Survey on Particle Swarm Optimization accelerated on GPGPU

Joanna Kołodziejczyk

Abstract— The paper presents an overview of recent research on the Particle Swarm Optimization (PSO) algorithm parallelization on the Graphics Processing Unit for general-purpose computations (GPGPU). This survey attempts to collect, organize, and present reports in the area published since 2007 in a unified way. In order to organize the literature a classification by objective functions and PSO variants is proposed. The paper also compares experimental results taking into account the most popular factor, the calculating acceleration ratio called speedup. Results of the survey are given in a very compact and comprehensive way and could be used as a guide in this area. As a summary, conclusions from categorization, a comparability problem, and possible research areas are discussed.

Index Terms—General-Purpose computing on Graphics Processor Units, NVDIA CUDA, Particle Swarm Optimization

1 INTRODUCTION

THE Particle Swarm Optimization algorithm is a popular tool for continuous domains exploration presented for the first time in [1]. The main PSO attributes are: 1) it finds a satisfactory solution for complex and large-scale problems 2) it converges fast 3) it is easy to implement 4) the number of adjustable factors is relatively small. The major problem with the practical PSO implementation is its runtime especially in multidimensional optimization tasks.

One of the most promising choices to speed up the computational process is the use of parallel implementations. All algorithms based on the population/swarm are ideally suited for parallelization, including PSO. Starting in 2001 developers can use GPUs, which are high-performance parallel accelerators. A PC equipped with a programmable graphics unit can be perceived as a dual processors device, where depending on the calculations, tasks can be split between GPU and CPU.

Due to the wide availability, programmability, and high-performance of consumer level GPUs, NVIDIA corporation invented the Compute Unified Device Architecture (CUDA) platform and implemented it on GPUs they produce. This programming model becomes very popular because it eases the GPUs code development. The CUDA platform allows writing GPU code in C functions called kernels. Many GPU threads in a Single-Instruction-Multiple-Thread (SIMT) fashion execute each kernel. Each thread executes the entire kernel once [2].

GPGPU popularity as a platform for parallel implementation of population based meta-heuristic optimization methods resulted in two publications presenting a summary of recent results in the area. Kromer et al. [3] presented a general description of twenty-three GPGPU PSO implementations from the CUDA programming point of view. A summary of optimization problems, data organization and most interesting results and problems were given. The second report by Kromer et al. [4] provides a brief overview of the latest state-of-the-art research on the design, implementation, and applications of parallel GA, DE, PSO, and SA-based methods on GPUs. The authors shortly described all presented meta-heuristics and gave a detailed description of the parallel CUDA programming model. They described eighteen PSO GPGPU implementations between 2012 and 2014, giving information about: the application area, the most important results and when possible the graphic card used. Both Kromer et al. surveys lack a method for literature classification or organization.

The objective of this paper is to collect, organize and present publications on GPGPU PSO implementations. In order to organize the growing amount of literature in this field, the paper presents a categorization of the different types of GPU PSO implementations. Categories come from the implementation diversity (standard benchmark functions or real-world optimization problems) and concern PSO algorithm variants. Other attributes, which helped in the papers' organization, were chosen in order to compare experimental results (runtime, speedup ratio, and effectiveness in the optimum discovery).

This paper is organized as follows. The next section is a brief introduction to the particle swarm algorithm and indicates categories coming from its different variants. Section 3 describes objective functions applied in the literature. Section 4 presents emerged categories used in the paper classification. Section 5 shows the literature analysis and discussion. The conclusions describe the comparability problem and further research areas.

2 PSO ALGORITHM VARIANTS

This section briefly describes the PSO algorithm in his standard version. Subsections present different PSO variants distinguished based on the velocity update rule, neighborhood and number of swarms. PSO variations will be used as categories in the literature organization.

• *Joanna Kołodziejczyk is currently assistant professor at Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, Szczecin, Poland. E-mail: jkolodziejczyk@wi.zut.edu.pl*

2.1 Standard PSO

The main inspiration for PSO was the social behavior of biological organisms seeking for food. In the PSO classic algorithm particles move through the search space and they are attracted by the best particle in the swarm and the best solution they individually have found in order to find the optimum [5], [6].

The optimization problem solved by PSO in continuous domain is to find the minimum value in function $f: \mathcal{R}^D \to \mathcal{R}$, which is the objective function, or cost function, of an application problem and D is the problem dimensionality:

$$\text{minimize } f(\vec{x}) \quad (1)$$

The vector $\vec{x}$ contains the problem's decision variables. Although (1) is considered an unconstrained optimization problem, in practice only solutions belonging to a subset of $\mathcal{R}^n$ are considered:

$$\Omega = [x_1^L, x_1^U] \times [x_2^L, x_2^U] \times \cdots \times [x_D^L, x_D^U] \quad (2)$$

where: x_d^L is the lower and x_d^U the upper bound of the search space among dimensions $d = 1,2, \ldots, D$.

The PSO algorithm works on the particle's population of size s. Each individual particle i is a potential solution to an optimization problem and is given by the position vector $\vec{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where i = 1,2, ... s. The swarm is initialized by random positions drawn from a uniform distribution within the search space Ω. Each particle keeps a memory of its own best position, it individually has found, called personal best $\vec{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. This position is only updated when the particle's new position at step t yields a better function value than the previous personal best in step t − 1:

$$\vec{p}_i(t) = \begin{cases} \vec{x}_i(t) & \text{if } f(\vec{x}_i(t)) < \vec{p}_i(t-1) \\ \vec{p}_i(t-1) & \text{otherwise} \end{cases} \quad \text{(Error! Bookmark not defined.3)}$$

The *global best* position is the position with the smallest fitness value of all positions in the neighborhood in current step t:

$$\vec{g} = \underset{\vec{p}_i \in P}{\operatorname{argmin}} f(\vec{p}_i), \quad \text{(Error! Bookmark not defined.4)}$$

where P is the set of personal best vectors from the given neighborhood.

Particle i moves from its current position to a new one along velocity vector $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, using adjustingjutingthe position update equation:

$$\vec{x}_i = \vec{x}_i + \vec{v}_i \quad \text{(Error! Bookmark not defined.5)}$$

The velocity is first updated as:

$$\vec{v}_i = \vec{v}_i + \varphi_1 \vec{r}_1 \circ (\vec{p}_i - \vec{x}_i) + \varphi_2 \vec{r}_2 \circ (\vec{g}_i - \vec{x}_i) \quad \text{(Error! Bookmark not defined.6)}$$

where operator ∘ denotes a Hardmard product and

- $\vec{v}_i$ denotes the velocity vector of particle i
- $\vec{x}_i$ denotes the position vector of particle i
- φ_1 is the cognitive acceleration coefficient
- φ_2 is the social acceleration coefficient
- $\vec{p}_i$ denotes the personal best position vector of particle i
- $\vec{g}$ is the best position vector found in the entire neighborhood
- $\vec{r}_1$ and $\vec{r}_2$ are vectors with pseudo-random numbers selected from a uniform distribution $U(0,1)$ at every update.

