial algorithm.
   (d) **Initialize** randomly, the positions $x_1, x_2, ..., x_m$ for the particles. The initial positions must be within the domain of the objective function.
   (e) **Initialize** randomly the velocities $u_1, u_2, ..., u_m$.
   (f) **For** $i = 1..m$ do $p_i = x_i$. The vector p_i represents the best located position of particle i.
   (g) **Set** $p_{\text{best}} = \arg\min_{i \in 1..m} f(x_i)$
2. **Termination Check Step** . The termination criterion for the serial algorithm is checked. If the termination criterion is true, then the method terminates.
3. **For** $i = 1..m$ **Do**
   (a) **Compute** the velocity u_i as a combination of the vectors u_i, p_i and p_{best}
   (b) **Set** the new position for the particle as: $x_i = x_i + u_i$
   (c) **Calculate** the $f(x_i)$ for particle x_i using the objective function $f(x)$.
   (d) **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** iter $=$ iter $+ 1$.
7. **Goto** Step 2

The base PSO algorithm as described in Algorithm 1, computes at every iteration the new position of the particle i using the following operation:

$$x_i = x_i + u_{i+1} \tag{2}$$

In most cases the new velocity could be a linear combination of the previously computed velocity and the best values p_i and p_{best}. A typical computation of the new velocity is defined as:

$$u_i = \omega u_i + r_1 c_1 (p_i - x_i) + r_2 c_2 (p_{\text{best}} - x_i) \tag{3}$$

where

1. The variables $r_1,\ r_2$ are random numbers defined in $[0, 1]$.
2. The constants $c_1,\ c_2$ are defined in $[1, 2]$.
3. The variable ω is called inertia and it was suggested by Shi and Eberhart [28]. In the current article a simple inertia calculation is used and calculated using:

$$\omega_{\text{iter}} = 0.5 + \frac{r}{2} \tag{4}$$

The variable r is a a random number with $r \in [0, 1]$.

2.1 The parallel algorithm

The overall parallel algorithm, which runs on N_I independent computing units, is shown in algorithm 2.

Algorithm 2 The implemented parallel algorithm.

1. **Set** as N_I the total number of parallel processing units.
2. **Set** as N_R as the number of iterations, after which each processing unit will send its best particles to the remaining processing units.
3. **Set** N_P the number of migrated particles between the parallel processing units.
4. **Set** $K = 0$ the iteration number.
5. **For** $j = 1, .., N$ do in parallel
   (a) **Execute** an iteration of the PSO algorithm described in algorithm 1 on processing unit j.
   (b) **If** K **mod** $N_R = 0$,**then**
      i. **Get** the best N_P particles from algorithm j.
      ii. **Propagate** these N_P particles to the rest of processing units using some propagation scheme that will be described subsequently.
   (c) EndIf
6. **End For**
7. **Update** $K = K + 1$
8. **Check** the proposed termination rule. If the termination rule is valid, then goto step 8a else goto step 5.
   (a) **Terminate** and report the best value from all processing units.. Apply a local search procedure to this located value to enhance the located global minimum. In the proposed algorithm a BFGS variation of Powell [88] was used a local search procedure.

The two main components of the proposed parallel implementation are the particle propagation mechanism between the parallel computing units and the proposed termination rule. In the first case, it must be decided which units will send the best particles to other parallel units. Units receiving particles replace their worst members with the ones they receive. In the second case, a termination rule based on asymptotic observations is proposed for early termination of the parallel technique. In the proposed termination technique, the overall algorithm is terminated if it holds with some certainty if some asymptotic termination criterion holds for at least one parallel computing unit. These components will be discussed in the following subsections. The overall algorithm is graphically outlined in Figure 1.

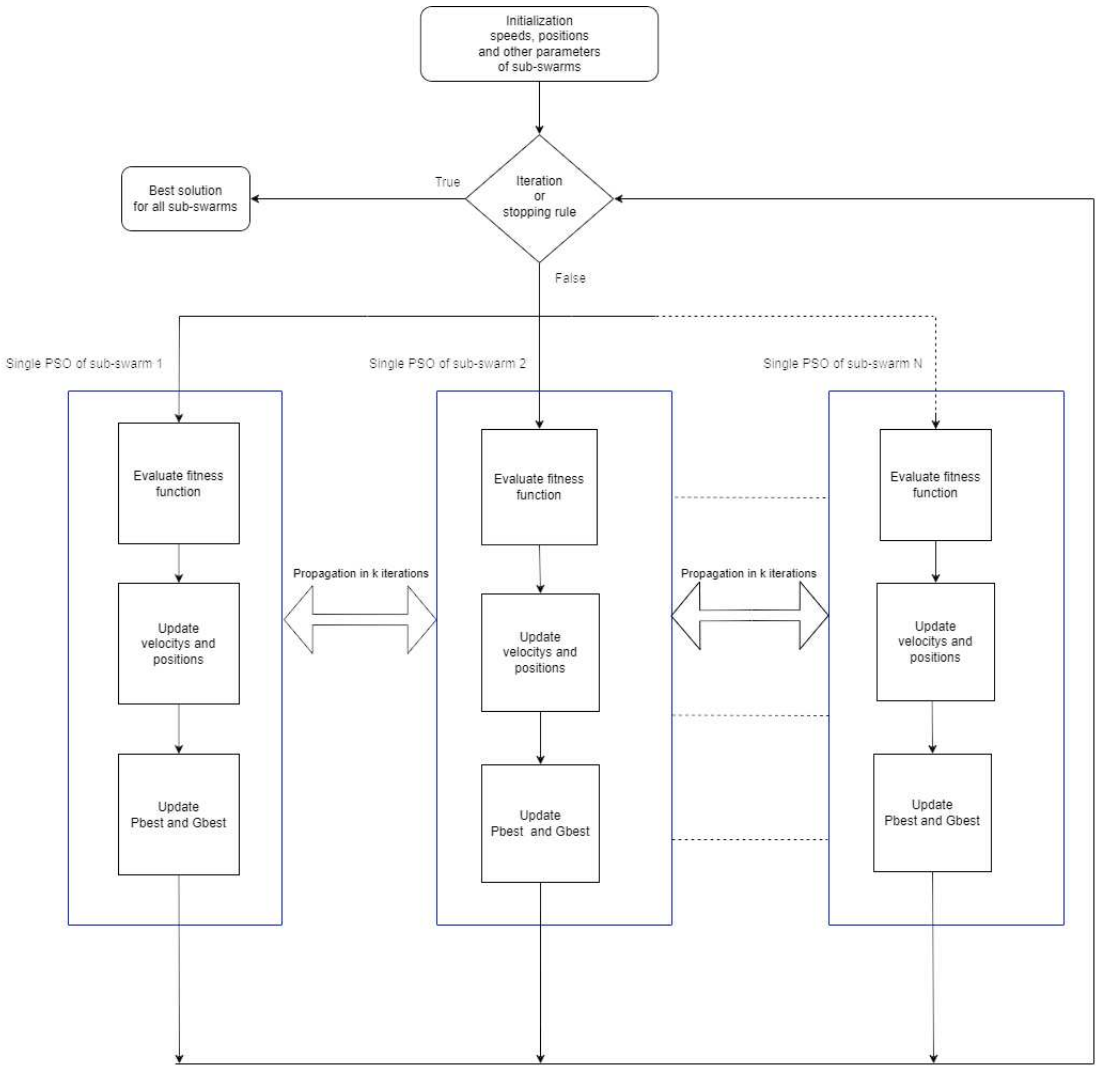

Figure 1: The overall algorithm.

