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, NVIDIA 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 its 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.
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 $f: \mathbb{R}^D \rightarrow \mathbb{R}$, which is the objective function, or cost function, of an application problem and D is the problem dimensionality:

\[
\min f(\vec{x})
\]

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 $\mathbb{R}^D$ are considered:

\[
\Omega = [x_1^L, x_1^U] \times [x_2^L, x_2^U] \times \ldots \times [x_D^L, x_D^U]
\]

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 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 $i = 1, 2, \ldots, s$. The swarm is initialized by random positions drawn from a uniform distribution within the search space $\Omega$. 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 $t - 1$:

\[
\vec{p}_i(t) = \begin{cases} 
\vec{x}_i(t) & \text{if } f(\vec{x}_i(t)) < \vec{p}_i(t - 1) \\
\vec{p}_i(t - 1) & \text{otherwise}
\end{cases}
\]

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

\[
\vec{g} = \arg \min_{\vec{p}_i \in P} f(\vec{p}_i),
\]

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 adjusting the position update equation:

\[
\vec{x}_i(t + 1) = \vec{x}_i(t) + \vec{v}_i(t + 1)
\]

The velocity is first updated as:

\[
\vec{v}_i = \vec{v}_i + \varphi_1 \vec{r}_1 \circ (\vec{p}_i - \vec{x}_i) + \varphi_2 \vec{r}_2 \circ (\vec{g} - \vec{x}_i)
\]

where operator $\circ$ denotes a Hadamard product and $\vec{p}_i$ denotes the 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 $\Omega$. 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 $i = 1, 2, \ldots, s$ 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

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:

\[
\vec{v}_i = w\vec{v}_i + \varphi_1 \vec{r}_1 \circ (\vec{p}_i - \vec{x}_i) + \varphi_2 \vec{r}_2 \circ (\vec{g}_i - \vec{x}_i).
\]

The inertia weight can be static or can be changed dynamically. When $w$ is well adjusted the swarm has a greater tendency to constrain 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:

\[
\vec{v}_i = \chi(\vec{v}_i + \varphi_1 \vec{r}_1 \circ (\vec{p}_i - \vec{x}_i) + \varphi_2 \vec{r}_2 \circ (\vec{g}_i - \vec{x}_i)).
\]

$\chi$ is known as a constriction factor and is derived from the existing cognitive and social coefficients:
\[
\chi = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}}
\]
\[
\phi = \phi_1 + \phi_2.
\]

The constriction factor balances global and local searches. It was found that when \( \phi > 4 \) 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 \( \tilde{g}_t \) [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 \( P = \{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_s\} \). 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 \( P = \{\tilde{p}_{i-1}, \tilde{p}_i, \tilde{p}_{i+1}\} \).

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]](image)

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 \( \tilde{x}_i \) and \( \bar{v}_i \)

for each step \( t \) do

for each particle \( i = 1, 2, \ldots, s \) do

Evaluate particle fitness \( f(\tilde{x}_i) \)

Update personal best \( \tilde{p}_i \)

Update global best in the neighborhood \( \tilde{g}_t \)

Update position \( \tilde{x}_i \) 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
- **Eq** - long function name
- **Domain**
- **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 literature from the collection under study