Each particle's velocity is randomly initialized to lie within $[v_d^{min}, v_d^{max}]$ in every dimension d . This velocity clamping allows particles to step through the same maximum percentage of the search space. Without this, particles were prone to shift outside Ω. The update process is presented as the Algorithm 1 [6].

Algorithm 1. Basic Particle Swarm Optimization

Initialize randomly $\vec{x}_i$ and $\vec{v}_i$
for each step t **do**
  for each particle i = 1,2, ... s **do**
    Evaluate particle fitness f($\vec{x}_i$)
    Update personal best $\vec{p}_i$
    Update global best in the neighborhood $\vec{g}_i$
  end for
  for each particle i = 1,2, ... s **do**
    Update position $\vec{x}_i$ using equation (5) and (6)
  end for
end for

The algorithm can be allowed to run either for a number of iterations expected to produce a good solution or until a user-specified criterion or a threshold is reached.

2.2 Velocity update

PSO can be distinguished based on differences in the velocity update rule (equation (6)).

The PSO with an *inertia weight* (w) is a method of adjusting the previous particle velocities to the optimization process:

$$\vec{v}_i = w\vec{v}_i + \varphi_1 \vec{r}_1 \circ (\vec{p}_i - \vec{x}_i) + \varphi_2 \vec{r}_2 \circ (\vec{g}_i - \vec{x}_i). \quad \text{(Error! Bookmark not defined.7)}$$

The inertia weight can be static or can be changed dynamically. When w is well adjusted the swarm has a grater tendency to constrict in the area containing best fitness and explore this area in detail.

A canonical PSO is another popular rule [5], [8] where the velocity is update as follows:

$$\vec{v}_i = \chi(\vec{v}_i + \varphi_1 \vec{r}_1 \circ (\vec{p}_i - \vec{x}_i) + \varphi_2 \vec{r}_2 \circ (\vec{g}_i - \vec{x}_i)). \quad \text{(Error! Bookmark not defined.8)}$$

χ is known as a constriction factor and is derived from the existing cognitive and social coefficients:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \quad \varphi = \varphi_1 + \varphi_2. \qquad \textbf{(Error! Bookmark not defined.9)}$$

The constriction factor balances global and local searches. It was found that when $\varphi > 4$ the swarm moves quickly and converges to the best found position in the search space.

Besides the three presented velocity update rules there are many other modifications. Some of them will be mentioned further in the paper when reports from their application will be discussed. Most of those variations were presented once in the entire collection. A single occurrence in the literature is not sufficient to design a category because categorization ought to introduce a generalization.

When the velocity update rule is the category/class in the designed reports organization, three attributes are distinguished: 1) standard PSO, 2) PSO with the inertia weight, and 3) canonical PSO.

2.3 Neighborhood topology

A neighborhood in PSO is the subset of particles in which each particle is able to communicate with each other, in order to determine the best particle denoted as $\vec{g}_i$ [7], [8].

Gbest model or global topology is defined as a neighborhood topology composed of the entire population. In this model the P vector from equation (4) is composed of all personal bests in the swarm $P = \{\vec{p}_1, \vec{p}_2, \dots, \vec{p}_s\}$. This topology is also known as a star because each particle is connected to all particles in the swarm (Fig. 1).

Lbest model or local topology is a neighborhood topology comprising some number of adjacent neighbors in the population. One of the most popular local topology is the ring model (Figure 1), where the P vector from equation (4) is composed of previous, the particle and the next particles personal bests $P = \{\vec{p}_{i-1}, \vec{p}_i, \vec{p}_{i+1}\}$.

In a global neighborhood, information is constantly distributed to all particles. When solving some optimization problems this resulted in quick attraction to the same region in the search space. Local topologies were used to prevent the PSO from stacking in a local optimum.

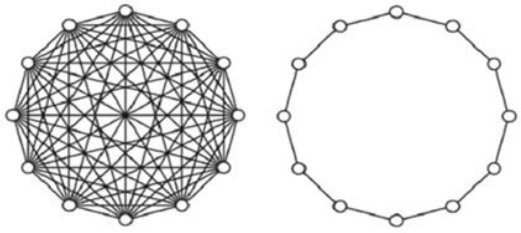

Fig. 1 The star (left) and ring (right) topology [5]

Whenever the neighborhood (difference in particle connections) is the category/class in the designed reports organization, two attributes are distinguished: 1) gbest and 2) lbest.

2.4 Multi-swarms PSO

Standard PSO is a one-population algorithm. A common procedure in all optimization heuristic methods is population multiplication. The GPU parallelism encourages multi-swarm models, but they must solve the swarms' communication problem.

In this paper, the author made an assumption to avoid a more detailed categorization than distinguishing one and multi-swarm PSOs. The argument behind this decision is that multi-swarms' implementations mainly change data structures. The data structure manipulation is connected closely to neither the objective function nor PSO variants, which were chosen by the author to perform classification. The data structure is a matter of parallel implementation i.e. CUDA kernels and threads coding. The PSO parallelization on GPUs is a very interesting but also a broad topic. If included into this survey, it will make classification complex and vague.

When the number of swarms is the category/class in the designed reports organization, two attributes are distinguished: 1) one and 2) multi.

2.5 Synchronous and asynchronous PSO

In the Algorithm 1. all particles' personal bests and global bests within their neighborhood are updated first. Then the particles are moved. These are called *synchronous* updates as opposed to *asynchronous* updates, where once the personal best is updated the particle is immediately moved (Algorithm 2).

Algorithm 2. Asynchronous PSO

```
Initialize randomly x_i and v_i
for each step t do
    for each particle i = 1,2, ... s do
        Evaluate particle fitness f(x_i)
        Update personal best p_i
        Update global best in the neighborhood g_i
        Update position x_i using equation (5) and (6)
    end for
end for
```

In consequence each particle can be moved in no special order and the swarm moved immediately in the area of newly found optima.

When the global best update step is the category/class in the designed reports organization, two attributes are distinguished: 1) synchronous and 2) asynchronous.

3 Objective functions

The PSO algorithm solves different optimization problems. As described in section 2.1, it could be a process of some function (the objective function) minimization in the continuous domain. Problems from discrete domains can also be solved. In this section, GPGPU PSO implementations are distinguished based on optimization problems they were applied to.

The objective function is a mathematical form of the optimization goal. Its properties determine the behavior of the PSO algorithm. Functions may be expensive or inexpensive in terms of time per function evaluation. Test functions or optimization problems have a great effect on the PSO

performance and must be considered when tuning and running the algorithm.

In many experiments presented in the literature standard test functions in continuous domain are used. Benchmark functions are intended to share interesting properties with real-life functions while being inexpensive in experimentation. These functions are divided into categories [5].

Test functions are listed in Table 1, where columns are labeled as follows:

- *F* – a short function name
- *Name* – long function name
- *Eq* - equations' locations in the literature
- *Domain*,
- *Min* - coordinates of the global minima
- *O* - value of the global optima
- *C* – function's categories: **S** - simple, unimodal problems, and **C** - highly complex multimodal problems with many local minima.

