Overview and Applications of Particle Swarm Optimization on GPGPU

Sandeep U. Mane
Dept. of CSE, RIT
Rajaramnagar, Sangli
MS, India

Monica R. Pethkar
Dept. of CSE
Rajaramnagar, Sangli
MS, India

Pradnyarani K. Mahind
Dept. of CSE
Rajaramnagar, Sangli
MS, India

ABSTRACT
Particle Swarm Optimization is robust and effective method to solve optimization problems. Particle Swarm Optimization takes more time to find optimal solutions for complex real world problems. Execution time required to find optimal solutions depends on nature of problem as well as population and dimension size of the application. Compute intensive problems can be solved efficiently on General Purpose Graphics Processing Unit using Particle Swarm Optimization to diminish processing time. Graphics Processing Unit is used to provide speedup and to find optimal solutions of compute intensive problems earlier than central processing unit. Particle Swarm Optimization has eased to parallelize on Graphics Processing Unit using CUDA. This paper's main contribution is the review of parallelization techniques for Particle Swarm Optimization, performance optimization strategies and brief about different applications solved using Particle Swarm Optimization on GPGPU.

General Terms
General Purpose Graphics Processing Unit (GPGPU), Swarm Intelligence

Keywords
CUDA, Particle Swarm Optimization, Parallel Particle Swarm Optimization

1. INTRODUCTION
Particle Swarm Optimization (PSO) is a branch of Swarm intelligence used to solve many Engineering optimization problems. Among the stochastic approaches to continuous optimization, Swarm Intelligence algorithms offer a number of attractive features: robust and reliable performance, global search capability and easy implementation etc. PSO is a heuristic population based global optimization method developed by Kennedy and Eberhart in 1995[1]. It is based on the research of bird flocking and fish schooling behavior. Many times, PSO faces problems like premature convergence and curse of dimensionality, PSO reveal tremendous search performance with small population but performance can be improved by increasing the population size. As we increase the population size the search space explored for greater extent and this very large size of population require long time to perform massive calculations. It is called as population size problem. PSO reveals tremendous performance with small dimension, but PSO bear from the curse of dimensionality problem when applied to large dimensionality problems. As the dimension of search space increases it worsens performance of algorithm called curse of dimensionality.

Multi-Core CPUs are enhanced for sequential codes execution using only few number of cores; it never allows a CPU's to achieve its peak speed. On the other hand, GPU's are created with only one objective in mind i.e. fast parallel computation on large volume of data. A GPU is inherently good at processing a large amount of data in parallel. Another reason why the GPU is so fast is because of its extremely high memory bandwidth. E.g. DDR3 1333MHz system memory has a bandwidth of 32GBps (Gigabytes per second), the corresponding bandwidth of a GeForce 590GTX is 328 GBps. During computation, the sequential part is optimally executed on CPU and parallel part takes advantage of GPGPU execution in parallel way. GPGPU also reduces time to certain level so that different industrial and commercial people uses it to solve different compute intensive problems, which have scope for parallel operation [2]. [18].

GPGPU have many features like low cost, high performance capability and high availability, due to these reasons many researchers move towards it. In 2007, NVIDIA has developed GPGPU based Compute Unified Device Architecture (CUDA) as a platform designed to implement application using parallel programming.

PSO algorithm solves problem by adjusting particle's local position and based on locally best, updates its global position to obtain best results. PSO has a huge scope to perform different operation in parallel on GPGPU and improve performance while minimizing the data transfer between CPU and GPU with near optimal results. During development of PSO on CUDA, Fitness function, position and velocity update of particles in the swarm can be computed on GPU while initialization can be done on CPU [2].

This paper aims to present review and familiarize research community about GPGPU & CUDA, PSO and problems solved using PSO on GPGPU. The section 2 describes brief about CUDA and GPGPU architecture. Section 3 delivers issues and strategies to implement PSO on GPGPU. This review clarifies various performance optimization strategies for implementation of PSO on GPGPU. In section 4, recent research work about PSO using CUDA, carried out by various researchers is discussed. Finally, in section 5, conclusions based on study are presented.

