Article

New ideas in parallel Particle Swarm Optimization

Vasileios Charilogis[2], Ioannis G. Tsoulos[1,*]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece;itsoulos@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilogis@uoi.gr
* Correspondence: itsoulos@uoi.gr;

Abstract: In global optimization there are techniques where they find the optimal solutions of the objective problems, but they waste a lot of computational time. The PSO parallelization technique proposed in this paper significantly reduces the computation time and at the same time participates in the solution finding algorithm by iterative communication between the parallel computing units. Apart from the sequential algorithm, the communication strategies 1to1, 1toN, Nto1 and NtoN are compared where each computing unit sends its knowledge to the other clusters. In addition, a new and more appropriate termination rule is proposed here. From the results of the experiments, it appears that the overall parallelization technique is more than an accelerator of the classical algorithm.

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

Citation: Tsoulos, I.G.; Charilogis, V. New ideas in parallel Particle Swarm Optimization. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

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. Many problems faced by researchers can be formulated as global minimization problems such as problems in physical science [1–3], chemistry [4–6], economics [7,8] and medicine [9,10]. The global optimization methods are usually divided into two major categories: deterministic and stochastic methods. In the deterministic category, the most common method is the so - called interval method [11,12], where the set S is iteratively divided into subregions and those that do not contain a global solution are discarded using certain criteria. In the case of stochastic methods, the finding of the global minimum is based on randomness operations, although there is no guarantee of locating the global minimum. Nevertheless, it is the category of methods that is often used due to the simplicity and the effectiveness provided. Several researchers have proposed stochastic methods such as: controlled random search methods[13–15], simulated annealing methods [16–18], differential evolution methods [19,20], particle swarm optimization methods [21–23], Ant Colony Optimization [24,25], Genetic Algorithms [26–28], etc. Also, recently, many studies have appeared that utilize the modern parallel processing units [29–31] to tackle the global optimization problem. Some research that one can study regarding metaheuristic algorithms is presented in some recent papers [32–34]. 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 electrical engineer Russell C. Eberhart and social psychologist James Kennedy, observed the behavior of birds looking for food, presented a technique where the atoms or otherwise "particles" fly through the search space seeking for the best position that minimizes or maximizes a problem. These particles have two basic characteristics: their position at any instant of time, which is referred to as $\overrightarrow{x}$ and the speed at which they are moving, which is referred to as $\overrightarrow{u}$. The purpose of this method is to move the particles iteratively and calculate their next position based on three elements: the current position, the best position they had in the past and the best position of the population. The PSO method was successfully used in a variety of scientific and practical problems in physics [35,36], chemistry [37,38], medicine [39,40], economics [41] etc. Due to its high popularity, the method has received a number of interventions in recent years, such as combination with the mutation mechanism [42–44], improved initialization of the velocity vector [45], hybrid techniques [46–48], parallel techniques [49–51], methods aim to improve the velocity calculation [52–54] etc. The method of PSO has been integrated into other optimization techniques like the work of Bogdanova et al [55] who combined Grammatical Evolution with swarm techniques like PSO [56], the work of Pan et al [57] to create a hybrid PSO method with simulated annealing. Also, Mughal et al [58] used a hybrid technique of PSO and Simulated Annealing for photovoltaic cell parameter estimation. Similarly, the work of Lin et al [59] utilized a hybrid method of PSO and Differential Evolution for numerical optimization problems. Variations of PSO that aim at the global minimum in a shorter time may include the use of a local optimization method in each iteration of the algorithm [60,61]. Of course, the above process can be extremely time consuming and, depending on the termination method used and the number of local searches performed, may require a long execution time.

The proposed method creates a number of sub-populations of particles that run independently on parallel computing units that will also be called islands. Also, a series of modifications to the original particle swarm optimization method are proposed in order to make it more efficient in parallel computing environments. These modifications 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 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

In this section the discussion will begin with the steps of the serial method as well as a general outline of the parallel technique followed. Then the basic components of the proposed process, such as the calculation of the speed, the proposed termination rule and the method of propagating points between the parallel computing units will be thoroughly analyzed.