Table 1 Standard benchmarks used in the litereature from the collection under study

F	*Name*	*Eq*	*Domain*	*Min*	*O*	*C*
fSp	Sphere/Parabola	[5]	(-100,100)	0	0	S
fEl	Ellipse	[15]	(-5, 5)	0	0	S
fgRo	Generalized Rosenbrock	[5]	(-30,30)	1	0	S
fSw1.2	Schwefel 1.2, Rotated hyper-ellipsoid	[5]	(-100,100)	0	0	S
fgRa	Generalized Rastrigin	[5]	(-5.12,5.12)	0	0	H
fgGr	Generalized Griewank	[5]	(-600, 600)	0	0	H
fSw	Schwefel	[21]	(-500,500)	420	0	H
fgSw2.6	Generalized Schwefel 2.6	[5]	(-500,500)	420	0	H
fAc	Ackley	[5]	(-32,32)	0	0	H
fP8	Penalized Function P8	[5]	(-50,50)	-1	0	H
fP16	Penalized Function P16	[5]	(-50,50)	1	0	H

F – short name, Eq – reference to the equation, Min – minimum position, O – minimum function value, C – category: S – unimodal, H- multimodal

These test problems are widely used and especially designed to test different properties of optimization algorithms.

Except test functions, other benchmarks or real-world optimization problems are presented in the literature. When the objective function is the category/class in the designed reports organization two attributes are distinguished: 1) standard global optimization test functions and 2) other benchmarks and real-world optimization problems.

4 Categories

Previous sections presented possible GPGPU PSO implementations categorization based on the problem they solved and on the algorithm variation. Bringing together all previously presented classes the following classification schemata is proposed (Fig. 2). There are two categories: 1) Objective function and 2) PSO variant. PSO variant is divided into four subcategories: 1) velocity update, 2) neighborhood topology, 3) number of swarms and 4) global best update. Each category and subcategory has a set of attributes (bubbles in Fig. 2). A GPGPU PSO can be one of 48 different types. The diagram downward tracing obtains a specific PSO type. For example, the path: "standard test function ➔ inertia weight ➔ lbest ➔ one population ➔ synchronous" is one of 48 possible types.

5 Literature Organization

[3] and [4] described 23 reports on GPGPU PSO implementation and do not propose any reports organization. Unlike [3] and [4] this study demonstrates different and synthetic review. The outcome is a structured catalog in the form of three tables for anyone looking for the research summation in the area. The presented collection consists of 45 different reports on GPGPU PSO implementations. The very first publication in the area was published in 2007 and the last in 2014.

In the first publication [9] particles were mapped into

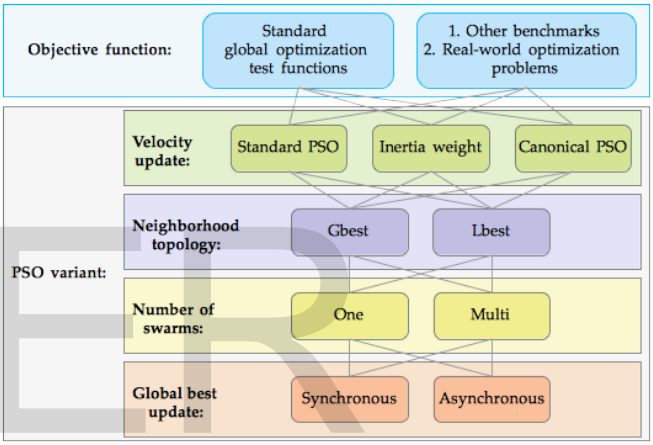

Fig. 2 A diagram of categories designed for the literature on GPGPU PSO classification

textures on a graphics card and calculated in parallel without CUDA support. This implementation differs from other implementations on CUDA and will not be further analyzed. [53] publication data are incomplete because of the restricted access to the paper and will not be analyzed as well. This reduces the total number of references in the collection to 43.

The entire collection was split into three subsets. The key to assigning to adequate subset was categories. The first subset (Table 2) contains all publications presenting PSO tested on standard benchmarks and using any of the three attributes in the 'velocity update' subcategory (Fig. 2). The second subset (Table 3) stores all reports describing PSO tested on other benchmarks and using any of the three attributes in the 'velocity update' subcategory (Fig. 2). The third subset gathers all the publications that uses different than the standard, inertia weight or canonical velocity update rules.

Summing-up, from all 43 collected papers and reports on GPGPU PSO implementation:

- 19 (44%) tested on standard benchmarks and used PSO defined variants (first subset - Table 2)
- 14 (37%) tested on other benchmarks and used PSO defined variants (second subset Table 3)
- 10 (23%) used modified velocity update rules (third

subset - Table 4).

5.1 Guidelines on table reading

Table 2 presents the following information:

1. The first author name and a reference.
2. A publication year.
3. If used, an algorithms' acronym.
4. PSO variant (V - velocity update rule, N – neighborhood topology, S – synchronization type, M – number of swarms).
5. Swarm size – number of particles used in experiments, for example, range 400-2800 means that beside 400 and 2800 some other sizes in-between were also tested.
6. Short name of standard benchmarks used in experiments. For example, '*fgRa (-10,10)*' means that the generalized Rastrigin function was tested in a domain other than given in Table 1. The word *shifted* indicates that some constant value is added to the objective function in order to move the global optimum location.
7. Benchmark dimensions used in experiments, for example,'*30, 60, 120* denotes tests on functions with 30, 60 and 120 arguments.
8. Runtime range (min-max) in seconds, for example, notation '<*1-100*' means that tests performed shorter than a second and not longer than 100 seconds. < or > symbols denote inability to present a precise value, because they were retrieved from charts.
9. Speedup (*Sup* column) (the number of times the GPGPU PSO implementation runtime was shorter than sequential PSO runtime) range.
10. The function name and conditions if the global optimum was found. For example, *fSp (D<100)* denotes that the global optimum was found in Sphere function but only if it had less than 100 arguments. If is only the name given e.g. *fAc*, then the global optimum was every time found.
11. Graphic card used in experiments.

In Table 3 columns from '*Reference*' to '*Swarm size*' include the same data as in Table 2. The column titled '*Objective function*' describes the optimization goal. The next column called *'problem description'* describes the optimization problem that was tested with GPGPU during experiments. In many cases, PSO is only an element of some complex system. In the collection cited in Table 2 most parallel implementations were compared to a sequential PSO and then speedup ratio in the '*Sup*' column was reported. There was only one exception – [37] – where runtime is given instead. In the last column, the graphic card name is presented.

Table 4 collects reports, which do not match the designed categorization. It contains reports presenting rare or new ideas of the velocity update rule modifications and three publications on PSO applied in the discrete domain. The '*variant*' column describes velocity rule modifications. The '*name*' column presents the algorithm's name. The four next columns contain the same information as in Table 3.

5.2 General information

The beginning of CUDA usage in PSO parallelization (year 2009) abounded in standard benchmark testing (5 from 6 reports). The main goal was to demonstrate acceleration and all experiments confirmed the speed up. Disparities in speedup values (from 1 to 270) are surprising. Experiments show e.g. [10], [12] that the speedup depends proportionally on the dimensions and swarm sizes. [10] and [20] show that by changing swarm size, test dimensions, and graphic card without other improvements it is possible to gain a threefold speedup increase. Speedup variations are also related with the data structures, memory usage and kernels design in CUDA. The CUDA implementation details are not discussed here therefore the exact reasons for speedup differences are not known.