<table>
<thead>
<tr>
<th>F</th>
<th>Name</th>
<th>Eq</th>
<th>Domain</th>
<th>Min</th>
<th>O</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sp</td>
<td>Sphere/Parabola</td>
<td>[5]</td>
<td>(-100,100)</td>
<td>0</td>
<td>0</td>
<td>S</td>
</tr>
<tr>
<td>El</td>
<td>Ellipse</td>
<td>[15]</td>
<td>(-5,5)</td>
<td>0</td>
<td>0</td>
<td>S</td>
</tr>
<tr>
<td>Ra</td>
<td>Generalized Rosenbrock</td>
<td>[5]</td>
<td>(-30,30)</td>
<td>1</td>
<td>0</td>
<td>S</td>
</tr>
<tr>
<td>Sw1.2</td>
<td>Schwefel 1.2, Rotated hyper-ellipsoid</td>
<td>[5]</td>
<td>(-100,100)</td>
<td>0</td>
<td>0</td>
<td>S</td>
</tr>
<tr>
<td>Ra2</td>
<td>Generalized Rastrigin</td>
<td>[5]</td>
<td>(-5.12,5.12)</td>
<td>0</td>
<td>0</td>
<td>H</td>
</tr>
<tr>
<td>Gr</td>
<td>Generalized Griewank</td>
<td>[5]</td>
<td>(-600,600)</td>
<td>0</td>
<td>0</td>
<td>H</td>
</tr>
<tr>
<td>Se</td>
<td>Schwefel</td>
<td>[21]</td>
<td>(-500,500)</td>
<td>420</td>
<td>0</td>
<td>H</td>
</tr>
<tr>
<td>Sw6</td>
<td>Generalized Schwefel 2.6</td>
<td>[5]</td>
<td>(-500,500)</td>
<td>420</td>
<td>0</td>
<td>H</td>
</tr>
<tr>
<td>Ac</td>
<td>Ackley</td>
<td>[5]</td>
<td>(-32,32)</td>
<td>0</td>
<td>0</td>
<td>H</td>
</tr>
<tr>
<td>P8</td>
<td>Penalized Function P8</td>
<td>[5]</td>
<td>(-50,50)</td>
<td>-1</td>
<td>0</td>
<td>H</td>
</tr>
<tr>
<td>P16</td>
<td>Penalized Function P16</td>
<td>[5]</td>
<td>(-50,50)</td>
<td>1</td>
<td>0</td>
<td>H</td>
</tr>
</tbody>
</table>

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

Except function tests, 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 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
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, ‘fgrRa (-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.

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 effectiveness of parallelization (only [23], [37] and [49] do not report the speedup value). 9 out of 43 publications reported speedup greater 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 multi-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