2.1. The base algorithm

The base PSO algorithm executed in every parallel processing unit is listed 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 $\text{iter}_{\text{max}}$ as the maximum number of allowed generations.
   (d) Set randomly, the initial positions of the $m$ particles $x_1, x_2, ..., x_m$.
   (e) Initialize randomly the velocities $u_1, u_2, ..., u_m$.
   (f) For $i = 1..m$ do $p_i = x_i$. The vector $p_i$ stands for 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 . If the termination criteria are hold, then terminate.
3. For $i = 1..m$ Do
   (a) Compute the velocity $u_i$ using $u_i,\ p_i$ and $p_{\text{best}}$
   (b) Set the new position $x_i = x_i + u_i$
   (c) Calculate the $f(x_i)$ for particle $x_i$
   (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 described in Algorithm 1 calculates at every iteration the new position x_i with the following operation:

$$x_i = x_i + u_i \tag{2}$$

In most cases the new speed is a linear combination of the old speed and the best values p_i and p_{best} and it 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 constant number $c_1,\ c_2$ are in the range $[1, 2]$.
3. The variable ω is commonly called inertia and typically $\omega \in [0, 1]$. The inertia was proposed by Shi and Eberhart [21]. In the current article the same inertia calculation as proposed in [62] is used. The inertia is calculated through the following equation:

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

with r being a random number and $r \in [0, 1]$.