The peak of research activity in the subject falls in 2012 (Fig. 3). The downward trend could be a sign of ideas exhaustion. In the last three years authors focused their attention on real-world optimization problems (22 papers). While, at the same time, only six papers presented experiments on standard test functions.

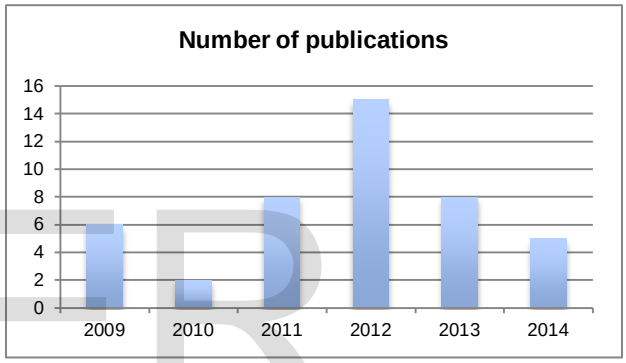

Fig. 3 Number of GPGPU PSO publication yearly

Table 2 and 3 provide statistics on PSO variants. The most popular velocity update rule is the one with the inertia weight (22 papers), followed by the canonical rule (7 papers). Global and local neighborhoods were equally often used (16 times gbest and 15 times lbest). The sequential PSO algorithm dominates with 29 occurrences. 25 papers report one-swarm PSO variant and 9 papers multi-swarms. All multi-swarm GPGPU PSOs are characterized by short runtime compared to one-swarm PSO.

5.3 Test environment

40 reports from the entire collection tested benchmarks in continuous domain, being the primary area of PSO application. The continuous domains benchmarks are better examined and discrete tests are still rare, but not missed. Only 3 papers tested benchmarks in discrete domain.

20 out of 40 publications in continuous domain presented experiments on standard test functions. The most popular were unimodal Rosenbrock (15 papers), Sphere (14 papers) and multimodal Rastrigin (16 papers) and Griewank (9 papers). The interest in standard test functions showed that it is an accepted experimental environment by the scientific community. The authors' choice on the benchmarks set, domains, dimensions and other coefficients were arbitrary. The lack of test environment unification forbade a comparison

of experimental results.

22 out of 40 publications in continuous domain presented experiments on other than standard test functions. Seven reports described PSO optimizing sampling process in the motion tracking systems. Fourteen papers presented the set of factors optimization in some complex parameterized system. [35] and [53] described PSO based classifiers.

5.4 Experimental results

Tables 2, 3 and 4 show speedups obtained in experiments. It was the most common factor used to estimate the effectives of parallelization (only [23], [37] and [49] do not report the speedup value). 9 out of 43 publications reported speedup grater than 100 times. Such high values were only reported in very specific environment conditions (number of problem dimensions, number of particles). Average speedups are few times lower. All authors underline the highest values, which is rather inadequate. To show general tendency it is More suitable to present the average acceleration. It is worth remembering, that the speedup factor expresses only the parallelization effect and does not help in comparing results especially from different optimization tests.

The experiments analysis (Table 2) raises a question if the results on standard test functions are correctly announced. It is very popular for authors to report great speedups when at the same time the objective function values are far away from the global optimum area. Of course it is a question of the main goal: if it is the runtime decreases or the optimization improvement. In the author's opinion both goals should be fulfilled at the same time. Some reports presented that the closeness to the optimum was not worse than those reached by the sequential PSO in the same test environment [14], [15], [16], [17], [23], [26], [27]. [32], [38] directly addressed the problem and showed an optimization improvement and acceleration.

6 Conclusions

This paper organizes 45 publications on GPGPU PSO implementation published since 2007 applying a comprehensive papers classification helps to sort the publications. Three tables present a synthetic literature review. They produce the handy guide for anyone looking for brief summation of recent works in the area.

The proposed collection's organization allowed statistical analysis in different categories, to generalize, and to look for dependencies in approaches and results.

One of the observations is that the common efficiency measure used by authors (93%) is the speedup. It seems reasonable to use it to compare results, but this factors only shows the parallel acceleration. Unfortunately speedup varies on different optimization problems and increases with problem dimensions and swarm size. The enormous speedup seems a success, but sometimes to make an application practical a few fold acceleration is sufficient.

The next conclusion is that using standard benchmarks is the very popular practice in population based meta-heuristic optimization methods testing. Although it is very popular and accepted by the research community the testing environment it is still hard to compare results. The problem arrives mostly from many parameters changing the test function itself and arbitrary chosen coefficients in the algorithm e.g. PSO. This is the main obstacle in comparing results. From all cited reports only [16] presented direct comparison to other results and even this seems unsatisfying. Bratton and Kenedy [5] tried to define standards in the testing environment. They proposed a list of popular test functions with their dimensions and domains, ranges for initial values and values for constant coefficients. Even authors who cited this publication did not follow those rules. The solution is to convince authors to use a unified testing environment like CUTEst or just follow some standard, for example [5].

The survey reveals new areas of research such as: other that reported tests or real-world applications (e.g. Heat Exchanger Network Synthesis), other PSO variants (e.g. Fully Informed Particle Swarm), comparison study (e.g. different PSO variants efficiency on GPU). Another suggestion is wide mutli-swarm solutions exploration because their runtime is very short. The discrete PSO parallelization was not very popular and could be recommended for survey.

The proposed organization could be expanded by additional analysis. Most publications describe the data structure and CUDA kernels code with sufficient details. Such a survey could show the relation between data structure and the results.

References

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proceedings IEEE International Conference on Neural Networks*, pp.1942–1948, Nov 1995.

[2] CUDA C Best Practices Guide, DG-05603-001 v6.0 ed., http:// nVidia.com, February 2014.

[3] P. Krömer, J. Platoš, and V. Snášel, "A Brief Survey of Advances in Particle Swarm Optimization on Graphic Processing Units," 2013 *World Congress on Nature and Biologically Inspired Computing (NaBIC)*, , vol., no., pp.182,188, 12-14 Aug. 2013

[4] P. Krömer, J. Platoš, and V. Snášel. "Nature-Inspired Meta-Heuristics on Modern GPUs: State of the Art and Brief Survey of Selected Algorithms." *International Journal of Parallel Programming*, 42.5, 681-709, 2014.

[5] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," *Swarm Intelligence Symposium*, 2007. SIS 2007. IEEE, pp.120–127, April 2007.

[6] J. Kennedy and R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[7] M. Clerc and J. Kennedy, "The Particle Swarm Explosion, Stability, and Convergence in a Multidimensional Complex Space", *IEEE Transactions on Evolutionary Computation*, 6(1): pp. 58-73, 2002.

[8] J. Kennedy and R. Mendes, "Neighborhood Topologies in Fully Informed and Best-of-Neighborhood Particle Swarms," *IEEE Transactions on Systems, Man, and Cybernetics*, Part C: Applications and Reviews, vol.36, no.4, pp.515–519, July 2006.