2. GPGPU AND CUDA ARCHITECTURE
The General Purpose Graphic Processor Unit or simply GPU has evolved into a highly parallel, multithreaded, many-core processor with great computational power and very high memory bandwidth [2]. GPU cores optimized for data parallelism and throughput computation. Figure 1 and Figure 2 shows Floating-Point Operations per Second for the CPU - GPU and memory bandwidth for CPU and GPU respectively.

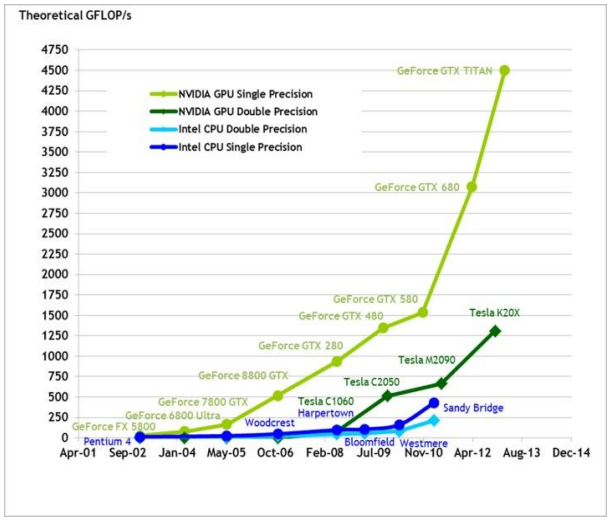

Fig 1: FOPS for CPU & NVidia's GPU [2]

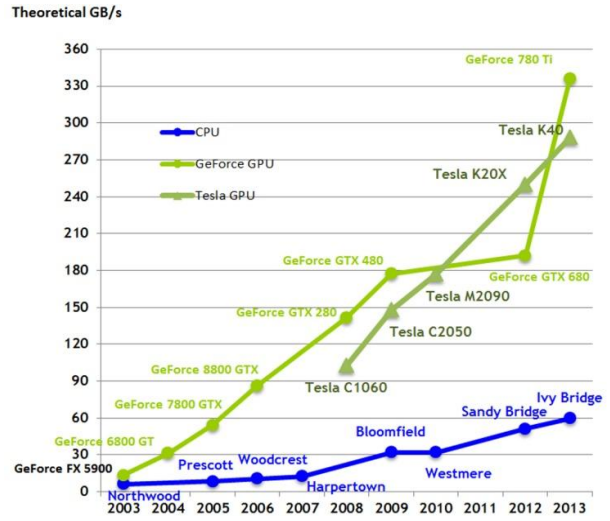

Fig 2: Memory bandwidth for CPU & NVidia's GPU [2]

CUDA is an API that can support many languages and programming environments, including C, C++, JAVA, FORTRAN, OpenCL, Python and DirectX Compute by which we can write application for GPU. CUDA consist of different components like kernel, host, device, grid, block and thread etc. CPU is act as a Host which executes sequential code and it also has control of a program. GPU is act as a device which executes parallel code. A Kernel Function is an implicitly parallel subroutine that executes under the CUDA execution and memory model for every Thread in a Grid.

Grid is a set of blocks that perform same kernel and data can be shared by global memory. On the other hand, Block is a collection of threads in which data can share by shared memory. Thread is a basic entity of parallel execution. The NVidia's Fermi architecture GPU consists of grid of size 65 535 X 65 535 blocks and each block has up to 1024 threads. CUDA is suitable for problems where by identical instruction set is executed on multiple threads i.e. Single Instruction Multiple Threads (SIMT). When a CUDA program on the host CPU invokes a kernel grid, the blocks of the grid are distributed on multiprocessors. The threads in the block execute concurrently on multiprocessors, and multiple thread blocks can execute concurrently on multiprocessors. The GPGPU based CUDA architecture is shown in Figure 3.

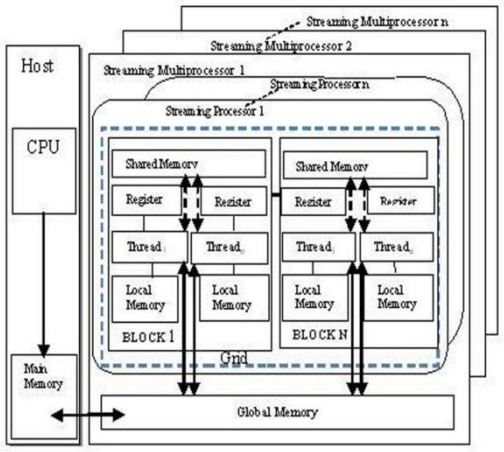

Fig 3: GPGPU based CUDA Architecture

Presently, NVIDIA and AMD are top most GPU venders and their recent top ten graphics cards are shown in Table I with rating.