2.2. 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 termination rule. If the termination rule holds then goto step 9 else goto
   step 5.
9. Terminate and report the best value from all processing units.. Apply a local search
   procedure to this located value to enchance the located global minimum. . In the
   proposed algorithm a BFGS variation of Powell [63] was used a local search procedure.
```

The main aspects of the parallel algorithm are the propagation mechanism and the proposed termination rule, that is properly adjusted to the parallel computation environment. These aspects will be discussed in the following subsections.

2.3. Propagation mechanism

During the execution of the parallel algorithm and periodically, the processing units propagate their best particles (those with the lowest value in the objective function) to the remaining processing units. This dissemination can be done 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 1.

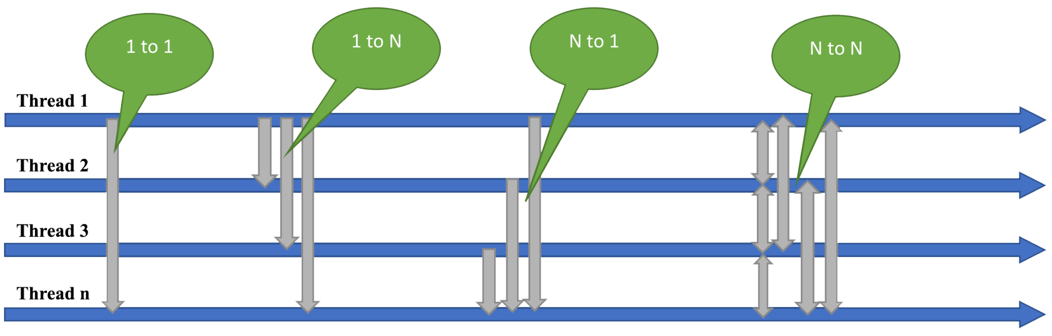

Figure 1. A graphic presentation of all propagation schemes.

2.4. Stopping rule

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

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

This quantity is calculated on every iteration k. The value $f_{i,\min}^{(k)}$ is the best located 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

To measure the reliability and efficiency of the proposed technique, experiments were performed on a wide range of objective functions from the relevant literature[64,65], which have been studied by many researchers[66–69]. In these experiments the ability of the method to find the global minimum was measured and also a study of the basic parameters of the proposed technique was made.

3.1. Test functions

The definition of the test functions used are given below

- ***Bent Cigar function*** The function is

$$f(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$$

The value $n = 10$ was used in the conducted experiments.
- **Bf1** (Bohachevsky 1) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$$

where $x \in [-100, 100]^2$.
- **Bf2** (Bohachevsky 2) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$$

where $x \in [-50, 50]^2$.
- **Branin** function: $f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ with $-5 \le x_1 \le 10,\ 0 \le x_2 \le 15$.
- **CM** function:

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

where $x \in [-1, 1]^n$. The value $n = 4$ was used in the conducted experiments.
- **Camel** function:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5, 5]^2$$

- **Discus function** The function:

$$f(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$$

The value $n = 10$ was used in the conducted experiments.

- **Easom** function:

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2-\pi)^2-(x_1-\pi)^2\right)$$

with $x \in [-100, 100]^2$.

- **Exponential** function, defined as:

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \le x_i \le 1$$

The values $n = 4, 16, 64$ were used in the executed experiments.

- **Griewank2** function:

$$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}, \quad x \in [-100, 100]^2$$

- **Gkls** function. $f(x) = \text{Gkls}(x, n, w)$, is a function with w local minima, described in [70] with $x \in [-1, 1]^n$ and n a positive integer between 2 and 100. The value of the global minimum is -1 and in our experiments we have used $n = 2, 3$ and $w = 50,\ 100$.
- **Hansen** function: $f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]$, $x \in [-10, 10]^2$.
- **Hartman 3** function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$$

with $x \in [0, 1]^3$ and $a = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}$$

- **Hartman 6** function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$$

with $x \in [0, 1]^6$ and $a = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

- **Potential** function. The molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard-Jones potential[71] is used a test function here and it is defined by:

$$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{6}$$

  For our experiments we used: $N = 3,\ 5$

- **Rastrigin** function.

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1,1]^2$$

- ***Rosenbrock*** *function*.

$$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \le x_i \le 30.$$

  In our experiments we used the values $n = 4,\ 8$.

- **Shekel 7** function.

$$f(x) = -\sum_{i=1}^{7}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

  with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}$.

- **Shekel 5** function.

$$f(x) = -\sum_{i=1}^{5}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

  with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$.

- **Shekel 10** function.

$$f(x) = -\sum_{i=1}^{10}\frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

  with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$.

- **Sinusoidal** function:

$$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right), \quad 0 \leq x_i \leq \pi.$$

  The values of $n = 4, 8$ and $z = \frac{\pi}{6}$ was used in the experimental results.