2.2 Propagation mechanism

During the execution of the parallel algorithm and periodically, the processing units propagate their best particles (those with the lowest function value) to the remaining processing units. This propagation can be achieved in the following possible ways:

1. **1 to 1**. In this propagation scheme, a randomly selected processing unit will send to some other randomly selected unit its N_P best particles.

2. **1 to N**. During this scheme, a randomly selected unit will send its best N_P particles to the remaining units.

3. **N to 1**. In this scheme, all processing units will send the corresponding N_P best particles of each unit to a randomly selected unit.

4. **N to N**. For this scheme, all processing units will send the corresponding N_P best particles to all.

All propagation schemes are demonstrated graphically in Figure 2.

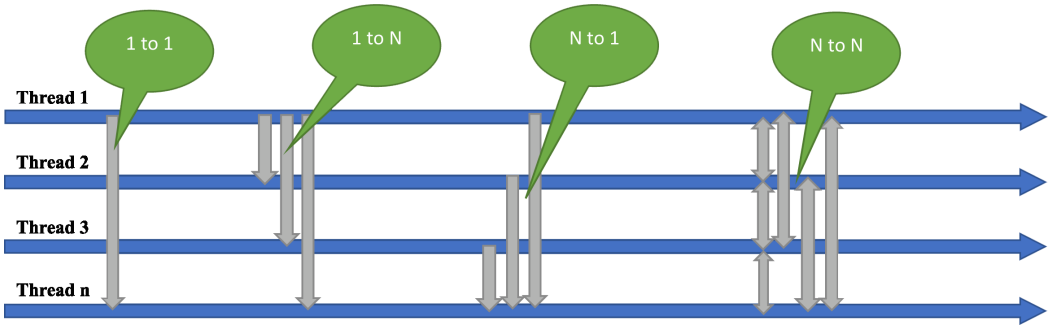

Figure 2: A graphic presentation of all propagation schemes.

2.3 Stopping rule

In the proposed technique, a distinct termination rule is checked on each parallel processing unit. This rule is formulated as follows:

$$\delta_i^{(k)} = \left| f_{i,\min}^{(k)} - f_{i,\min}^{(k-1)} \right| \tag{5}$$

This quantity is calculated on every iteration k. The value $f_{i,\min}^{(k)}$ is the best function value for unit i at iteration k. If the above quantity is less than a predetermined limit ϵ for N_M continuous repetitions, then the algorithm executed on this unit is terminated. In present work, if a parallel processing unit is terminated, then the overall process is also terminated.

3 Experiments

In order to be able to establish the efficiency of the method in finding the total minimum of continuous functions, a series of experiments were performed in which objective functions from the relevant literature were used[89, 90]. These objective functions have been extensively studied by many researchers in many publications[91, 92, 93, 94]. In this series of experiments, beyond the ability of the method to find the global minimum, the effect of changes in a number of critical parameters, such as propagation techniques, on the behavior of the

proposed method was also evaluated. In addition, experiments were conducted in which the actual execution time was measured for various objective functions. In these experiments, the actual reduction in execution time was also recorded as the number of parallel processing units increased.

3.1 Test functions

The test functions used in the conducted experiments are listed in Table 1.

Table 1: The objective functions used in the experiments

NAME	FORMULA	DIMENSION
Cigar function	$f(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$	$n = 10$
Bf1	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	$n = 2$
Bf2	$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	$n = 2$
Branin	$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	$n = 2$
CM4	$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	$n = 4$
Camel	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$n = 2$
Discus10	$f(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$	$n = 10$
Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2-\pi)^2 - (x_1-\pi)^2\right)$	$n = 2$
Exp	$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \leq x_i \leq 1$	$n = 4, 16, 64$
Griewank2	$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}$	$n = 2$
Gkls[95]	$f(x) = \mathrm{Gkls}(x, n, w)$	$n = 2, 3$
Hansen	$f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]$	$n = 2$
Hartman 3	$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$n = 3$
Hartman 6	$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$n = 6$
Potential[96]	$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$	$n = 3, 5$
Rastrigin	$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$	$n = 2$
Rosenbrock	$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	$n = 4, 8$
Shekel 5	$f(x) = -\sum_{i=1}^{5}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	$n = 4$
Shekel 7	$f(x) = -\sum_{i=1}^{7}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	$n = 4$
Shekel 10	$f(x) = -\sum_{i=1}^{10}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$	$n = 4$
Sinu	$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right)$	$n = 4, 8$
Test2n	$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i$	$n = 4, 5, 6, 7$
Test30n	$\frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i-1)^2\left(1+\sin^2(3\pi x_{i+1})\right)\right) + (x_n-1)^2\left(1+\sin^2(2\pi x_n)\right)$	$n = 3, 4$

3.2 Experimental results

Experiments were performed on the objective functions presented previously. Each experiment was run 30 times using different random numbers each time and the averages of the experiments are plotted in the tables below. For the parallelization the freely available OpenMP library [97] was utilized. The program-

ming language used was ANSI C++ and the OPTIMUS optimization package freely available from `https://github.com/itsoulos/OPTIMUS` (accessed on 3 March 2023) was used. The most critical parameters of the proposed method are listed in table 2.

Table 2: The values for most critical parameters of the algorithm.

PARAMETER	MEANING	VALUE
m	Number of particles	200
$\text{iter}_{\max}$	Maximum number of generations	200
c_1	Cognitive parameter for PSO	1.0
c_2	Social learning parameter for PSO	1.0
N_R	The particle exchange frequency	15
ϵ	Small value used in termination rule	10^{-6}
N_M	Number of repetitions used in termination rule	15

In the first set of experiments, the serial version of the proposed technique, i.e. running on a single processor, was compared with the folllowing widely used global optimization techniques:

1. A genetic algorithm, denoted as GENETIC in the the following table.
2. A simple version of Particle Swarm Optimization, denoted as PSO in the following table.

In all cases the same number of particles or chromosomes and the same maximum number of generations were used. In addition, after each method was executed, the local optimization method BFGS was used to improve the global minimum found. The results for this experiment are reported in Table 3.

Table 3: Comparison of the proposed method against two other global optimization techniques. The number of processing units is set to $N_I = 1$.