[9] J. Li, D. Wan, Z. Chi, and X. Hu, "An Efficient Fine-Grained Parallel Particle Swarm Optimization Method Based on GPU-Acceleration," *International Journal of Innovative Computing, Information and Control*, vol.3, no. 6(B), p.1707-1714, December 2007.

[10] Zhou and Y. Tan, "GPU-based Parallel Particle Swarm Optimization," *IEEE Congress on Evolutionary Computation*, 2009. CEC '09., pp.1493–1500, May 2009.

[11] L. de P.Veronese and R. Krohling, "Swarm's Flight: Accelerating the Particles Using C-CUDA," *IEEE Congress on Evolutionary Computation*, CEC '09., pp.3264–3270, May 2009.

[12] W. Wang, "Particle Swarm Optimization on GPU", *Workshop on GPU Supercomputing, Center for Quantum Science and Engineering National Taiwan University*, 2009, presentation available on website: <http:// http://cqse.ntu.edu.tw/cqse/gpu2009.html >

[13] G.A. Laguna-Sanchez and M. Olguin-Carbajal, N.C.C. and Ricardo Barren-Fernendez, and J.A. Alvarez Cedillo, "Comparative Study of Parallel Variants for a Particle Swarm Optimization Algorithm Implemented on a Multithreading GPU" *Journal of applied research and technology*, vol.7, no.3, pp.292–307, 2009.
[14] L. Mussi and S. Cagnoni, "Particle Swarm Optimization within the CUDA Architecture", *11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, 2009
[15] Y. Zhou and Y. Tan, "Particle swarm optimization with triggered mutation and its implementation based on GPU," *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, New York, NY, USA, pp.1–8, ACM, 2010.
[16] L. Mussi, F. Daolio, and S. Cagnoni, "Evaluation of Parallel Particle Swarm Optimization Algorithms within the CUDA architecture," *Information Sciences*, vol.181, no.20, Special Issue on Interpretable Fuzzy Systems, pp.4642 – 4657, 2011.

Table 2 PSO GPU implementations tested on standard benchmarks by publication date.

References	Year	Name	PSO					Experiments					GPUs
			Variant				Swarm size	Objective function		Results			
			V	N	S	M		Name from Table 1	D	Runtime[s]	Sup	Optimum found	
Zou et al. [10]	2009	*GPU-SPSO*	C	L ring	S	1	400-2800	*fSp, fgRa (-10,10), fgGr, fgRo (-10,10)*	50, 100, 150, 200	13- 370	3.8-11.4	*fSp, fgGr* (D<100)	GForce 8600GT
Veronose et al. [11]	2009	-	C	G	S	1	100-1000	*fgSw2.6, fgRa, fAc, fgGr, fP8, fP16*	100	2.7-340	5.4-22.3	?	GTX 280
Wang [12]	2009	*GPSO*	?	?	?	1	100-1mln	*fSp (-5.12,-5.12), fSp (shifted), fgRa, fAc, fgGr, fgRo (-10,10) , fSw +5 others*	100	<1 – 100	2-270	*fSp, fSp (shifted)*	Tesla C 1060
Laguna-Sanchez et al. [13]	2009	-	W	L	S	1	60-1024	*fgGr, fgRa, fgRo*	30, 60, 120	6.6 – 200	1-28	Always over 30000 generations	GForce 8600GT
Mussi et al. [14]	2009	*CUDAPSO*	St	L ring	S	1,2,3	?	*fgRa*	1-100	0.25-0.5	1-50	?	GForce 8800GT
Zhou et al. [15]	2010	*PSO-TM*	C	L ring	S	1	1024 - 8192	*fEl, fRa (-10,10), fAc, fgRo (-10,10) + 25 other*	30	1.4 – 53.7	7.2 – 25.5	*fEl, fAc, fRos*	Geforce 9800GT
Mussi et al. [16]	2011	*Ring PSO*	W	L ring	S	1	32	*fSp, fgRa, fgRo,*	1-120	0.1-21.7	8-138	*fSp* (D<100), *fgRa* (D<20), *fgRo* (D<10)	Geforce: EN8800GT, GTX260AMP; Quadro FX5800
		SyncPSO	W	L ring	A	1-112	32, 64, 128	*fgRa*	1-9	0.02-0.35	7-30	?	Geforce EN8800GT
Mussi et al. [17]	2011	-	W	L ring	A	1-112	27, 32	*fSp, fSw1.2, fgRo, fgRa, fgGr*	1-120	0.02-0.35	2-250	*fSp, fSw1.2* (D<1120), *fgRa* (D<20), *fgRo* (D<10), *fgGr* (D<5)	Geforce GTX260AMP Geforce GTS450
Cardenas-Montes et al. [18]	2011	-	?	?	?	1	20	*fSw1.2*	1000-15000	4.5-130.9	1.8-20.7	No	GeForce GTX295
Cardenas-Montes et al. [19]	2011	-	St	G	S	1	20	*fSw1.2*	20000	?	20.4-26.7	?	GTX 295
											43.3-43.8	?	TESLA C2050
Zhou et al. [20]	2011	*GPU-PSO*	C	L ring	S	1	512-1280	*fSp, fgRa (-10,10), fgGr, fgRo (-10,10)*	50-200 1000, 2000	1.17-128.2	2.7 – 39.7	*fSp* (D=50), *fgGr* (D=50,100)	NVIDIA Geforce 9800GT
Hung et al. [21]	2012	*GPSO*	W	G	S	1	16 - 1048576	*fSp (-5.12,-5.12), fSp (shifted), fgRa, fAc, fgGr, fgRo (-10,10) , fSw +5 others*	100	<1 – 100	1-270	*fSp, fSp (shifted), fAc, fgGr*	Tesla C1060
Cagnoni et al. [22]	2012	-	W	L ring	S	1	32-8192	*fSp, fgRa, fSw1.2 , fgRo, fgGr*	32, 64, 128	0.1-10	1-6	?	GT-540M GTX-560Ti
Calazan et al. [23]	2012	-	W	G	S	7-224	32-1024	*fSp, fgGr, fRa (-10,10)*	30	0.38-1.13	?	*fSp , fGri, fRas* (error < 0.011)	GeForce GTX 460
Roberge et al. [24]	2012	*CUDA-PSO*	W	G	S	1	256-16384	*normalized fgRo*	20	?	20-256	?	GTXS60Ti
Roberge et al. [25]	2012	*parallel PSO*	W	G	S	1	256-16384	*normalized fgRo*	20	0.1-100	20-215	*?*	GTX560Ti
Calazan et al. [26]	2013	*PDPSO*	W	L ring	S	1	6-1024	*fSp, fSw, fgRo (-16,16),*	2-256	0.05-9	1-81.5	*fSp, fSw,* (both error < 0.0001) *fgRo* (D<256)	GeForce GTX 460
Calazan et al. [27]	2013	*CPPSO*	W	L ring	S	2,4	4-256	*fSp, fSw, fgRo (-16,16), fgRa*	2-256	0.02->1	1-81.5	*fSp , fSw* (both error < 0.0001) *fgRo* (D<128), *fgRa* (D<16)	GeForce GTX 460
Kumar et al. [28]	2013	-	W	G	S	16-224	500,100, 1500	*Shifted: fSp, fEl, fgRa, fgRo, fAc*	1000	96-1211	9-60	*fSp, fEl, fgRa, fAc*	Tesla M-2070

Name – algorithm name, V- variants of the velocity calculation: St – standard PSO, W – PSO with the inertia weight, C – canonical PSO; N – neighborhood topology: G, L – Gbest and Lbest respectively, S –

synchronous PSO, A – asynchronous PSO, M – number of swarms, Sup –Speedup ratio; ?- no data available

Table 3 PSO GPU implementations tested on real-world optimization problems or other test functions by publication date.

