PARALLEL PARTICLE SWARM OPTIMIZATION ALGORITHM: A REVIEW

Mr. Kamble S.V[1], Mr. Sakhare V. C[2]
[1,2]Assistant Professor in Information Technology Dept.,
DKTE'S Textile & Engineering Institute, Ichalkaranji.

Abstract: In this paper we focus on the Parallel Particle Swarm Optimization Algorithm. Particle Swarm Optimization (PSO) is a met heuristic global optimization paradigm that has mostly used in the last two decades due to its ease of application in unsupervised, complex multidimensional problems that cannot be solved using traditional deterministic algorithms. The complex optimization problems with large size have a required higher computational cost of these problems has rendered the development of optimization algorithms with parallelization. Particle swarm optimization (PSO) algorithm is the most popular swarm intelligence-based algorithm, which is enriched with robustness, simplicity and global search capabilities. PSO algorithms are population-based, they are intrinsically parallel.
Keywords: parallel particle swarm optimization (PPSO).

I. INTRODUCTION

An optimization algorithm is a method which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found. There are several powerful optimization algorithms introduced in the field of Computational Intelligence from last two decades have seen unprecedented development with the advent of parallel processing capabilities that make little or no assumption about the nature of the problem .

The inner workings of the PSO make sufficient use of probabilistic transition rules to make parallel searches of the solution hyperspace without explicit assumption of derivative information [3][6][9].

PSO is a population-based metaheuristic algorithm that has proven itself to be one of the most efficient nature-inspired algorithms to deal with constrained or unconstrained and global optimization problems with one or many objectives . PSO has found use in an ever-increasing array of complex, real-world optimization problems where conventional approaches either fail or render limited usefulness [3][9].

Parallelization proposes consist of an excellent path to increase the system performance. For parallelization, multi-core CPU or GPU can be occupied. There are important issues in parallelization are the operating system, communication topologies, programming languages enriched with modules, functions and libraries.

The parallelization options include:

Hadoop MapReduce, CUDA, MATLAB parallel computing toolbox, R Parallel package, Julia: Parallel for and MapReduce, OpenCL, OpenGL, Parallel computing module in python, OpenMP with C++ [1][7].

The rest of the paper is organized as follows: historical overview and motivation for the Particle Swarm Optimization algorithm in section II, The details of PSO and its parallelization strategies discussed in section III, and section IV, presenting the summary of the studies performed on PPSO.

II. HISTORICAL OVERVIEW OF PARTICLE SWARM OPTIMIZATION :

The Particle Swarm Optimization algorithm
was formally introduced in 1995 by Eberhart and Kennedy through an extension of Reynold's work

PSO is a population based evolutionary algorithm and is motivated from the simulation of social behavior. It is an optimization technique based on swarm intelligence, which simulates the bio-inspired behavior. PSO is a popular global search method and the algorithm is being widely used in conjunction with several other algorithms in different fields of study.

Working Mechanism of PSO

A PSO algorithm consists of behavior of flying birds and it means that they exchange of information to solve optimization problems. PSO has been introduced as an optimization technique in real-number spaces. PSO is initialized with a population ("Swarm") of random solutions. Each individual or potential solution, called a particle, flies in the D-dimensional problem space with a velocity that is dynamically adjusted according to the flying experience of the individual and its colleagues.

Each particle remembers the best position that it has found so far during the search process personal best (pbest), and knows the best position of the swarm global best (gbest). Therefore, each particle interacts with other and every particle in the swarm tries to gradually move toward the promising areas of the search space and in this way an optimum solution is found[3][6][9].

The global model of equations is given below:

$$Mid = W \cdot M_{id} + C1 \cdot Rand() \cdot (P_{id} - N_{id}) + C2 \cdot Rand() \cdot (P_{gd} - N_{id}) \quad (1)$$

$$Nid = Nid + Mid \quad (2)$$

The first part equation (1) represents the inertia of previous velocity; whereas the second part is the "cognition" part, which represents individuals thinking independently; and the third part is the "social" part, represents cooperation among the particles [6][9].

In the equation (1), M_{id} the velocity for particle i represent the distance to be traveled by particle I from its current position. N_{id} represents the particle position; P_{id} represents

 Copyright 2020.All rights reserved.