Table 1. TOP 10 GRAPHICS CARDS [3]

Graphics Card	Vendor	Rating
GeForce GTX 690	NVidia	9.98
Radeon HD 7970	AMD	9.63
GeForce GTX 680	NVidia	9.58
GeForce GTX 670	NVidia	9.40
Radeon HD 7870	AMD	9.13
GeForce GTX 660 Ti	NVidia	8.73
Radeon HD 7850	AMD	8.25
GeForce GTX 580	NVidia	7.78
GeForce GTX 650	NVidia	7.53
Radeon HD 6670	AMD	7.15

10-9 (Excellent), 8-6 (Good), 5-4 (Average), 3-2 (Poor) and 1-0(Bad)

3. STRATEGIES FOR OPTIMIZATION AND PARALLELIZATION OF PSO ON GPGPU

The optimization of PSO implementation on GPGPU in terms of memory usage, communication between CPU and GPU can help to improve performance in terms of speedup, maximum utilization of resources, etc. The probable performance optimization strategies of PSO on GPGPU discussed in [2] [4] are:

1) Maximize parallel execution to obtain maximum utilization.
2) Minimize data transfers between host and device with low bandwidth.
3) Maximizing usage of on-chip memory: shared memory and caches.
4) Optimize memory usage to achieve maximum memory throughput.
5) Optimize instruction usage to achieve maximum instruction throughput.
6) Another strategy is to have threads arranged into blocks; where each block runs on one multiprocessor. It is also possible to have more blocks than multiprocessors and more threads per block than cores, to get optimal use of GPU.
7) Shared memory may be accessible only within the block and thread synchronization is possible also only within the block.

8) To achieve high bandwidth, shared memory is divided into equally-sized memory modules, called banks, it can be accessed concurrently.
9) Maximizing overall memory throughput for the application is to minimize data transfers with low bandwidth.

The way in which PSO is parallelized, is also important factor to optimize implementation on GPGPU. The parallelization of PSO depends upon different parameters like how to calculate fitness of each particle, how velocity and position updated, and which topology used. Researchers working in the area of high performance computing have proposed different strategies for implementing PSO on different parallel platforms. Most popular parallelization strategies are shown in Figure 4. There is also diffusion model and island model. In diffusion model unique population is considered and is suitable for shared memory architecture. In island model, population is divided into subpopulations and is well suited for implementation on cluster or grid computing [5]. Multi-population and repulsive multi-population methods are found in literature [6].

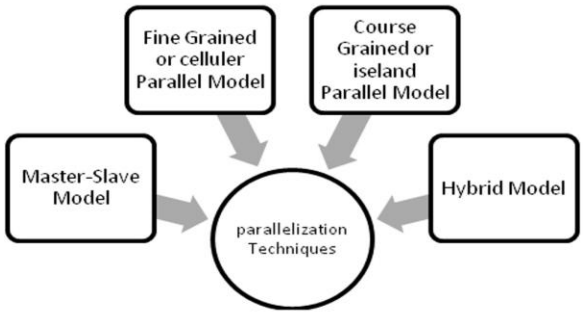

Fig 4: Parallelization strategies (Taken from [4])

The PSO algorithm on GPGPU can be designed using two different strategies, namely heterogeneous approach and homogeneous approach. In heterogeneous approach sequential part is executed on CPU while compute intensive or parallel part is executed on GPGPU. In homogeneous approach all computations are carried out on GPGPU.

Each of these strategies has its own merits and demerits. If optimization problem has huge search space then multi-population method is applicable. Master-slave approach is suitable for optimization problems with complex fitness function. Hybrid parallelization strategy is suitable for complex fitness function as well as huge search space of an optimization problem [6].

Though, here different researcher's implementation of PSO on

GPGPU is discussed; the PSO has scope of implementation on distributed systems like HPC, cloud, cluster, grid, and even on Hadoop to improve its performance in terms of time and speedup.