Function	GENETIC	PSO	GALIB	PGBWPSO	PROPOSED
BF1	9581	19144(0.83)			12625
BF2	10014	19121(0.90)			13108
BRANIN	9289	17760			8574
CIGAR10	40226	39553			40274
CM4	15360	22829			11512
DISCUS10	40216	22359			37848
EASOM	9994	2897			4608
ELP10	40273	31192			23436
EXP4	14084	21375			9062
EXP16	40215	27755			22408
EXP64	40237	26155			40238
GKLS250	8361	16217			8070
GKLS350	12697(0.97)	20393			10696(0.97)
GRIEWANK2	9298(0.97)	20004(0.87)			11064
POTENTIAL3	27799	20876			12876
POTENTIAL5	40240	25809			38377
HANSEN	14951(0.93)	16945			12467
HARTMAN3	11268	22259			10018
HARTMAN6	21396(0.63)	33679(0.33)			15082(0.53)
RASTRIGIN	8967	16044			9286(0.93)
ROSENBROCK4	40233	26367			25120
ROSENBROCK8	40271	32750			38577
SHEKEL5	19403(0.70)	29079(0.33)			15409(0.43)
SHEKEL7	19376(0.80)	27817(0.47)			14989(0.63)
SHEKEL10	19829(0.77)	26479(0.83)			15087(0.67)
SINU4	15788	23915			12298
SINU8	30928	27834(0.97)			15500
TEST2N4	17109	23983(0.97)			14520(0.70)
TEST2N5	19464	30817			14801(0.47)
TEST2N6	24217	29067(0.90)			17444(0.23)
TEST2N7	26824	32337(0.60)			22780(0.23)
TEST30N3	17575	15660			7814
TEST30N4	17395	23519			8014
TOTAL	**732878(0.96)**	**791990(0.91)**			**573980(0.87)**

In the table, each number in each cell represents the average of the function calls for 30 runs. Also, the numbers in parentheses stand for the percentage of runs in which the global minimum was successfully found. If this percentage is not present, it implies 100% success. In addition, a line has been added at the end of the table showing the total number of function calls for each method. The

experimental results demonstrate that even running the proposed technique on a single computing unit can result to better results than the other techniques in terms of the required number of function calls and, by extension, the required time required to find the global minimum.

In order to evaluate the effect of executing the method on parallel processing units, another experiment was done in which the number of parallel processing units was increased from 1 to 10 and the results are presented in Table 4. Also, a box plot for this experiment is shown in Figure 3. Additionally, a Friedman test was conducted to determine whether there were significant differences in the scores of function calls on five (N_I =1, 2, 4, 5, and 10) repeated number of units and the results are shown in Figure 4. In addition, in order to give greater reliability to the experimental results, the total number of particles was in all runs constant and equal to 200, as defined in the table 2. This means, for example, that when the computing units are 4 each unit will run a serial version of PSO with 50 particles and when the computing units become 5 the number of particles in each unit will be reduced to 40.

Table 4: Experimental results using the proposed method, the propagation scheme was set to 1to1 and the value of N_P was set to 5. In the conducted experiments the number of parallel processing units was varied from 1 to 10.

Function	$N_I = 1$	$N_I = 2$	$N_I = 4$	$N_I = 5$	$N_I = 10$
BF1	12625	11660	9984	10315	6667
BF2	13108	11600	10363	9403	6964
BRANIN	8574	6953	5412	5170	4141
CIGAR10	40274	40180	39763	38887	21291
CM4	11512	12019	12203	12339	9910
DISCUS10	37848	26044	13211	10989	4171
EASOM	4608	3927	3660	3513	3110
ELP10	23436	26469	14268	11100	7462
EXP4	9062	9691	9678	9556	9431
EXP16	22408	15608	18025	21307	21991
EXP64	40238	40177	39856	39731	24234
GKLS250	8070	7809	7225	6853	5591
GKLS350	10696(0.97)	11488	10578	10095	7279
GRIEWANK2	11064	10681	9127	8926	5604
POTENTIAL3	12876	5568	5018	4756	4333
POTENTIAL5	38377	4905	4455	4221	4016
HANSEN	12467	5067	4340	4031	3518
HARTMAN3	10018	10263	10162	9711	8234
HARTMAN6	15082(0.53)	9816(0.73)	7212(0.97)	7194	5935
RASTRIGIN	9286(0.93)	9432	6227	5974	4254
ROSENBROCK4	25120	20084	16454	12244	7574
ROSENBROCK8	38577	25195	22531	18508	9587
SHEKEL5	15409(0.43)	14112(0.77)	8575(0.87)	7898(0.93)	4948
SHEKEL7	14989(0.63)	13800(0.90)	9227(0.93)	8717(0.97)	5050
SHEKEL10	15087(0.67)	14662(0.87)	10268(0.93)	8229(0.93)	4871(0.97)
SINU4	12298	12997	13172	12842	10316
SINU8	15500	15475	16442	16375	12732
TEST2N4	14520(0.70)	15043(0.87)	12346	9769	4566
TEST2N5	14801(0.47)	16097(0.77)	12358(0.90)	9440(0.93)	4813(0.93)
TEST2N6	17444(0.23)	16224(0.47)	9103(0.73)	7855(0.57)	4410(0.53)
TEST2N7	22780(0.23)	20330(0.47)	12699(0.40)	9773(0.50)	4243(0.47)
TEST30N3	7814	7967	7010	6382	5014
TEST30N4	8014	7683	6568	6317	5090
TOTAL	**573980(0.87)**	**479026(0.96)**	**397519(0.96)**	**368460(0.96)**	**251350(0.97)**

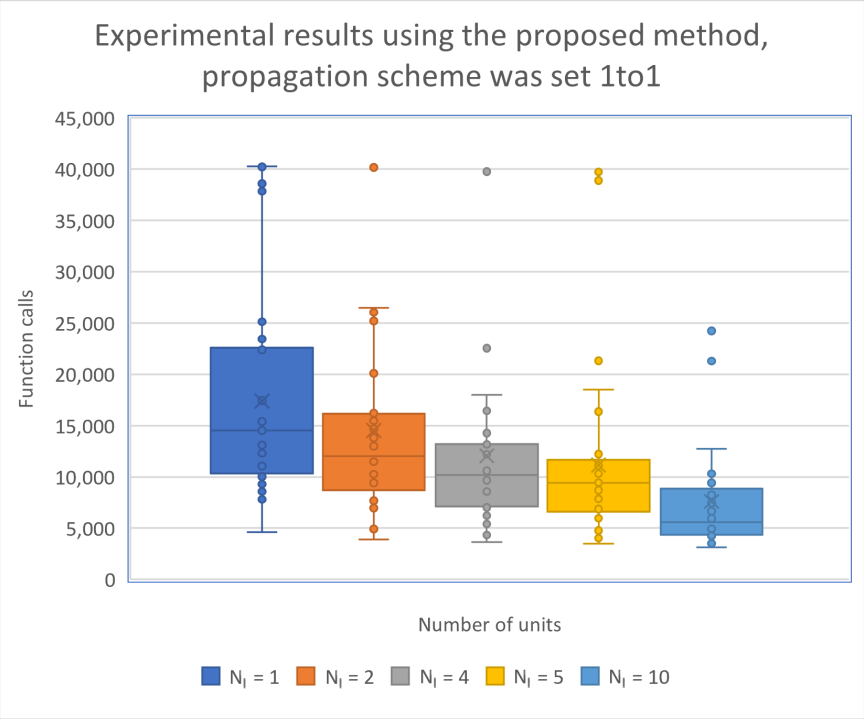

Figure 3: Box - plot for the comparison between different number of processing units. The propagation method was set to 1to1.