		PSO					*Experiments*			
References	*Year*	*Variant*				*Swarm size*	*Objective function*	*Problem description*	*Sup*	*GPUs*
		V	*N*	*S*	*M*					
Mussi et al. [29]	2009	St	L ring	S	1	64	Weighted Bhattacharyya coefficients in 3 channels HSV color space.	Road sign detection in Advanced Driving Assistance Systems, which takes into account shape and color to detect signs. PSO estimate the pose of the sign in the 3D space and the position of the sign in the image.	15	Gforce 8800GT
Mussi et al. [30]	2010	W_t	?	S	1	10	Compares the silhouettes generated by the model with the silhouettes extracted from images.	Marker-less full-body articulated human motion tracking system from multi-view video sequences acquired in a studio environment. Detecting location, orientation and scale of each body part.	20	Quadro FX 5800
Solomon et al. [31]	2011	W	G	A	1-60	128	Max Machine Available Time (MAT): the total amount of time required by the machine to complete all tasks.	Task matching/mapping problem: composed of two distinct components: 1. The set of tasks, T, to be mapped, and, 2. The set of machines, M, which tasks can be mapped to. A discrete PSO was also tested.	32	Geforce GTX260AMP
Wachowiak et al. [32]	2012	W_t	G	S	1	500-4000	Toy protein folding function (3D), logistic function (2D), disequilibrium function (8D)	Three problems: Toy protein folding, realistic two-dimensional logistics problem: it is a maximum likelihood estimator; disequilibrium problem in econometrics concerns determining the supply and demand components of a time series of transacted quantities	298	Tesla S1070
Datta et al. [33]	2012	C	G	S	1	256-1024	Self Potential model (5D), Magnetic Model (4D), Resistivity Model (2D)	Geology – invert Self Potential (Surda Area of Jharkhand, India), Magnetic (anomaly of Boston Township) and Resistivity (Satkui) models. Optimization of model parameters.	22	NVIDIA 9200M GS
Nobile et al. [34]	2012	W	G	S	3, 4	32	Distance between the sampled in the experiment biochemical species, and a simulated dynamics from stochastic simulation algorithm.	Estimation of the stochastic constants of two simple systems: the Michaelis-Menten kinetics (2D) and a prokaryotic auto-regulatory gene network (6D)	24	Tesla C1060
Platos et al. [35]	2012	W	?	?	?	4-10240	Combination of precision and recall (classification measures)	Document classification form data-sets: Reuters-21578, Iris collection, 20 Newsgroup with different number of tokens.	10.5	Tesla C2050
Rabinovich et al. [36]	2012	St	G	S	2-?	256-28160	Optimize the sum of the priorities of the signals that fall within the placement of the three receivers/jammers (3x48D)	Radio Frequency Resource Allocation Optimizer – allocation of radio frequency resources with constraints of bandwidth and power.	5	GeForce GTX 465
Reguera-Salgado et al. [37]	2012	C	L	S	1	50	Minimum Root Mean Square Error (RMSE) of the differences between the GCPs and associated projected pixels locations. (6D)	Geocorrection – PSO is used to find the set of corrections of the navigation data that produces the best match between the projected pixels and the Ground Control Point. Used for digital airborne pushbroom images (Cies Islands) to Digital Terrain Models.	time 20s-220s	GeForce 9500GT
Roberge et al. [24]	2012	W	G	S	1	32-512	Harmonic minimization in multilevel inverters (2D).	Problem of optimal switching angles to reduce or eliminate harmonics in multilevel inverters. For some given circuit the optimal angles to control the DC sources and generate a current while minimizing harmonic.	115	GTXS60Ti
Roberge et al. [25]	2012	W	G	S	1	32-256	Distance between the virtual marker projected on the 2D image and the actual marker identified on the 2D image	High-speed camera on the aircraft records multiple bomb drops as 2D video images. Bomb's position in 3D is obtained from 2D image. Problem: 6 degrees of freedom (6DOF) (x, y, z, yaw, pitch, and roll).	140	GTX560Ti
Rymut et al. [38]	2013	W	G	S	1	100-400	An overlap between person silhouette and 3D model + image to camera edge distance (4 cameras)	Marker-less body human motion tracking system – recovery of humane pose. Combination of Particle Filter and PSO. Result: 16 frames/s	7.5	GeForce 590GTX
Zhang et al. [39]	2013	C	L ring	S	1	128	Minimize error: the sum of the differences (distances) in two models for every sampling set (31D)	Marker-less 3D articulated human motion tracking system. Searching of the optimal pose is the hybrid of important sampling (Monte Carlo method) and niching PSO. PSO resampled after M generations.	30	GeForce GTX 295
Rymut et al. [40]	2014	W	G	S	1	100-1000	An overlap between person silhouette and 3D model + image to camera edge distance (4 cameras)	Real-time body human motion tracking system – recovery of humane pose. 3D model has 26 DOF. Fitness function is decomposed. Result: 12 frames/s	12	GeForce 590GTX
Ma et al. [41]	2014	W	G	S	1	64-2560	The difference between the measured data and the calculated current is gradually minimized	Extract and estimate the parameters of a photovoltaic (PV) model. The single diode model (SDM) with five parameters: photocurrent, saturation current, diode ideality constant, series resistance, and shunt resistance, that	80 30	GTX760 9400M

								need to be estimated was used.		
Van Heerden et al. [42]	2014	W	L ring	S	1	1024	Minimum of Euclidean distance from the goal plus penalty.	Optimization of model predictive control: continuous non-linear dynamic system of the Acrobot motion control. The particles re-sampling in the area of previous best is used.	8.3	GeForce GTS 450

V- variants of the velocity calculation: St – standard PSO, W – PSO with the inertia weight, subindex t-tuned inertia, C – canonical PSO; N – neighborhood topology: G, L – Gbest and Lbest respectively, S – synchronous PSO, A – asynchronous PSO, M – number of swarms; Sup –Speedup ratio; ?- no data available

Table 4 PSO GPU implementations tested on real-world optimization problems or other test functions by publication date, PSO variants other than in Table 2.