4. APPLICATIONS OF PSO ON GPGPU: AN OVERVIEW

The nature inspired techniques were developed to solve different categories of NP problems in polynomial time. Researchers are working to improve performance of such techniques with respect to speedup, throughput, solution quality, efficient utilization of available resources. In this section, different applications or problems solved by PSO on GPGPU are presented.

Miguel Cardenas-Montes et al. [7] observed Schwefel's Problem (full-non-separable function) which is solved using PSO on GPU. This problem requires high CPU time consumption for evaluation. The results show the excellent performance improvement on GPU. They observed that higher dimensionality show a better exert the parallelism capacity of GPU.As the number of variables increases, the GPU maps data to threads that make parallelism more powerful.

Mussi et al. [8] proposed GPU based Road Sign Detection using Particle Swarm Optimization. Running speed of GPU version is 20 times as fast as that of CPU.

Wenna and Zhenyu [9] proposed A CUDA based Multi-Channel Particle Swarm Algorithm. Parallelism performed on benchmark functions like Sphere, Rastrigin, Griewangk and Rosenbrock. Comparison of result on GeForce 480GTXGPU with Intel Core i7 860 shows, as population gradually increases, speedup also increases.

Zhou and Tan [10] presented parallel approach to run Standard Particle Swarm optimization (SPSO) on Graphic Processing Unit (GPU). Some experiments are conducted by running SPSO both on GPU and CPU, respectively. The running time of the SPSO based on GPU (GPU-SPSO) is greatly shortened compared to that of the SPSO on CPU (CPU-SPSO). Running speed of GPU-SPSO can be more than 11 times as fast as that of CPU-SPSO, with the same performance. As compared to CPU-SPSO, GPU-SPSO shows better speed advantages on large swarm population applications and high dimensional problems that can be widely used in real optimizing problems.

Zhu and Guo [11] suggested Euclidean Particle Swarm Optimization (EPSO) with CUDA. As the dimensions of optimization and local optima increases, EPSO require large scale of computing and long time in calculation. In order to overcome the drawback of EPSO require long time of massive calculation, fine grained data parallelism employed to calculate fitness with GPU to implement PEPSO based on CUDA. Experimental results shown that compared with EPSO the EPSO's maximum speedup increased 16.27 times.

Calazan et al. [12] proposed GPU based Parallel Dimension Particle Swarm Optimization. The optimization problems with low computational complexity i.e. low dimensions, CPU based PSO gives better performance than GPU based PDPSO. GPU provides positive impact on large optimization problems. Fine Grained model is used i.e. distribute one dimension to one thread. GPU-PSPSO is 85 times faster than CPU-PSO.

Papadakis and Bakrtzis [13] proposed Economic Dispatch Problem using Comprehensive Learning Particle Swarm Optimizer (CLPSO). Course grained and fine grained parallelism performed and compared these two parallel strategies with sequential implementation. From the experimental reading author observed that fine grained parallelism approach gives more speedup than course grained approach. Veronese and Krohling [14] implemented parallel PSO on NVIDIA GTX 280 using C-CUDA with different standard test functions. The results show the excellent performance improvement on GPU.

Hsieh and Chu [15] proposed GPU-Based Optimization of Tool Path Planning in 5-Axis Flank milling using PSO. Computation time of GPU is much shorter than CPU with negligible difference between results on GPU and CPU respectively.

Platos et al. [16] implemented PSO based Document Classification algorithm and compare the time complexity of problem with CPU, one GPU and two GPU. After comparison they observed that the speed of the 2 GPUs implementation is almost two times more than the single GPU implementation.

Zhang et al. [17] implemented Frequency Selective Surface using PSO algorithm. They observed that with GPU, FSS runs 100 times faster than CPU. GPU takes up 2 hours and 48 Minutes for entire optimization process, while CPU simulation takes up more 7 days for the same optimization.

Bastos et al. [18] investigates impact of the Random Number Generator Quality on PSO Algorithm Running on GPU. They analyze the computational time sequential implementation and parallel implementations of CUDA synchronous and CUDA asynchronous. Both GPU versions are much faster than sequential but asynchronous version is slightly faster than synchronous version.