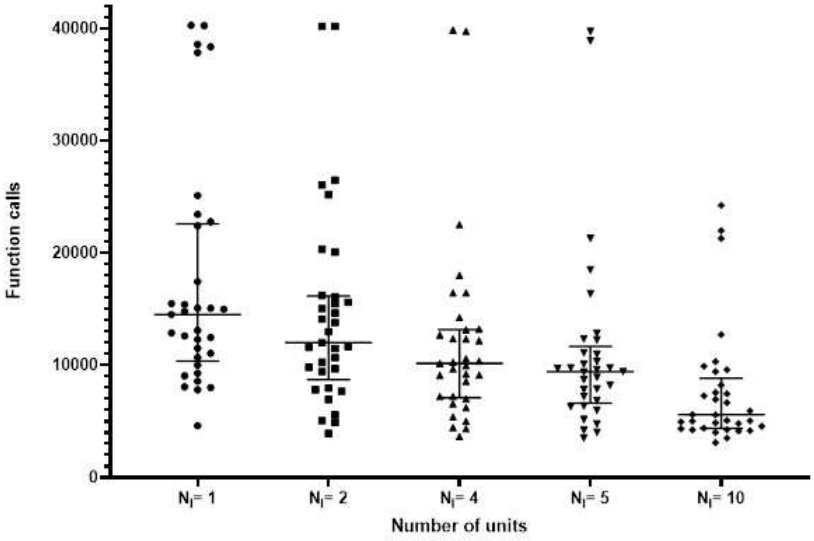

Figure 4: We conducted a Friedman test to determine whether there were significant differences in the scores of function calls on five (N i =1, 2, 4, 5, and 10) repeated number of units. The Friedman chi- square was 94.38 with 4 degrees of freedom, indicating a significant difference between the groups ($p < 0.0001$).

From the experimental results it is clear that increasing the number of parallel processing nodes further reduces the total number of function calls, since more units are used in the attempt to find the global minimum. In addition, a slight increase in the reliability of the method in finding the total minimum is also observed.

Furthermore, to determine the effect of the propagation mechanism on the reliability and speed of the method, another comparative experiment was performed, in which the number of parallel processing units was set to 5 ($N_I = 5$) and all propagation mechanisms were used. The results for this experiment are presented in Table 5.

Table 5: Comparison of different propagation schemes. The number of processing units was set to 5.

Function	1to1	1toN	Nto1	NtoN
BF1	10315	8408	9471	8647
BF2	9403	8024	10578	7730
BRANIN	5170	4633	5203	5789
CIGAR10	38887	25035	35527	34258
CM4	12339	14195	12296	12565
DISCUS10	10989	6484	7667	6154
EASOM	3513	3072	3496	3121
ELP10	11100	13027	9598	7091
EXP4	9556	10654	9517	10607
EXP16	21307	29289	23833	27307
EXP64	39731	11959	38191	26175
GKLS250	6853	6568	6966	7808
GKLS350	10095	10366	10400	9674
GRIEWANK2	8926	6022	7791	5432
POTENTIAL3	4756	4011	4591	4075
POTENTIAL5	4221	4002	4176	3870
HANSEN	4031	3092	4320	3265
HARTMAN3	9711	10154	9316	11110
HARTMAN6	7194	6914(0.73)	6760(0.97)	11242(0.73)
RASTRIGIN	5974	4077	5383	4804
ROSENBROCK4	12244	13930	11511	14710
ROSENBROCK8	18508	15423	16659	20848
SHEKEL5	7898(0.93)	9604(0.90)	7513(0.93)	10657(0.67)
SHEKEL7	8717(0.97)	12204	9404	10805(0.70)
SHEKEL10	8229(0.93)	13418(0.93)	9693	12677(0.83)
SINU4	12842	14757	13154	13376
SINU8	16375	22754	17026	20121
TEST2N4	9769	6633(0.97)	10289	7483(0.83)
TEST2N5	9440(0.93)	4819(0.93)	8077(0.90)	5429(0.80)
TEST2N6	7855(0.57)	5358(0.77)	8354(0.60)	5574(0.43)
TEST2N7	9773(0.50)	5183(0.33)	7417(0.53)	6312(0.40)
TEST30N3	6382	6538	6176	7462
TEST30N4	6317	6938	6473	6305
TOTAL	**368460(0.96)**	**327545(0.96)**	**356826(0.97)**	**352483(0.92)**

This time, the experimental results did not show any clear superiority of one of the propagation methods. However, it appears that the 1-to-N propagation method has slightly better performances than the rest of the best particle propagation techniques among the sub-populations.

The last experiment had to do with the effect of the N_P parameter on

the speed of the method. In it, 5 parallel computing units were used and the propagation method was set to **Nto1**. The experimental results are presented in the Table 6.

Table 6: The effect of the parameter N_P to the speed of the proposed method. The number of threads was set to 5 and the value of N_P was changed from 1 to 10. The propagation scheme used was Nto1.

Function	$N_P = 1$	$N_P = 2$	$N_P = 3$	$N_P = 5$	$N_P = 10$
BF1	10114	9976	10257	9471	9307
BF2	9224	10413	10051	10578	9530
BRANIN	7037	6027	6238	5203	4851
CIGAR10	39244	35793	35811	35527	35835
CM4	11839	11588	12061	12296	11878
DISCUS10	8078	6950	9671	7667	11977
EASOM	3669	3538	3589	3496	3477
ELP10	11656	12382	10643	9598	9641
EXP4	9257	9340	9395	9517	9626
EXP16	27275	24008	22906	23833	23190
EXP64	39679	37334	32126	38191	31119
GKLS250	7250	6893	7116	6966	6568
GKLS350	10281	9236	10088	10400	9552
GRIEWANK2	9259	10096	10297	7791	9527
POTENTIAL3	8471	6694	5770	4591	4829
POTENTIAL5	7127	5869	5301	4176	4844
HANSEN	7978	6230	5591	4320	4573
HARTMAN3	10162	9939	10131	9316	10081
HARTMAN6	10614	9033	8059	6760	7507
RASTRIGIN	7491	6384	6876	5383	5540
ROSENBROCK4	22600	16513	13631	11511	12169
ROSENBROCK8	34125	23004	21027	16659	19740
SHEKEL5	12299	11923	9521	7513	7256
SHEKEL7	13895	12358	10239	9404	9471
SHEKEL10	14130	11536	10235	9693	6936
SINU4	12760	12601	12268	13154	11905
SINU8	16957	17327	16346	17026	17030
TEST2N4	12215	9152	8136	10289	10024
TEST2N5	11384	8429	7934	8077	8935
TEST2N6	11833	8101	7825	8354	9655
TEST2N7	11162	8362	8692	7417	9486
TEST30N3	7178	7015	6829	6176	6216
TEST30N4	6518	6676	6509	6473	6756
TOTAL	**442761**	**390720**	**371169**	**356826**	**359031**

As is evident, increasing the value of the N_P parameter significantly reduces

the required number of function calls up to a value of 5. From then on, the total number of calls appears to be constant. Practically, this means that sending not only the best particle but also those with the immediately lower value further reduces the required number of function calls.