- **Test2N** function:

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5, 5].$$

  The function has 2^n in the specified range and in our experiments we used $n = 4, 5, 6, 7$.
- **Test30N** function:

$$f(x) = \frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i - 1)^2\left(1 + \sin^2(3\pi x_{i+1})\right)\right) + (x_n - 1)^2\left(1 + \sin^2(2\pi x_n)\right)$$

  with $x \in [-10, 10]$, with 30^n local minima in the search space. For our experiments we used $n = 3, 4$.

3.2. Experimental results

The proposed method was tested on the previously mentioned test functions. Every experiment mentioned was executed 30 times and the average number of function calls was reported. The most critical parameters of the proposed method are listed in table 1.

Table 1. The values for most critical parameters of the algorithm.

PARAMETER	VALUE
m	200
iter_{max}	200
c_1	1.0
c_2	1.0
N_R	15
ϵ	10^{-6}
N_M	15

The method was compared against a genetic algorithm and a simple PSO method with the same set of parameters (chromosomes and particles) and the results are reported in Table 2.

Table 2. 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	PROPOSED with $N_I = 1$
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 values for 30 independent runs. Also, the numbers in parentheses represent 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. From the experimental results, it is evident that the proposed technique significantly reduces the required number of function calls even if it is executed on a single processing unit.

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 3. Also, a box plot for this experiment is shown in Figure 2. In order to have reliability in the measurements, the total number of particles remained constant as the number of units increased. So, for example, in the case of the two computing units, in each unit the particles were 100 while in 5 units the particles were 40. In this way, the total number of particles used remains constant at 200.

Table 3. 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 varyied 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)**

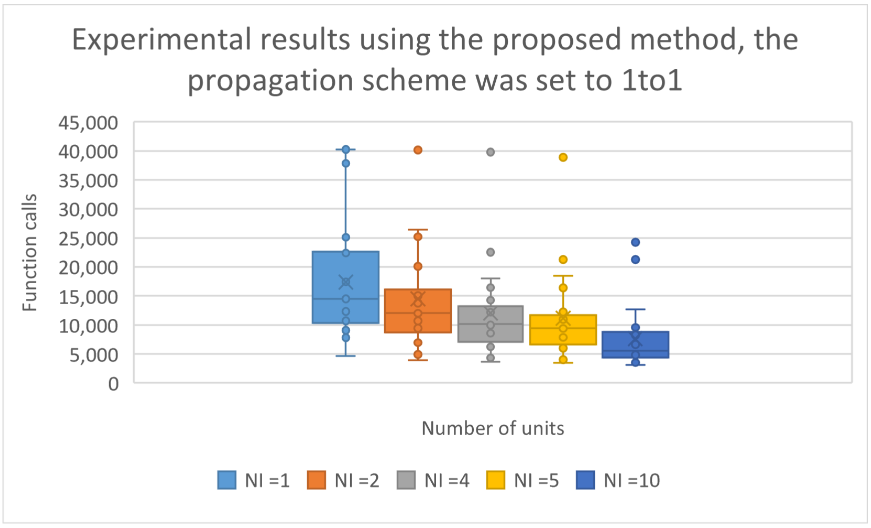

Figure 2. Box - plot for the comparison between different number of processing units. The propagation method was set to 1to1.

From this experiment, it is clear that the proposed technique drastically reduces the required number of function calls as the parallel processing units increase, while at the same time the average reliability of the method in finding the global minimum also increases.

In addition, 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 4.

Table 4. 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)**

From the experimental results, 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 5.

Table 5. 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**

Increasing the value of the parameter from 1 to 5 drastically reduces the required number of function calls, and this remains almost constant for increasing the value for that parameter.

4. Conclusions

In this paper, a number of new ideas for parallel implementation of the well-established particle optimization method were presented. In the new method, 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. 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. In addition, the increase in parallel processing units improved to some extent the efficiency of the method in finding the global minimum.

In the future, more and more effective termination techniques than the proposed one should be developed and possibly better techniques for propagating the best particles among computing units.

Author Contributions: I.G.T. and V.C. conceived of the idea, the methodology implemented the software. I.G.T. conducted the experiments, employing datasets, and provided the comparative experiments. V.C. performed the statistical analysis and prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

Acknowledgments: The experiments of this research work were performed at the high performance computing system established at Knowledge and Intelligent Computing Laboratory, Department of Informatics and Telecommunications, University of Ioannina, acquired with the project "Educational Laboratory equipment of TEI of Epirus" with MIS 5007094 funded by the Operational Programme "Epirus" 2014–2020, by ERDF and national funds.

Conflicts of Interest: The authors declare no conflict of interest.

Sample Availability: Not applicable.

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. W. L. Price, Global optimization by controlled random search, Journal of Optimization Theory and Applications **40**, pp. 333-348, 1983.
14. Ivan Křivý, Josef Tvrdík, The controlled random search algorithm in optimizing regression models, Computational Statistics & Data Analysis **20**, pp. 229-234, 1995.
15. 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.
16. S. Kirkpatrick, CD Gelatt, , MP Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.
17. L. Ingber, Very fast simulated re-annealing, Mathematical and Computer Modelling **12**, pp. 967-973, 1989.