Jambhlekar et al. [19] implemented Multi Objective PSO (MOPSO) crowding distance algorithm on GPU using CUDA and OpenCL. The computation time is reduced by GPU efficiently. As concern with different benchmark functions, CUDA gives better performance than OpenCL.

Chang and Fang [20] proposed Band Selection for Hyper spectral Images Based on Parallel Particle Swarm Optimization Schemes and compare the result of PPSO with PSA. They analyze that PPSO significantly improves the computational loads and provide a more reliable quality of solution than PSA.

Zhu and Curry [21] proposed Particle Swarm- Pattern Search Optimization algorithm. They concentrate on performance analysis and parallelization analysis. Performance is improved by hybridizing PSO and Pattern search algorithm as PS2. Speedup is increased by parallelizing algorithm using GPU.

Sharma et al. [22] introduced Normalized PSO (NPSO) to solve Portfolio Management. The overall speedup is about 40 for Portfolio Management when compared with CPU based method.

Salgado and Herrero [23] introduced Ground Control Point (GCB) based nonlinear registration of airborne push broom imagery, based on an implementation of PSO. It is observed that, solving this problem on GPU using PSO is speedy.

Rabinovich et al. [24] introduced Gaming PSO (GPSO) for radiofrequency resource allocation optimizer which implemented on GPU. When serial version is compared with proposed method, the speed up of 5X is observed.

Kromer et al. in [25] provided a brief survey about design, implementation and applications of parallel PSO on GPGPU.

Applications developed using PSO and its variants are summarized in Table 2.

Table 2. SUMMARY OF PROBLEMS SOLVED USING PSO & ITS VARIANTS ON GPGPU

PSO Type	Selected Problem /Objective of Study	Year
PSO [19]	Band Selection for Hyperspectral Images	2009
LSPSO [20]	Bound Constrained Problems	2009
PSO [7]	Road Sign Detection	2009
SPSO [9]	Benchmark functions	2009
PSO [13]	Benchmark functions	2009
SIMT PS2 [21]	12 benchmark optimization functions	2009
PSO [14]	Optimization of Tool Path Planning in 5-Axis Flank Milling	2010
PSO [17]	Impact of the Random Number Generator Quality on PSO	2010
PSO [6]	Schwefel's Problem	2011
PSO [8]	Benchmark functions	2011
EPSO [10]	Benchmark functions	2011
MOPSO [18]	Benchmark functions	2011
CLPSO [12]	Economic Dispatch Problem	2011
SyncPSO and ringPSO [26]	Standard benchmark function	2011
Parallel Multi-objective PSO [5]	Two objective Test functions	2011
PSO [15]	Documentation Classification	2012
GBC based PSO [23]	Orthorectification of Airborne Push broom Imagery	2012
PSO[22]	Portfolio Management	2012
BPSO [17]	Frequency Selective Surface	2012
Parallel PSO [27]	Real-Time Harmonic Minimization in Multilevel Inverters	2012
PDPSO [11]	Benchmark functions	2013
PSO and DE [28]	model-based object detection	2013
Parallel PSO [29]	Impact of problem properties on execution time	2013
Continuous Cellular Automaton with PSO [30]	Simulation of deep reactive ion etching	2013
PSO [31]	PSO as a hardware coprocessor to the MicroBlaze processor	2014
Cooperative coevolutionary PSO [32]	Automatic calibration of urban Cellular Automata (CA) models	2014
PSO [33]	Real-time trajectory planning of the under-actuated nonlinear Acrobot	2014
Parallel PSO [34]	Parameter Estimation for Photovoltaic Models	2014

From the literature, after reviewing different applications to achieve better optimal result for complex optimization problems, the search space can be explored and that can be implemented by increasing swarm size. If population size increased then to get optimal solution, time required will be more so such problems can be tackled by parallelizing PSO on GPGPU. The real time problems, those have Single Instruction Multiple Data (SIMD) nature, such problems efficiently solved by PSO on GPGPU.

From this study few things can be drawn out:

- PSO is efficient and robust swarm intelligence method whose performance can be improved with latest high performance computing paradigm to achieve high speed up.
- GPU is helpful to reduce the computational time and improve the speedup of large compute intensive applications.
- Problem of metaheuristic like population size and course of dimensionality could be effectively solved using GPGPU.