		PSO		*Experiments*			
References	*Year*	*Variant*	*Name*	*Objective function*	*Problem description*	*Sup*	**GPUs**
Papadakis et al. [43]	2011	The gbest component is removed and particles are attracted by their personal best based on the learning probability (random value). Tournament selection chooses better correction.	Comprehensive Learning PSO (CLPSO)	Minimize sum of incremental cost function of each unit penalized by technical constraints (9-72 D).	Economic Dispatch problem (ED). ED considers power system, comprising N units. Problem: calculate the output of each unit so that the total operating cost is minimized, providing power balance and technical limit constraints.	36.6	Geforce GTX 260
Zhu et al. [44]	2011	The Euclidean interference factor is added into a inertia weight PSO. It is a sigmoid function. Its argument is Euclidean distance from a particle to gbest.	Euclidean PSO (EPSO)	*fSp*, *fgRo (-30,30)*, *fgRa*, *fgGr*, *fAc*	Standard test function for global optimization. Functions with (1000≤D≤8000) arguments were tested.	0.7-16.3	Geforce GTX 480
Chen et al. [45]	2012	Personal best attraction is a mutation by 2-exchange using personal best position. Analogously global best attracts to its position.	LaPSO (discrete)	Minimize weighted function that calculates inner Hamming distances between points in LHD.	n-run and k-factor Latin hypercube designs (LHDs) – is a method for generating samples of plausible collections of parameter values. A sample is the only one in each k-axis.	59	Tesla C2070
Sharma et al. [46]	2012	The confident factor was exchanged by personal best and global best position ratio and social ratio was exchanged by market volatility.	Normalized PSO (NPSO)	Minimize a portfolio value that is the total estimated cost of the holdings of the investor.	Financial application: option pricing based on 1) current stock price, 2) strike price, 3) expiration time, 4) rate of interest and 5) market volatility. The portfolio is composed of European and American call and put options.	45	?
Zhang et al. [47]	2012	NPSO is a multi-swarm approach. The Velocity is updated with bare bone rule. The new position is updated by random diffusion. Algorithms steps were modified.	Niching Bare Bone PSO (NPSO)	Minimize the sum of error from all body parts. Hierarchical optimization.	The body poses tracking in 3D - 3D volumetric reconstruction of the real-world dynamic scenes. Searching of the optimal pose is the hybrid of stochastical generative sampling algorithm and niching PSO.	30	GeForce GTX 295
Zhao et al. [48]	2012	The velocity update with inertia weight was modified. Beside personal best and global best positions the best-so-far in sub-swarm additionally attracts the particle.	Parallel multi-swarm PSO	Minimize LS-SVM model.	Prediction for gas holder level in the Linz Donawitz converter gas system based on least square support vector. The multiple sub-swarms (PSO) optimizes a model parameters.	65	GeForce GTX 260
Souza et al. [49]	2013	Mutation of inertia weight, global best and cognition and social coefficient is performed, and then better solutions are selected. Velocity updates use mutants instead random numbers	Cooperative Evolutionary Multi-Swarm PSO (CEMSO)	Minimizing 1) the manufacturing cost of a steel beam 2) total weight of speed reducer 3) the quantity of material	1) Welded Beam Design (WBD): (4D) 2) Speed Reducer Design with 25 restrictions (SRD-25) (7D). 3) Air Tank Design (ATD): (5D); All problems with constraints	?	GeForce GT 330M
Kilic et al. [50]	2013	A logistic transformation is used to accomplish the velocity vector. A probability of keeping 1 or exchanging to 0 is determined.	Binary PSO (BPSO) discrete	1) optimum pixel set of the impedance (32D) 2) optimize the height of each layer and the periodicity of the gratings	1) The optimization for tuning the shape of the antenna to the user-specified frequency. 2) The antireflective surface design.	10	4x Tesla C1060 graphics
Zan et al. [51]	2014	A logistic transformation is used to accomplish the velocity vector. A probability of keeping 1 or exchanging to 0 is determined.	Binary PSO (BPSO) discrete	Maximum of linear composition of profit and penalty (64-1024D)	The Multidimensional Knapsack Problem (MKP): select items from the available set to knapsacks of limited capacity.	9.6	GeForce GTX 580
Ouyang et al. [52]	2014	The velocity update with inertia weight is used. Hybrid method joining Conjugate gradient method with PSO. The best particle in PSO is replaced by best from CGM.	PHPSO	Minimize the weighted sum of problem variables	One dimensional nonclassical heat conduction equation is modified into linear equation systems then transformed into an unconstrained optimization problem, which is optimized by PSO.	21 23	GTX465 Tesla C2050

?- no data available

[17] L. Mussi, Y.S. Nashed, and S. Cagnoni, "GPU-based Asynchronous Particle Swarm Optimization," *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11*, New York, NY, USA, pp.1555–1562, ACM, 2011.

[18] M. Cardenas-Montes, M. Vega-Rodriguez, J. Rodriguez-Vazquez, and A. Gomez-Iglesias, "Accelerating Particle Swarm Algorithm with GPGPU," *2011 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp.560–564, Feb 2011.

[19] M. Cardenas-Montes, M.A. Vega-Rodríguez, J. J. Rodríguez-Vázquez, and A. Gómez-Iglesias, "Effect of the Block Occupancy in GPGPU over the Performance of Particle Swarm Algorithm." Adaptive and Natural Computing Algorithms, Springer Berlin Heidelberg, 2011. 310-319.

[20] Y. Zhou and Y. Tan, "Parallel Particle Swarm Optimization Algorithm Based on Graphic Processing Units", Handbook of Swarm Intelligence, Series Adaptation, Learning, and Optimization, Springer Berlin Heidelberg, pp. 133-154, 2011,

[21] Y. Hung and W. Wang, "Accelerating Parallel Particle Swarm Optimization via GPU", *Optimization Methods and Software*, 27:1, 33-51, 2012

[22] S. Cagnoni, A. Bacchini, and L. Mussi, "OpenCL implementation of Particle Swarm Optimization: A Comparison Between Multi-Core CPU and GPU Performances," *Proceedings of the 2012T European Conference on Applications of Evolutionary Computatio*n, Berlin, Heidelberg, pp.406–415, Springer-Verlag, 2012.

[23] R. Calazan, N. Nedjah, and L. de Macedo Mourelle, "Swarm Grid: A proposal for High Performance of Parallel Particle Swarm Optimization Using GPGPU," *Lecture Notes in Computer Science and Its Applications ICCSA*, vol.7333, pp.148–160, Springer Berlin Heidelberg, 2012.

[24] V. Roberge and M. Tarbouchi, "Efficient Parallel Particle Swarm Optimizers on GPU for Real-Time Harmonic Minimization In Multilevel Inverters," *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pp.2275,2282, 25-28 Oct. 2012

[25] V. Roberge, and M. Tarbouchi, "Parallel particle swarm optimization on graphical processing unit for pose estimation." WSEAS Transactions on Computers, 11.6 (2012): pp. 170-179, 2012

[26] R. Calazan, N. Nedjah, and L. de Macedo Mourelle, "Parallel GPU-based Implementation of High Dimension Particle Swarm Optimizations," *2013 IEEE Fourth Latin American Symposium on Circuits and Systems (LASCAS)*, pp.1–4, Feb 2013.