<table>
<thead>
<tr>
<th>References</th>
<th>Year</th>
<th>Name</th>
<th>Variant</th>
<th>Swarm size</th>
<th>Objective function</th>
<th>Experiments</th>
<th>Results</th>
<th>GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zou et al. [10]</td>
<td>2009</td>
<td>GPU-SPSO</td>
<td>C L ring</td>
<td>S 1</td>
<td>400-2800</td>
<td>$f_{sp}$, $f_{Gr}$ (-10,10), $f_{G}$, $f_{Ro}$ (-10,10)</td>
<td>50, 100, 150, 200</td>
<td>13-370, 3.8-11.4, $f_{sp}$, $f_{G}$ (D=100)</td>
</tr>
<tr>
<td>Veronose et al. [11]</td>
<td>2009</td>
<td>-</td>
<td>C G ring</td>
<td>S 1</td>
<td>100-1000</td>
<td>$f_{S}or_{2.6}$, $f_{G}$, $f_{Ac}$, $f_{G}$, $f_{Ro}$ (-10,10)</td>
<td>100</td>
<td>2.7-340, 5.4-22.3, $f_{sp}$, $f_{sp}$ (shifted)</td>
</tr>
<tr>
<td>Wang [12]</td>
<td>2009</td>
<td>GFSO</td>
<td>? ? ring</td>
<td>100-1mün</td>
<td>$f_{sp}$ (-5.12-5.12), $f_{sp}$ (shifted), $f_{G}$, $f_{Ac}$, $f_{G}$, $f_{Ro}$ (-10,10), $f_{S}or_{5}$ others</td>
<td>100</td>
<td>&lt;1 - 100, 2-270, $f_{sp}$, $f_{sp}$ (shifted)</td>
<td>Tesla C 1060</td>
</tr>
<tr>
<td>Laguna-Sanchez et al. [13]</td>
<td>2009</td>
<td>-</td>
<td>W L ring</td>
<td>S 1</td>
<td>60-1024</td>
<td>$f_{G}$, $f_{G}$, $f_{S}$, $f_{Ro}$</td>
<td>30, 60, 120</td>
<td>6.6 - 200, 1-28, Always over 30000 generations</td>
</tr>
<tr>
<td>Mussi et al. [14]</td>
<td>2009</td>
<td>CUDA-PSO</td>
<td>St L ring</td>
<td>1,2,3 ring</td>
<td>7</td>
<td>$f_{S}$</td>
<td>1-100</td>
<td>0.25-0.5</td>
</tr>
<tr>
<td>Zhou et al. [15]</td>
<td>2010</td>
<td>FSO-TM</td>
<td>C L ring</td>
<td>S 1</td>
<td>1024 - 8192</td>
<td>$f_{S}$, $f_{G}$, $f_{S}$, $f_{Ro}$ (-10,10)</td>
<td>30</td>
<td>1.4 - 53.7, 7.2 - 25.5, $f_{L}$, $f_{Ac}$, $f_{Ro}$</td>
</tr>
<tr>
<td>Mussi et al. [16]</td>
<td>2011</td>
<td>Ring PSO</td>
<td>W L ring</td>
<td>S 1</td>
<td>32</td>
<td>$f_{sp}$, $f_{G}$, $f_{Ro}$, $f_{S}$</td>
<td>1-120</td>
<td>0.1-21.7, 8-138, $f_{sp}$ (D&lt;100), $f_{G}$, $f_{Ro}$ (D&gt;20)</td>
</tr>
<tr>
<td>Mussi et al. [17]</td>
<td>2011</td>
<td>-</td>
<td>A 1-122</td>
<td>32, 64, 128</td>
<td>$f_{S}$</td>
<td>1-9</td>
<td>0.02-0.35</td>
<td>7-30</td>
</tr>
<tr>
<td>Cardenas-Montes et al. [18]</td>
<td>2011</td>
<td>-</td>
<td>27, 32</td>
<td>$f_{sp}$, $f_{S}$, $f_{G}$, $f_{Ro}$</td>
<td>1-120</td>
<td>0.02-0.35</td>
<td>2-250, $f_{sp}$, $f_{S}$ (D&lt;1120), $f_{G}$, $f_{Ro}$ (D&lt;20), $f_{Ro}$ (D&lt;10), $f_{G}$ (D&lt;5)</td>