18. R.W. Eglese, Simulated annealing: A tool for operational research, Simulated annealing: A tool for operational research **46**, pp. 271-281, 1990.
19. 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.
20. J. Liu, J. Lampinen, A Fuzzy Adaptive Differential Evolution Algorithm. Soft Comput **9**, pp.448–462, 2005.
21. 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.
22. Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, Swarm Intelligence **1**, pp 33-57, 2007.
23. Ioan Cristian Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters **85**, pp. 317-325, 2003.
24. M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, IEEE Computational Intelligence Magazine **1**, pp. 28-39, 2006.
25. K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operational Research 185, pp. 1155-1173, 2008.
26. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.
27. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.
28. S.A. Grady, M.Y. Hussaini, M.M. Abdullah, Placement of wind turbines using genetic algorithms, Renewable Energy **30**, pp. 259-270, 2005.
29. Y. Zhou and Y. Tan, "GPU-based parallel particle swarm optimization," 2009 IEEE Congress on Evolutionary Computation, pp. 1493-1500, 2009.
30. 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.
31. Barkalov, K., Gergel, V. Parallel global optimization on GPU. J Glob Optim 66, 3–20 (2016).
32. I. BoussaïD, J. Lepagnot, P. Siarry, P., A survey on optimization metaheuristics. Information sciences **237**, pp. 82-117, 2013.
33. T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering **137**, 106040, 2019.
34. K. Hussain, M.N.M. Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey.Artificial Intelligence Review **52**, pp. 2191-2233, 2019.
35. 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.
36. Ranjit Shaw, Shalivahan Srivastava, Particle swarm optimization: A new tool to invert geophysical data, Geophysics **72**, 2007.
37. 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.
38. 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.
39. 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.
40. Yannis Marinakis. Magdalene Marinaki, Georgios Dounias, Particle swarm optimization for pap-smear diagnosis, Expert Systems with Applications **35**, pp. 1645-1656, 2008.
41. 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.
42. A. Stacey, M. Jancic, I. Grundy, Particle swarm optimization with mutation, In: 2003 Congress on Evolutionary Computation, 2003. CEC '03., pp. 1425-1430, 2003.
43. 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.
44. 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.
45. A. Engelbrecht, "Particle swarm optimization: Velocity initialization," 2012 IEEE Congress on Evolutionary Computation, pp. 1-8, 2012.
46. 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.
47. 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.
48. Harish Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Applied Mathematics and Computation **274**, pp. 292-305, 2016.
49. 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.

50. 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.
51. 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.
52. 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.
53. 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.
54. Y. Jiang, T. Hu, C. Huang, X. Wu, An improved particle swarm optimization algorithm, Applied Mathematics and Computation **193**, pp. 231-239, 2007.
55. 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.
56. M. O'Neill, C. Ryan, Grammatical evolution,IEEE Transactions on Evolutionary Computation **5**, pp. 349-358, 2001.
57. X. Pan, L. Xue, Y. Lu et al, Hybrid particle swarm optimization with simulated annealing, Multimed Tools Appl **78**, pp. 29921–29936, 2019.
58. M.A. Mughal, Q. Ma, C. Xiao, Photovoltaic Cell Parameter Estimation Using Hybrid Particle Swarm Optimization and Simulated Annealing, Energies **10**, 2017.
59. 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.
60. 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.
61. 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.
62. V. Charilogis, I.G. Tsoulos, Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions, Information **13**, 217, 2022.
63. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
64. 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.
65. 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.
66. M.M. Ali and P. Kaelo, Improved particle swarm algorithms for global optimization, Applied Mathematics and Computation **196**, pp. 578-593, 2008.
67. 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.
68. Patrick Siarry, Gérard Berthiau, François Durdin, Jacques Haussy, ACM Transactions on Mathematical Software **23**, pp 209–228, 1997.
69. I.G. Tsoulos, I.E. Lagaris, GenMin: An enhanced genetic algorithm for global optimization, Computer Physics Communications **178,** pp. 843-851, 2008.
70. 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.
71. J.E. Lennard-Jones, On the Determination of Molecular Fields, Proc. R. Soc. Lond. A **106**, pp. 463–477, 1924.