3.3 Experiments for the execution time

To understand the efficiency of the method with respect to execution time, three additional series of experiments were performed. In all experiments, the total number of particles in the particle swarm optimization was fixed at 500 and from 1 to 20 parallel processing units were used and the total execution time was measured for 30 different independent runs. The experiments were conducted on AMD Epyc 7552 equipped with 32GB of RAM. The operating system was the Ubuntu 20.04 operating system.

During the first series of the additional experiments, the High Conditioned Elliptic function, defined as

$$f(x) = \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} x_i^2$$

was used as a test case. In the experiments, the values n=10,20,30,40,50 were used and the experimental results are displayed graphically in the figure 5.

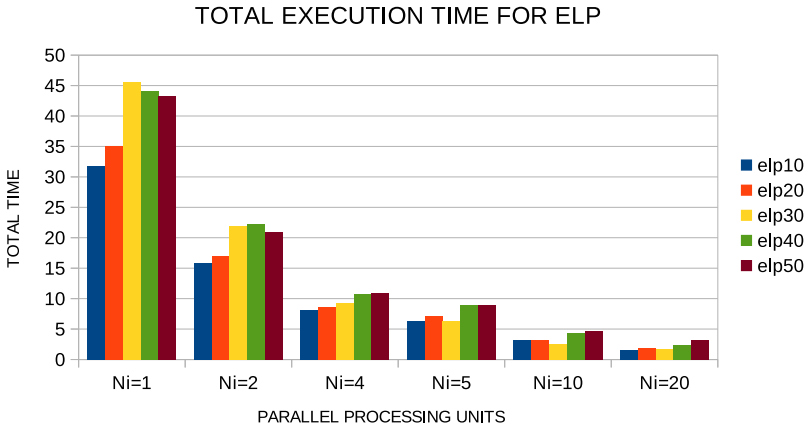

Figure 5: Total execution time for different number of parallel processing threads for the objective function ELP.

From the experimental results, it follows that doubling the parallel processing units almost halves the required execution time in most cases.

During the second series of experiments, different versions of the GKLS function were used for dimensions from 2 to 5. This function was thoroughly tested

also in the work of Sergeyev et al [18]. In each dimension, 10 different instances of the function were used to have a safe estimate of both the performance of the method and the possible decrease in execution time as the parallel processing units. The average execution time for different dimensions of the GKLS function and for different numbers of processing units is graphically shown in Figure 6.

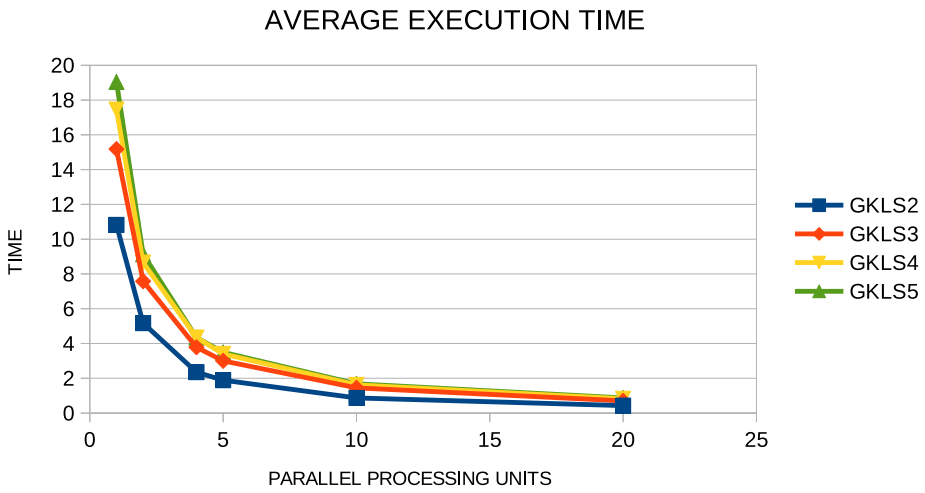

Figure 6: Average time execution for the GKLS function. The dimension of the function varies from 2 to 5.

As is shown, the execution time decreases rapidly with increasing parallel processing units in all instances of the function. Furthermore, the average number of function calls for every case is also measured and the results are plotted in Figure 7.

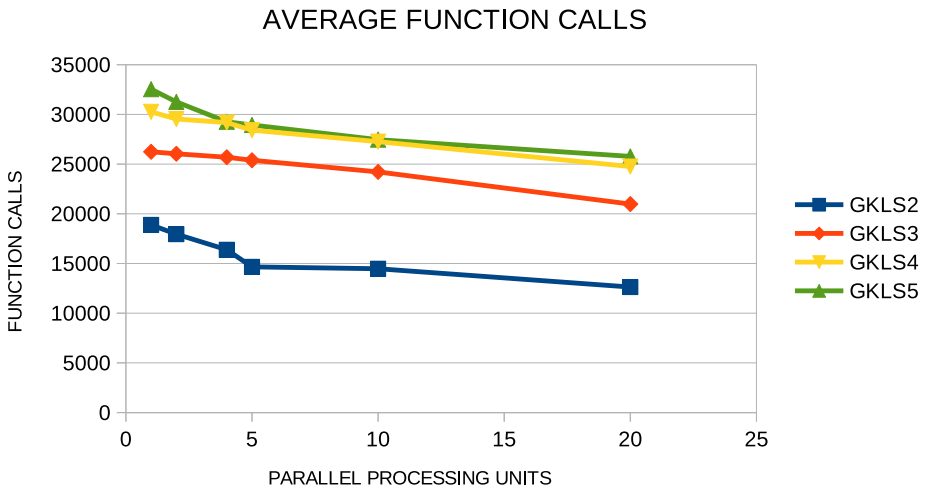

Figure 7: Average function calls for different cases of the GKLS function.

Increasing the number of parallel processing units also reduces the number of required function calls, especially for the low-dimensional cases (Gkls2 and Gkls3). However, the most interesting point of this series of experiments is the ability of the proposed technique to locate the global minimum as the number of parallel computing units increases. This capability is graphically represented in Figure 8.