<td>GeForce GTX260AMP, GeForce GT5450</td>
</tr>
<tr>
<td>Cardenas-Montes et al. [19]</td>
<td>2011</td>
<td>-</td>
<td>20</td>
<td>$f_{S}$, $f_{G}$, $f_{Ro}$</td>
<td>20000</td>
<td>?</td>
<td>20.4-26.7</td>
<td>?</td>
</tr>
<tr>
<td>Zhou et al. [20]</td>
<td>2011</td>
<td>GPU-PSO</td>
<td>C L ring</td>
<td>S 1</td>
<td>512-1280</td>
<td>$f_{sp}$, $f_{G}$, $f_{Ro}$ (-10,10), $f_{G}$, $f_{Ro}$</td>
<td>50-200, 1000, 2000</td>
<td>1.17-128.2</td>
</tr>
<tr>
<td>Hung et al. [21]</td>
<td>2012</td>
<td>GFSO</td>
<td>W G ring</td>
<td>S 1</td>
<td>16 - 104857</td>
<td>$f_{sp}$ (shifted), $f_{G}$, $f_{Ac}$, $f_{G}$, $f_{Ro}$ (-10,10), $f_{S}or_{5}$ others</td>
<td>100</td>
<td>&lt;1 - 100, 1-270, $f_{sp}$, $f_{sp}$ (shifted), $f_{Ac}$, $f_{G}$</td>
</tr>
<tr>
<td>Cagnoni et al. [22]</td>
<td>2012</td>
<td>-</td>
<td>W L ring</td>
<td>S 1</td>
<td>32-182</td>
<td>$f_{sp}$, $f_{G}$, $f_{S}$, $f_{S}or_{1.2}$, $f_{G}$</td>
<td>32, 64, 128</td>
<td>0.1-10</td>
</tr>
<tr>
<td>Calazan et al. [23]</td>
<td>2012</td>
<td>-</td>
<td>W G S 7-224 32-1024</td>
<td>$f_{sp}$, $f_{S}$, $f_{Ro}$ (-10,10)</td>
<td>30</td>
<td>0.38-1.13</td>
<td>?</td>
<td>$f_{sp}$, $f_{G}$, $f_{R}$s (error &lt; 0.011)</td>
</tr>
<tr>
<td>Roberge et al. [24]</td>
<td>2012</td>
<td>CUDA-PSO</td>
<td>W G S 1 256-1638</td>
<td>normalized $f_{Ro}$</td>
<td>20</td>
<td>?</td>
<td>20-256</td>
<td>?</td>
</tr>
<tr>
<td>Roberge et al. [25]</td>
<td>2012</td>
<td>parallel PSO</td>
<td>W G S 1 256-1638</td>
<td>normalized $f_{Ro}$</td>
<td>20</td>
<td>0.1-100</td>
<td>20-215</td>
<td>?</td>
</tr>
<tr>
<td>Calazan et al. [26]</td>
<td>2013</td>
<td>GPU-PSO</td>
<td>W L ring</td>
<td>S 1</td>
<td>6-1024</td>
<td>$f_{sp}$, $f_{Sw}$, $f_{Ro}$ (-16,16), $f_{G}$</td>
<td>2-256</td>
<td>0.05-9</td>
</tr>
<tr>
<td>Calazan et al. [27]</td>
<td>2013</td>
<td>CPPSO</td>
<td>W L ring</td>
<td>2 A 4-256</td>
<td>$f_{sp}$, $f_{Sw}$, $f_{Ro}$ (-16,16), $f_{G}$</td>
<td>2-256</td>
<td>0.02-1</td>
<td>1-81.5, $f_{sp}$, $f_{Sw}$ (both error &lt; 0.0001) $f_{Ro}$ (D&lt;128), $f_{Ro}$ (D&lt;16)</td>
</tr>
<tr>
<td>Kumar et al. [28]</td>
<td>2013</td>
<td>-</td>
<td>W G S 16-224</td>
<td>500,100, 1500</td>
<td>Shifted: $f_{sp}$, $f_{L}$, $f_{G}$, $f_{Ro}$, $f_{Ac}$</td>
<td>1000</td>
<td>96-1211</td>
<td>9-60</td>
</tr>
</tbody>
</table>