5. CONCLUSIONS

This article presents a general overview of GPGPU and CUDA, parallelization strategies for PSO and brief review of different applications of PSO on GPGPU.

CPUs are far behind than GPUs in terms of memory bandwidth and floating point operations performed per second. NVidia and AMD are topmost GPU vendors whose memory bandwidth is more than 270GB/s and 4500GFLOP/s floating point operations can be performed.

The PSO algorithm has various compute intensive calculations like evaluation of fitness function of each particle in the swarm, updating velocity and position of each particle at every iterations, etc. these operations could be performed on GPGPU effectively. This will minimize processing time of PSO. From the study, it can be stated that, variation of PSO are implemented on GPGPU to solve complex or compute intensive optimization problems.

The PSO algorithm, other than GPGPU, it can be implemented on cluster of GPGPUs, cloud, cluster of CPUs, grid, and even on Hadoop to improve its performance in terms of time and speedup. Parallel PSO implementation on latest computing paradigms can solve large dimension compute intensive problems.

6. REFERENCES

[1] Kennedy, J. and Russell, E.1995. Particle swarm optimization. In Proceeding of IEEE International Conference on Neural Networks.

[2] NVidia. CUDA-C Programming Guide version 5.1. 2013.

[3] Top Ten Review. Top ten graphics cards list, 25 August 2013. http://graphics-cards-review.toptenreviews.com.

[4] Umbarkar, A.J., Joshi, M.S., and Rothe, N.M. 2013. Genetic algorithm on general purpose graphical processing unit: Parallelism review. J. ICTACT Soft Computing. 3. 492–497.

[5] Zhou, Y. and Tan, Y. 2011. GPU-based parallel multiobjective particle swarm optimization. J. Artificial Intelligence. 7(A11). 125–141.

[6] Majd, A., and Sahebi, G. 2014. A Survey on Parallel Evolutionary Computing and Introduce Four General Frameworks to Parallelize All EC Algorithms and Create New Operation for Migration. J. Information and Computing Science, 9(2), 097-105.

[7] Miguel, C. M., Miguel, A., Rodrguez-Vazquez, J. J., and Antonio, G. I. 2011. Accelerating particle swarm algorithm with GPGPU. In IEEE 19th International Euro micro Conference on Parallel, Distributed and Network-Based Processing.

[8] Mussi, L., Stefano, C., and Daolio, F. 2009. GPU-based road sign detection using particle swarm optimization. In 9th IEEE International Conference on Intelligent Systems Design and Applications.

[9] Wenna, L. and Zhenyu. Z. 2011. A CUDA-based multi-channel particle swarm algorithm. In 4th International Conference on Control, Automation and Systems Engineering.

[10] Zhou, Y. and Tan, Y. 2009. GPU-based parallel particle swarm optimization. In IEEE Congress on Evolutionary Computation.

[11] Zhu, H. and Guo, Y. 2011. Paralleling Euclidean particle swarm optimization in CUDA. In 4th International Conference on Intelligent Networks and Intelligent Systems.

[12] Calazan, R. D. M., Nedjah, N., and Mourelle, L. D. M. 2013. A Cooperative Parallel Particle Swarm Optimization for High-Dimension Problems on GPUs. In IEEE Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC).

[13] Papadakis, S. E., and Bakrtzis, A.G. 2011. A GPU accelerated PSO with application to economic dispatch problem. In 16th IEEE International Conference on Intelligent System Application to Power Systems (ISAP).

[14] Veronese, L. P., and Krohling, R. A. 2009. Swarms flight: Accelerating the particles using C-CUDA. In IEEE Congress on Evolutionary Computation.

[15] Hsieh, H. T., and Chu, C. H. 2010. GPU-based optimization of tool path planning in 5-axis flank milling. In IEEE International Conference on Manufacturing Automation.

[16] Platos, J., Snasel, V., Jezowicz, T., Kromerand, P., and Abraham, A. 2012. A PSO-based document classification algorithm accelerated by the CUDA platform. In IEEE International Conference on Systems, Man and Cybernetics.