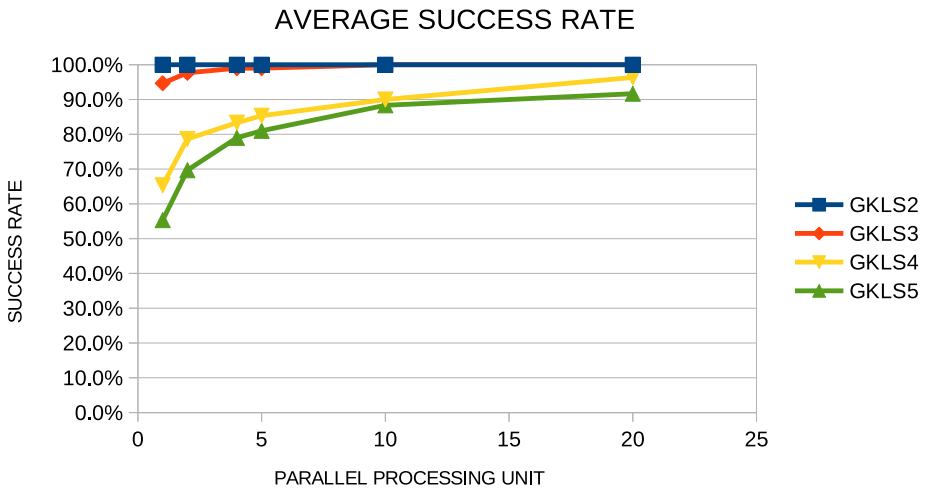

Figure 8: Average success rate in finding the global minimum for different cases of the Gkls function.

From this graph it can be concluded that the increase in parallel processing units significantly improves the efficiency of the technique in finding the global minimum.

In the third series of additional experiments, a more difficult experiment was used, but one that has multiple applications. The proposed method was used to train an artificial neural network [98, 99] with 10 processing nodes applied on two widely used classification problems from the relevant literature: the Wine problem [100, 101] and the Parkinsons problem [102]. The results are graphically outlined in Figure 9.

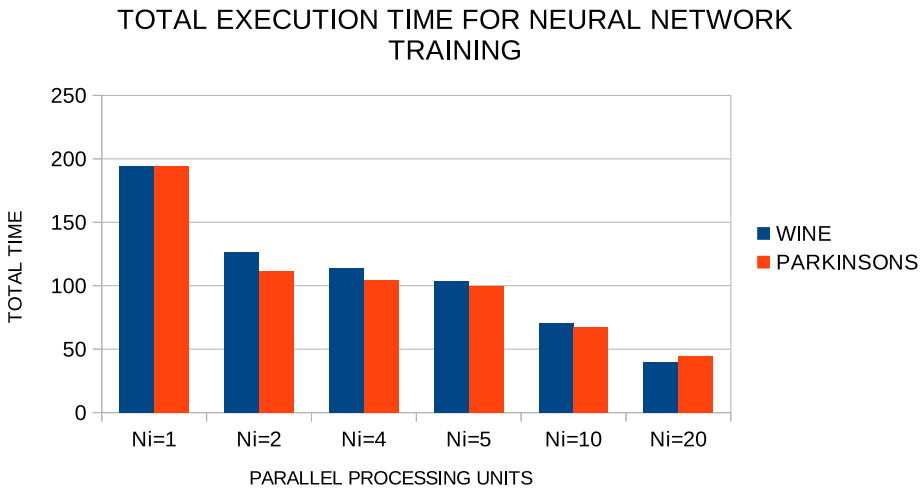

Figure 9: Total execution time for different number of parallel processing threads for neural network training for two classification datasets

Again, the required training time decreases drastically with increasing parallel processing units.

4 Conclusions

In the present work a number of ideas were presented that can lead to an efficient parallel implementation of the PSO technique. The need for parallelization of the method is great, as it is an iterative process involving repeated operations on vectors and these operations for large problems may require significantly large computational time. In the current work, a technique of propagating the best particles between computing units as well as a termination rule of the overall process were introduced. In the case of the propagation of the best particles from the experiments carried out, it appears that it is more efficient to send between the computing units more than the best particle. Furthermore, the propagation method between parallel computing units did not have a drastic effect on the efficiency and speed of the method, although the 1-to-N propagation method appeared to have slightly better results. Furthermore, the increase of the parameter N_P from 1 to 5 decreased the required number of function calls. However, the biggest gain from using the method lies in the increase in parallel processing units. From the experiments performed, it is evident that as parallel processing units increase, the total function calls required to find the global minimum decreases almost for every objective function. In addition, the increase in parallel processing units improved to some extent the efficiency of the method in finding the global minimum. The effectiveness of the proposed method was

also shown in the comparison between its serial version and two other widely available techniques in the relevant literature. In this comparison the serial version of the method required significantly fewer function calls to find each global minimum. Overall, the main contributions of the proposed method are: the velocity calculation mechanism, the propagation among parallel computing units, and the termination rule represents an innovative technique in improving the performance of the algorithm.

On a percentage basis, the proposed technique significantly outperforms the other two compared global optimization techniques. The serial version of the proposed technique reduces the required number of function calls by 22%-28% compared to the genetic algorithm and the particle swarm optimization techniques respectively. However, with the use of more processing units, the reduction in the number of function calls reaches approximately 70%. Furthermore, from the relevant experiments done on the actual execution time, it appears that any doubling of parallel processing threads halves the required execution time required to find the global minimum.

In the future, the important work done in the parallelization of the PSO technique can be continued with the use of new termination rules that will be suitable for execution in parallel computing environments but also a possible extension of the best particle propagation techniques between parallel computing units.

Conflicts of interest

The authors declare no conflict of interest.

References

[1] L. Yang, D. Robin, F. Sannibale, C. Steier, W. Wan, Global optimization of an accelerator lattice using multiobjective genetic algorithms, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **609**, pp. 50-57, 2009.

[2] E. Iuliano, Global optimization of benchmark aerodynamic cases using physics-based surrogate models, Aerospace Science and Technology **67**, pp.273-286, 2017.

[3] Q. Duan, S. Sorooshian, V. Gupta, Effective and efficient global optimization for conceptual rainfall-runoff models, Water Resources Research **28**, pp. 1015-1031 , 1992.

[4] S. Heiles, R. L. Johnston, Global optimization of clusters using electronic structure methods, Int. J. Quantum Chem. **113**, pp. 2091− 2109, 2013.

[5] W.H. Shin, J.K. Kim, D.S. Kim, C. Seok, GalaxyDock2: Protein–ligand docking using beta-complex and global optimization, J. Comput. Chem. **34**, pp. 2647– 2656, 2013.

[6] 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.

[7] Zwe-Lee Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on **18** Power Systems, pp. 1187-1195, 2003.

[8] C. D. Maranas, I. P. Androulakis, C. A. Floudas, A. J. Berger, J. M. Mulvey, Solving long-term financial planning problems via global optimization, Journal of Economic Dynamics and Control **21**, pp. 1405-1425, 1997.