**Name** – algorithm name; **V**- variants of the velocity calculation; **S** – standard PSO; **W** – PSO with the inertia weight; **C** – canonical PSO; **N** – neighborhood topology; **G** – Gbest and Lbest respectively, **S** –
synchronous PSO, A – asynchronous PSO, M – number of swarms, Sup – Speedup ratio; ?- no data available
<table>
<thead>
<tr>
<th>References</th>
<th>Year</th>
<th>Variant</th>
<th>Swarm size</th>
<th>Objective function</th>
<th>Problem description</th>
<th>Sup</th>
<th>GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mussi et al. [29]</td>
<td>2009</td>
<td>St L</td>
<td>S 1 64</td>
<td>Weighted Bhattacharyya coefficients in 3 channels HSV color space.</td>
<td>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.</td>
<td>15</td>
<td>GeForce 8800GT</td>
</tr>
<tr>
<td>Mussi et al. [30]</td>
<td>2010</td>
<td>Wt ?</td>
<td>S 1 10</td>
<td>Compares the silhouettes generated by the model with the silhouettes extracted from images.</td>
<td>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.</td>
<td>20</td>
<td>Quadro FX 5800</td>
</tr>
<tr>
<td>Solomon et al. [31]</td>
<td>2011</td>
<td>W G A</td>
<td>1-60 128</td>
<td>Max Machine Available Time (MAT): the total amount of time required by the machine to complete all tasks.</td>
<td>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.</td>
<td>32</td>
<td>GeForce GTX260AMP</td>
</tr>
<tr>
<td>Wachowiak et al. [32]</td>
<td>2012</td>
<td>Wt G S</td>
<td>1 500-400</td>
<td>Toy protein folding function (3D), logistic function (2D), disequilibrium function (8D)</td>
<td>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</td>
<td>298</td>
<td>Tesla S1070</td>
</tr>
<tr>
<td>Datta et al. [33]</td>
<td>2012</td>
<td>C G S</td>
<td>1 256-1024</td>
<td>Self Potential model (5D), Magnetic Model (4D), Resistivity Model (2D)</td>
<td>Geology – invert Self Potential (Surda Area of Jharkhand, India), Magnetic anomaly of Boston Township and Resistivity (Satkui) models. Optimization of model parameters.</td>
<td>22</td>
<td>NVIDIA 9200M GS</td>
</tr>
<tr>
<td>Nobile et al. [34]</td>
<td>2012</td>
<td>W G S</td>
<td>3, 4 52</td>
<td>Distance between the sampled in the experiment biochemical species, and a simulated dynamics from stochastic simulation algorithm.</td>
<td>Estimation of the stochastic constants of two simple systems the Michaelis-Menten kinetics (2D) and a prokaryotic auto-regulatory gene network (anomaly of Boston Township) and Resistivity (Satkui) models.</td>
<td>24</td>
<td>Tesla C1060</td>
</tr>
<tr>
<td>Platos et al. [35]</td>
<td>2012</td>
<td>W ? ?</td>
<td>4 10240</td>
<td>Combination of precision and recall (classification measures)</td>
<td>Document classification form data-sets: Reuters-21578, Iris collection, 20 News group with different number of tokens.</td>
<td>10.5</td>
<td>Tesla C2050</td>
</tr>
<tr>
<td>Rabinovich et al. [36]</td>
<td>2012</td>
<td>St G S</td>
<td>2 256-28160</td>
<td>Optimize the sum of the priorities of the signals that fall within the placement of the three receivers/jammers (3x48D)</td>
<td>Radio Frequency Resource Allocation Optimizer – allocation of radio frequency resources with constraints of bandwidth and power.</td>
<td>5</td>
<td>GeForce GTX 465</td>
</tr>
<tr>
<td>Reguera-Salgado et al. [37]</td>
<td>2012</td>
<td>C L S</td>
<td>1 50</td>
<td>Minimum Root Mean Square Error (RMSE) of the differences between the GCPs and associated projected pixels locations, (6D)</td>
<td>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.</td>
<td>20s-220s</td>
<td>GeForce 9500GT</td>
</tr>
<tr>
<td>Roberge et al. [24]</td>
<td>2012</td>
<td>W G S</td>
<td>1 32-512</td>
<td>Harmonic minimization in multilevel inverters (2D).</td>
<td>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.</td>
<td>115</td>
<td>GTXS60Ti</td>
</tr>
<tr>
<td>Roberge et al. [25]</td>
<td>2012</td>
<td>W G S</td>
<td>1 32-256</td>
<td>Distance between the virtual marker projected on the 2D image and the actual marker identified on the 2D image</td>
<td>High-speed camera on the aircraft records multiple bomb drops as 2D video images. Bombs' position in 3D is obtained from 2D image. Problem: 6 degrees of freedom (6DOF) (x, y, z, yaw, pitch, and roll).</td>
<td>140</td>
<td>GTXS60Ti</td>
</tr>
<tr>
<td>Rymut et al. [38]</td>
<td>2013</td>
<td>W G S</td>
<td>1 100-400</td>
<td>An overlap between person silhouette and 3D model + image to camera edge distance (4 cameras)</td>
<td>Marker-less human motion tracking system – recovery of humane pose. Combination of Particle Filter and PSO. Result: 16 frames/s</td>
<td>7.5</td>
<td>GeForce 590GTX</td>
</tr>
<tr>
<td>Zhang et al. [39]</td>
<td>2013</td>
<td>C L</td>
<td>S 1 128</td>
<td>Minimize error: the sum of the differences (distances) in two models for every sampling set (31D)</td>
<td>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.</td>
<td>30</td>
<td>GeForce GTX 295</td>
</tr>
<tr>
<td>Rymut et al. [40]</td>
<td>2014</td>
<td>W G S</td>
<td>1 100-1000</td>
<td>An overlap between person silhouette and 3D model + image to camera edge distance (4 cameras)</td>
<td>Real-time body human motion tracking system – recovery of humane pose. 3D model has 26 DOF. Fitness function is decomposed. Result: 12 frames/s</td>
<td>12</td>
<td>GeForce 590GTX</td>
</tr>
<tr>
<td>Ma et al. [41]</td>
<td>2014</td>
<td>W G S</td>
<td>1 64-2560</td>
<td>The difference between the measured data and the calculated current is gradually minimized</td>
<td>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</td>
<td>80</td>
<td>GeForce GTX760</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>9400M</td>
</tr>
</tbody>
</table>
need to be estimated was used.