[27] R. Calazan, N. Nedjah, and L. de Macedo Mourelle, "A Cooperative Parallel Particle Swarm Optimization for High-Dimension Problems on GPUs," *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC*), 2013 BRICS Congress on, vol., no., pp.356,361, 8-11 Sept. 2013

[28] J. Kumar, L. Singh, S. Paul, "GPU Based Parallel Cooperative Particle Swarm Optimization Using C-CUDA: A case study," *2013 IEEE International Conference on Fuzzy Systems (FUZZ)*, pp.1-8, 7-10 July 2013

[29] L. Mussi, S. Cagnoni, E. Cardarelli, F. Daolio, P. Medici and P.P. Porta, "GPU Implementation of a Road Sign Detector Based on Particle Swarm Optimization", *Evolutionary Intelligence*, 3, 3-4, pp. 155-169, 2010, Springer

[30] L. Mussi, S. Ivekovic, and S. Cagnoni. "Markerless Articulated Human Body Tracking from Multi-View Video with GPU-PSO." *Evolvable Systems: From Biology to Hardware*. Springer Berlin Heidelberg, pp. 97-108, 2010.

[31] S. Solomon, P. Thulasiraman, and R. Thulasiram, "Collaborative Multi-Swarm PSO for Task Matching Using Graphics Processing Units," *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11*, New York, NY, USA, pp.1563– 1570, ACM, 2011.

[32] M.P. Wachowiak and A.E.L. Foster, "GPU-based Asynchronous Global Optimization With Particle Swarm," *Journal of Physics, Conference HPCS2012 Series (Vol. 385, No. 1, p. 012012)*, IOP Publishing, 2012.

[33] D. Datta, S. Mehta, and R.S. Srivastava, "CUDA Based Particle Swarm Optimization for Geophysical Inversion," *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, pp. 416-420, 15-17 March 2012

[34] M. S. Nobile, D. Besozzi, P. Cazzaniga, G. Mauri, and D. Pescini, "A GPU-based Multi-swarm PSO Method for Parameter Estimation in Stochastic Biological Systems Exploiting Discrete-time Target Series." *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, Springer Berlin Heidelberg, pp. 74-85, 2012.

[35] J. Platos, V. Snasel, T. Jezowicz, P. Kromer, and A. Abraham, "A PSO-based Document Classification Algorithm Accelerated by the CUDA Platform," *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp.1936,1941, 14-17 Oct. 2012

[36] M. Rabinovich, P. Kainga, D. Johnson, B. Shafer, J.J. Lee, R. Eberhart, "Particle Swarm Optimization on a GPU," *2012 IEEE International Conference on Electro/Information Technology (EIT)*, pp.1-6, 6-8 May 2012

[37] J. Reguera-Salgado and J. Martin-Herrero, "High Performance GCP-based Particle Swarm Optimization of Orthorectification Of Airborne Pushbroom Imagery," *2012 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp.4086-4089, 22-27 July 2012

[38] B. Rymut, B. Kwolek, and T. Krzeszowski. "GPU-Accelerated Human Motion Tracking Using Particle Filter Combined with PSO." *Lecture Notes in Computer Science, Advanced Concepts for Intelligent Vision Systems*, Springer International Publishing, vol. 8192, pp. 426-437, 2013

[39] Z. Zhang, H. S. Seah, C. K. Quah and J. Sun, "GPU-accelerated Real-Time Tracking Of Full-Body Motion With Multi-Layer Search." *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 106-119, 2013.

[40] B. Rymut, and B. Kwolek. "Real-time multiview human pose tracking using graphics processing unit-accelerated particle swarm optimization", *Concurrency and Computation: Practice and Experience*, DOI: 10.1002/cpe.3329, 2014

[41] J. Ma; K.L. Man; T.O. Ting, N. Zhang; S. Guana and P.W.H Wong, "Accelerating Parameter Estimation for Photovoltaic Models via Parallel Particle Swarm Optimization," *2014 International Symposium on Computer, Consumer and Control (IS3C)*, pp.175,178, 10-12 June 2014

[42] K. Van Heerden, Y. Fujimoto, and A. Kawamura, "A Combination of Particle Swarm Optimization and Model Predictive Control on Graphics Hardware for Real-Time Trajectory Planning of the Under-Actuated Nonlinear Acrobot," *IEEE 13th International Workshop on Advanced Motion Control (AMC)*, pp. 464-469, 14-16 March 2014

[43] S.E. Papadakis, and A.G. Bakrtzis, "A GPU Accelerated PSO with Application to Economic Dispatch Problem," *2011 16th International Conference on Intelligent System Application to Power Systems (ISAP)*, pp.1,6, 25-28 Sept. 2011

[44] H. Zhu, Y. Guo, J. Wu; J. Gu and K. Eguchi, "Paralleling Euclidean Particle Swarm Optimization in CUDA," *2011 4th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pp.93,96, 1-3 Nov. 2011

[45] R.B. Chen, D. Hsieh, Y. Hung, and W. Wang, "Optimizing Latin Hypercube Designs by Particle Swarm." *Statistics and Computing*, vol. 23, no. 5, pp. 663-676, 2012.

[46] B. Sharma, R.K. Thulasiram, and P. Thulasiraman, "Portfolio Management Using Particle Swarm Optimization on GPU," *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pp.103,110, 10-13 July 2012

[47] Z. Zhang and S.S. Hock, "CUDA Acceleration of 3D Dynamic Scene Reconstruction and 3D Motion Estimation for Motion Capture," *2012 IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS)*, pp.284,291, 17-19 Dec. 2012

[48] J. Zhao, W. Wang, W. Pedrycz, and T. Xiangwei, "Online Parameter Optimization-Based Prediction for Converter Gas System by Parallel Strategies," *IEEE Transactions on Control Systems Technology*, vol.20, no.3, pp.835,845, May 2012

[49] D. L. Souza, O. N. Teixeira, , D. C. Monteiro and R. C. L. de Oliveira, "A New Cooperative Evolutionary Multi-Swarm Optimizer Algorithm Based on CUDA Architecture Applied to Engineering Optimization.", *Combinations of Intelligent Methods and Applications*, Springer Berlin Heidelberg, pp. 95-115, 2013

[50] O. Kilic, E. El Araby, Q. Nguyen, and V. Dang, "Bio-inspired Pptimization for Electromagnetic Structure Design Using Full-Wave Techniques on GPUs. *International Journal of Numerical Modelling: Electronic Networks*, Devices and Fields, vol. 26, no. 6, pp. 649-669, 2013

[51] D. Zan, J. Jaros, "Solving the Multidimensional Knapsack Problem Using a CUDA Accelerated PSO," *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2933-2939, 6-11 July 2014

[52] A. Ouyang, Z. Tang, X. Zhou, Y. Xu, G. Pan and K. Li, "Parallel Hybrid PSO with CUDA for lD Heat Conduction Equation", *Computers & Fluids (2014)*, Available online 28 May 2014,

[53] C. Cresswell-Miley and N. Kourosh. "A Stepwise Multi-centroid Classification Learning Algorithm with GPU Implementation." *Simulated Evolution and Learning*. Springer International Publishing, pp. 347-358, 2014.