[9] Eva K. Lee, Large-Scale Optimization-Based Classification Models in Medicine and Biology, Annals of Biomedical Engineering **35**, pp 1095-1109, 2007.

[10] Y. Cherruault, Global optimization in biology and medicine, Mathematical and Computer Modelling **20**, pp. 119-132, 1994.

[11] M.A. Wolfe, Interval methods for global optimization, Applied Mathematics and Computation **75**, pp. 179-206, 1996.

[12] T. Csendes and D. Ratz, Subdivision Direction Selection in Interval Methods for Global Optimization, SIAM J. Numer. Anal. **34**, pp. 922–938, 1997.

[13] C.D Maranas, C.A. Floudas, A deterministic global optimization approach for molecular structure determination, J. Chem. Phys. **100**, 1247, 1994.

[14] J. Barhen, V. Protopopescu, D. Reister, TRUST: A Deterministic Algorithm for Global Optimization, Science **276**, pp. 1094-1097, 1997.

[15] Y. Evtushenko, M.A. Posypkin, deterministic approach to global box-constrained optimization, Optim Lett **7**, pp. 819–829, 2013.

[16] Y.D. Sergeyev, D.E. Kvasov, A deterministic global optimization using smooth diagonal auxiliary functions, Communications in Nonlinear Science and Numerical Simulation **21**, pp. 99-111, 2015.

[17] C. Kunde, D. Michaels, J. Micovic, P. Lutze, A. Górak, A. Kienle, Deterministic global optimization in conceptual process design of distillation and melt crystallization, Chemical Engineering and Processing: Process Intensification **99**, pp. 132-142, 2016.

[18] Y.D. Sergeyev, D.E. Kvasov, M.S. Mukhametzhanov, On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Sci Rep **8**, 453, 2018.

[19] M. Yassami, P.A. Ashtari, novel hybrid optimization algorithm: Dynamic hybrid optimization algorithm, Multimedia Tools and Applications, 2023.

[20] W. L. Price, Global optimization by controlled random search, Journal of Optimization Theory and Applications **40**, pp. 333-348, 1983.

[21] Ivan Křivý, Josef Tvrdík, The controlled random search algorithm in optimizing regression models, Computational Statistics & Data Analysis **20**, pp. 229-234, 1995.

[22] M.M. Ali, A. Törn, and S. Viitanen, A Numerical Comparison of Some Modified Controlled Random Search Algorithms, Journal of Global Optimization **11**,pp. 377–385,1997.

[23] S. Kirkpatrick, CD Gelatt, , MP Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.

[24] L. Ingber, Very fast simulated re-annealing, Mathematical and Computer Modelling **12**, pp. 967-973, 1989.

[25] R.W. Eglese, Simulated annealing: A tool for operational research, Simulated annealing: A tool for operational research **46**, pp. 271-281, 1990.

[26] R. Storn, K. Price, Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization **11**, pp. 341-359, 1997.

[27] J. Liu, J. Lampinen, A Fuzzy Adaptive Differential Evolution Algorithm. Soft Comput **9**, pp.448–462, 2005.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[29] Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, Swarm Intelligence **1**, pp 33-57, 2007.

[30] Ioan Cristian Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters **85**, pp. 317-325, 2003.

[31] M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, IEEE Computational Intelligence Magazine **1**, pp. 28-39, 2006.

[32] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operational Research 185, pp. 1155-1173, 2008.

[33] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.

[34] Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.

[35] S.A. Grady, M.Y. Hussaini, M.M. Abdullah, Placement of wind turbines using genetic algorithms, Renewable Energy **30**, pp. 259-270, 2005.

[36] Y. Zhou and Y. Tan, "GPU-based parallel particle swarm optimization," 2009 IEEE Congress on Evolutionary Computation, pp. 1493-1500, 2009.

[37] L. Dawson and I. Stewart, "Improving Ant Colony Optimization performance on the GPU using CUDA," 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 1901-1908, doi: 10.1109/CEC.2013.6557791.

[38] Barkalov, K., Gergel, V. Parallel global optimization on GPU. J Glob Optim 66, 3–20 (2016).

[39] I. BoussaïD, J. Lepagnot, P. Siarry, P., A survey on optimization metaheuristics. Information sciences **237**, pp. 82-117, 2013.

[40] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering **137**, 106040, 2019.

[41] K. Hussain, M.N.M. Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey.Artificial Intelligence Review **52**, pp. 2191-2233, 2019.

[42] Anderson Alvarenga de Moura Meneses, Marcelo Dornellas, Machado Roberto Schirru, Particle Swarm Optimization applied to the nuclear reload problem of a Pressurized Water Reactor, Progress in Nuclear Energy **51**, pp. 319-326, 2009.

[43] Ranjit Shaw, Shalivahan Srivastava, Particle swarm optimization: A new tool to invert geophysical data, Geophysics **72**, 2007.

[44] 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.

[45] 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.

[46] 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.

[47] Yannis Marinakis. Magdalene Marinaki, Georgios Dounias, Particle swarm optimization for pap-smear diagnosis, Expert Systems with Applications **35**, pp. 1645-1656, 2008.

[48] Jong-Bae Park, Yun-Won Jeong, Joong-Rin Shin, Kwang Y. Lee, An Improved Particle Swarm Optimization for Nonconvex Economic Dispatch Problems, IEEE Transactions on Power Systems **25**, pp. 156-16**216**6, 2010.

[49] B. Liu, L. Wang, Y.H. Jin, An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **37**, pp. 18-27, 2007.

[50] J. Yang, L. He, S. Fu, An improved PSO-based charging strategy of electric vehicles in electrical distribution grid, Applied Energy **128**, pp. 82-92, 2014.

[51] K. Mistry, L. Zhang, S. C. Neoh, C. P. Lim, B. Fielding, A Micro-GA Embedded PSO Feature Selection Approach to Intelligent Facial Emotion Recognition, IEEE Transactions on Cybernetics. **47**, pp. 1496-1509, 2017.

[52] S. Han, X. Shan, J. Fu, W. Xu, H. Mi, Industrial robot trajectory planning based on improved pso algorithm, J. Phys.: Conf. Ser. **1820**, 012185, 2021.

[53] F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial, Chemometrics and Intelligent Laboratory Systems **149**, pp. 153-165, 2015.

[54] M. Jain, V. Saihjpal, N. Singh, S.B. Singh, An Overview of Variants and Advancements of PSO Algorithm, Applied Sciences **12**, 8392, 2022.

[55] A. Stacey, M. Jancic, I. Grundy, Particle swarm optimization with mutation, In: 2003 Congress on Evolutionary Computation, 2003. CEC '03., pp. 1425-1430, 2003.

[56] M. Pant, R. Thangaraj, A. Abraham, Particle Swarm Optimization Using Adaptive Mutation, In: 2008 19th International Workshop on Database and Expert Systems Applications, pp. 519-523, 2008.

[57] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706), pp. 72-79, 2003.