Van Heerden et al. [42] 2014 W L S T 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.

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

| Van Heerden et al. [42] | 2014 | W | L | S | T 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 |
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.

<table>
<thead>
<tr>
<th>References</th>
<th>Year</th>
<th>Variant</th>
<th>Objective function</th>
<th>Problem description</th>
<th>Sup</th>
<th>GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Papadakis et al.</td>
<td>2011</td>
<td>Comprehensive Learning PSO</td>
<td>Minimize sum of incremental cost function of each unit penalized by technical constraints (9-72 D).</td>
<td>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.</td>
<td>36.6</td>
<td>GeForce GTX 260</td>
</tr>
<tr>
<td>Zhu et al.</td>
<td>2011</td>
<td>Euclidean PSO (EPISO)</td>
<td>fSp, fgRo (-30,30), fgRa, fgGr, fAc</td>
<td>Standard test function for global optimization. Functions with (1000 ≤ D ≤ 8000) arguments were tested.</td>
<td>0.7-16.3</td>
<td>GeForce GTX 480</td>
</tr>
<tr>
<td>Chen et al.</td>
<td>2012</td>
<td>LaPSO (discrete)</td>
<td>Minimize weighted function that calculates inner Hamming distances between points in LHD.</td>
<td>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.</td>
<td>59</td>
<td>Tesla C2070</td>
</tr>
<tr>
<td>Sharma et al.</td>
<td>2012</td>
<td>Normalized PSO (NPISO)</td>
<td>Minimize a portfolio value that is the total estimated cost of the holdings of the investor.</td>
<td>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.</td>
<td>45</td>
<td>?</td>
</tr>
<tr>
<td>Zhang et al.</td>
<td>2012</td>
<td>Niching Bare Bone PSO (NPISO)</td>
<td>Minimize the sum of error from all body parts. Hierarchical optimization.</td>
<td>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.</td>
<td>30</td>
<td>GeForce GTX 295</td>
</tr>
<tr>
<td>Zhao et al.</td>
<td>2012</td>
<td>Parallel multi-swarm PSO</td>
<td>Minimize LSSVM model.</td>
<td>Prediction for gas holder level in the Linz Donawitz converter gas system based on least square support vector. The multiple sub-swarm (PSO) optimizes a model parameters.</td>
<td>65</td>
<td>GeForce GTX 260</td>
</tr>
<tr>
<td>Kilic et al.</td>
<td>2013</td>
<td>Binary PSO (BPSO) discrete</td>
<td>1) optimum pixel set of the impedance (32D) 2) optimize the height of each layer and the periodicity of the gratings</td>
<td>1) The optimization for tuning the shape of the antenna to the user-specified frequency. 2) The antireflective surface design.</td>
<td>10</td>
<td>4x Tesla C1060 graphics</td>
</tr>
<tr>
<td>Zan et al.</td>
<td>2014</td>
<td>Binary PSO (BPSO) discrete</td>
<td>Maximum of linear composition of profit and penalty (64-1024D)</td>
<td>The Multidimensional Knapsack Problem (MKP): select items from the available set to knapsacks of limited capacity.</td>
<td>9.6</td>
<td>GeForce GTX 580</td>
</tr>
<tr>
<td>Ouyang et al.</td>
<td>2014</td>
<td>PHPSO</td>
<td>Minimize the weighted sum of problem variables</td>
<td>One dimensional nonclassical heat conduction equation is modified into linear equation systems then transformed into an unconstrained optimization problem, which is optimized by PSO.</td>
<td>21</td>
<td>GTX465</td>
</tr>
</tbody>
</table>

?- no data available