[17] Zhang, B., Zheng, H.,Wei, M., Wu, R., and Sheng, X. 2012. Particle swarm optimization of frequency selective surface. In IEEE International Conference on Cross Strait Quad- Regional Radio Science and Wireless Technology.

[18] Bastos-Filho, C. J., Oliveira, M. A., Nascimento, D. N., and Ramos, A. D. 2010. Impact of the random number generator quality on particle swarm optimization algorithm running on graphic processor units. In 10th IEEE International Conference on Hybrid Intelligent Systems (HIS).

[19] Jambhlekar, P. A., Mishra, M., and Subramaniam, S. V. 2011. Parallel implementation of MOPSO on GPU using OpenCL and CUDA. In 18th IEEE International Conference on High Performance Computing (HiPC).

[20] Chang, Y., L., and Fang, J. P. 2009. Band selection for hyperspectral images based on parallel particle swarm optimization schemes. In IEEE International Geoscience and Remote Sensing Symposium.

[21] Zhu, W., and Curry, J. 2009. Particle swarm with graphics hardware acceleration and local pattern search on bound constrained problems. In IEEE Swarm Intelligence Symposium.

[22] Sharma, B., Thulasiram, R. K., and Thulasiraman, P. 2012. Portfolio management using particle swarm optimization on GPU. In 10th IEEE International Symposium on Parallel and Distributed Processing with Applications.

[23] Javier, R.S., and Julio, M. H. 2012. High performance GBC based particle swarm optimization for orthorectification of airborne push broom imagery. In IEEE International Geoscience and Remote Sensing Symposium.

[24] Rabinovich, M., Kainga, P., Johnson, D., Shafer, B., Lee, J., J., and Eberhart, R. 2012. Particle swarm optimization on a GPU. In IEEE International Conference on Electro/Information Technology.

[25] Kromer, P., Platos, J., and Snasel, V. 2013. A brief survey of advances in particle swarm optimization on graphic processing units. In IEEE World Congress on Nature and Biologically Inspired Computing (NaBIC).

[26] Mussi, L., Daolio, F., and Cagnoni, S. 2011. Evaluation of parallel particle swarm optimization algorithms within the CUDA architecture. J. Information Sciences.181. 4642–4657.

[27] Roberge, V., and Tarbouchi, M. 2012. Efficient parallel particle swarm optimizers on GPU for real-time harmonic minimization in multilevel inverters. In 38th IEEE Annual Conference on Industrial Electronics Society (IECON).

[28] Ugolotti, R., Nashed, Y. S., Mesejo, P., Ivekovic, S., Mussi, L., and Cagnoni, S. 2013. Particle Swarm Optimization and Differential Evolution for model-based object detection. J. Applied Soft Computing, 13(6). 3092-3105.

[29] Altinoz, O. T., Yilmaz, A. E., and Ciuprina, G. 2013. Impact of problem dimension on the execution time of parallel particle swarm optimization implementation. In 8th IEEE International Symposium on Advanced Topics in Electrical Engineering (ATEE).

[30] Li, Y., Xing, Y., Gosalvez, M. A., Pal, P., and Zhou, Y. 2013. Particle swarm optimization of model parameters: Simulation of deep reactive ion etching by the continuous cellular automaton. In The 17th IEEE International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS & EUROSENSORS XXVII).

[31] Calazan, R. M., Nedjah, N., and Mourelle, L. M. 2014. A hardware accelerator for Particle Swarm Optimization. J. Applied Soft Computing. 14. 347-356.

[32] Blecic, I., Cecchini, A., and Trunfio, G. A. 2014. Fast and Accurate Optimization of a GPU-accelerated CA Urban Model through Cooperative Coevolutionary Particle Swarms. Procedia Computer Science. 29.

[33] Van Heerden, K., Fujimoto, Y., and Kawamura, A. 2014. A combination of particle swarm optimization and model predictive control on graphics hardware for real-time trajectory planning of the under-actuated nonlinear Acrobot. In 13th IEEE International Workshop on Advanced Motion Control (AMC).

[34] Ma, J., Man, K. L., Ting, T. O., Zhang, N., Guan, S. U., and Wong, P. W. 2014. Accelerating Parameter Estimation for Photovoltaic Models via Parallel Particle Swarm Optimization. In IEEE International Symposium on Computer, Consumer and Control (IS3C).