[58] A. Engelbrecht, "Particle swarm optimization: Velocity initialization," 2012 IEEE Congress on Evolutionary Computation, pp. 1-8, 2012.

[59] B. Liu, L. Wang, Y.H. Jin, F. Tang, D.X. Huang, Improved particle swarm optimization combined with chaos, Chaos Solitons and Fractals **25**, pp. 1261-1271, 2005.

[60] X.H. Shi, Y.C. Liang, H.P. Lee, C. Lu, L.M. Wang, An improved GA and a novel PSO-GA based hybrid algorithm, Information Processing Letters **93**, pp. 255-261, 2005.

[61] Harish Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Applied Mathematics and Computation **274**, pp. 292-305, 2016.

[62] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, A. D. George, Parallel global optimization with the particle swarm algorithm, Int. J. Numer. Meth. Engng. **61**, pp. 2296-2315, 2004.

[63] B-Il Koh, A.D. George, R.T. Haftka, B.J. Fregly, Parallel asynchronous particle swarm optimization. Int. J. Numer. Meth. Engng., **67**, pp. 578-595, 2006.

[64] G. Venter, J. Sobieszczanski-Sobieski, Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations, Journal of Aerospace Computing, Information, and Communication **3**, pp. 123-137, 2006.

[65] Z.L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on Power Systems **18**, pp. 1187-1195, 2003.

[66] X. Yang, Jinsha Yuan, Jiangy Yuan, H. Mao, A modified particle swarm optimizer with dynamic adaptation, Applied Mathematics and Computation **189**, pp. 1205-1213, 2007.

[67] Y. Jiang, T. Hu, C. Huang, X. Wu, An improved particle swarm optimization algorithm, Applied Mathematics and Computation **193**, pp. 231-239, 2007.

[68] A. Bogdanova, J.P. Junior, C. Aranha, Franken-Swarm: Grammatical Evolution for the Automatic Generation of Swarm-like Meta-Heuristics, In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 411-412, 2019.

[69] M. O'Neill, C. Ryan, Grammatical evolution,IEEE Transactions on Evolutionary Computation **5**, pp. 349-358, 2001.

[70] X. Pan, L. Xue, Y. Lu et al, Hybrid particle swarm optimization with simulated annealing, Multimed Tools Appl **78**, pp. 29921–29936, 2019.

[71] M.A. Mughal, Q. Ma, C. Xiao, Photovoltaic Cell Parameter Estimation Using Hybrid Particle Swarm Optimization and Simulated Annealing, Energies **10**, 2017.

[72] G.H. Lin, J. Zhang, Z.H. Liu, Hybrid particle swarm optimization with differential evolution for numerical and engineering optimization. Int. J. Autom. Comput. **15**, pp. 103–114, 2018.

[73] S. Li, M. Tan, I. W. Tsang, J. T. -Y. Kwok, A Hybrid PSO-BFGS Strategy for Global Optimization of Multimodal Functions, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **41**, pp. 1003-1014, 2011.

[74] G. Wu, D. Qiu, Y. Yu, W. Pedrycz, M. Ma, H. Li, Superior solution guided particle swarm optimization combined with local search techniques, Expert Systems with Applications **41**, pp. 7536-7548, 2014.

[75] Z. Gao, J. Yu, A. Zhao, Q. Hu, S. Yang, Optimal chiller loading by improved parallel particle swarm optimization algorithm for reducing energy consumption, International Journal of Refrigeration **136**, pp. 61-70, 2022.

[76] J. Olenšek, T. Tuma, J. Puhan, Á Bűrmen, A new asynchronous parallel global optimization method based on simulated annealing and differential evolution, Applied Soft Computing **11**, pp. 1481-1489, 2011.

[77] R.G. Regis, C.A. Shoemaker., Parallel stochastic global optimization using radial basis functions, INFORMS Journal on Computing **21**, pp. 411-426, 2009.

[78] J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.

[79] E. Alba, G. Luque, S. Nesmachnow, Parallel metaheuristics: recent advances and new trends. International Transactions in Operational Research **20**, pp. 1-4, 2013.

[80] M. Essaid, L. Idoumghar, J. Lepagnot, M. Brévilliers, GPU parallelization strategies for metaheuristics: a survey. International Journal of Parallel, Emergent and Distributed Systems **34**, pp. 497-522, 2019.

[81] B.I. Koh, A.D. George, R.T. Haftka, B.J. Fregly, Parallel asynchronous particle swarm optimization. Int. J. Numer. Meth. Engng. **67**, pp. 578-595, 2006.

[82] G. S. Tewolde, D. M. Hanna, R. E. Haskell, Multi-swarm parallel PSO: Hardware implementation, In: 2009 IEEE Swarm Intelligence Symposium, Nashville, TN, USA, 2009, pp. 60-66, 2009.

[83] A. Ouyang, Z. Tang, X. Zhou, Y. Xu, G. Pan, K. Li, Parallel hybrid PSO with CUDA for lD heat conduction equation, Computers & Fluids **110**, pp. 198-210, 2015.

[84] A. de Campos, A.T.R. Pozo, E.P. Duarte, Parallel multi-swarm PSO strategies for solving many objective optimization problems, Journal of Parallel and Distributed Computing **126**, pp. 13-33, 2019.

[85] S. Lalwani, H. Sharma, S.C. Satapathy, K. Deep, J.C. Bansal, A survey on parallel particle swarm optimization algorithms. Arabian Journal for Science and Engineering **44**, pp. 2899-2923, 2019.

[86] R.C. Eberhart, Y.H. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Congress on Evolutionary Computation, Korea, 2001.

[87] V. Charilogis, I.G. Tsoulos, Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions, Information **13**, 217, 2022.

[88] M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.

[89] M. Montaz Ali, Charoenchai Khompatraporn, Zelda 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.

[90] 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.

[91] M.M. Ali and P. Kaelo, Improved particle swarm algorithms for global optimization, Applied Mathematics and Computation **196**, pp. 578-593, 2008.

[92] H. Koyuncu, R. Ceylan, A PSO based approach: Scout particle swarm algorithm for continuous global optimization problems, Journal of Computational Design and Engineering **6**, pp. 129–142, 2019.

[93] Patrick Siarry, Gérard Berthiau, François Durdin, Jacques Haussy, ACM Transactions on Mathematical Software **23**, pp 209–228, 1997.

[94] I.G. Tsoulos, I.E. Lagaris, GenMin: An enhanced genetic algorithm for global optimization, Computer Physics Communications **178,** pp. 843-851, 2008.

[95] 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.

[96] J.E. Lennard-Jones, On the Determination of Molecular Fields, Proc. R. Soc. Lond. A **106**, pp. 463–477, 1924.

[97] R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.

[98] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[99] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.

[100] M. Raymer, T.E. Doom, L.A. Kuhn, W.F. Punch, Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, **33** , pp. 802-813, 2003.

[101] P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.

[102] M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman, L.O. Ramig, Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Trans Biomed Eng. **56**, pp. 1015-1022, 2009.