"pbest" the local best solution of ith particle's best previous position. P_{gd} is gbest the global best solution represents the best position among all particles in the swarm.
W is the interia weight it regulates the trade of between the global exploration and local exploitation abilities of the swarm. The acceleration constants C1 and C2 represent the weight of the stochastic acceleration terms that pull each particle toward "pbest" and "gbest" positions. Rand ( ) are two random functions with range (0, 1) [1][3][9].

PSO Algorithm:
Initialize the positions and velocities of all particles
Repeat for 1 to particleNbr do
Compute Fitness value
Update best local position X_{pbest}
Update best global position X_{gbest}
Update Velocity $V_i(t + 1)$
Update position $X_i(t + 1)$
end for until the stopping criteria are reached

III. CPU BASED PARALLELIZATION METHOD

Parallelization:
Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously, with breaking computational problem into discrete parts. The computation may occur on a single machine as well as on multiple machines. Single machine processing includes computers utilizing multi-core, multiprocessor, and GPU with multiple processing elements. Multiple machine examples include clusters, grids [1].

Julia 1.0:
Julia is a modern open-source programming language which is modern, functional, and expressive, and has remarkable abstraction and metaprogramming capabilities. In Julia a design principle was to provide tools for easy and efficient parallelization of numerical code. One of those tools is the Distributed module, and its @distributed for prefix, designed for executing for loops in parallel. This way of parallelizing is most convenient for computing "small tasks," or situations where there are few computations on each parallel iteration. When implementing parallel computing in Julia, consider into account that the total computation speed is particularly sensitive to the use of global (@everywhere) variables and large SharedArray objects[1] [7].

Python:Parallel Computing module
Python is an open-source, interpreted, general purpose language that has become popular among many scientists due to its elegance, readability, and flexibility it containing multiple modules supporting parallel computations. Parallel function like joblib module, and Map-like function are most popular. The parallelization is done at every age through function Parallel, from the joblib package; it receives as inputs the number of workers of the parallel pool, the function Value, and a list of elements of the type modelState, one for each combination of values of the state variables. This function returns the function Value evaluated at each of the state variables [1] [7].

Parallel computing with R
R is a open-source language and environment for statistical computing and graphics. Developers have designed several parallel computing packages in R, out of which 'foreach' and 'doParallel' are extensively employed. Besides, C++ codes are embedded in R for executing parallel computing resulting Rcpp package [1].

GPU-Based Parallelization Strategies
From last decade parallel computation on GPUs has increasing popularity among researchers and in industry. GPU consist of thousands of cores installed and strength of multiple CPUs in one processor. Any CPU-based parallelization strategy can be implemented in GPU as well. GPU-based parallelization schemes are:

CUDA
Compute Unified Device Architecture (CUDA) is a parallel computing platform and application programming interface. The CUDA platform is specially designed to work with programming languages such as C, C++, and python. CUDA allows software developers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing – an approach termed GPGPU (General-Purpose computing on Graphics Processing Units).
General purpose parallel computing architecture called CUDA™ was introduced by nVIDIA™ in November 2016[Main]. User required to define function that run on the GPU once assign memory allocation of the variable. Then the process begins, starting from the initialization.
In CUDA parallel computing the first step to define functions that are going to be executed from within the GPU, as opposed to those that are executed from the CPU.The directive __device__, while the latter should be preceded by __global__: after that allocate memory in the device (GPU) for the variables that are going to be transferred from the host (CPU) at the beginning of the parallel computation [7].

IV. PARALLELIZATION OF PSO ALGORITHM

PSO is fulfilled to inherent parallelism, that particles in a swarm proceed in parallel, but the interactions of the particles remain non-simultaneous. The interaction of particles determines the gbest and pbest positions for velocity and position update. Communication between particle could be within the swam or sub- swarm.

Star PPSO:
Star PPSO is based upon master-slave topology. The communication done in a star shape i.e., one sub-swarm named as 'master'. Communicates the information between all the remaining sub-swarms named as 'slaves. There is no direct communication takes place between the slave in the process.

Star PPSO Algorithm [1]
Step 1: The master determines and shares all the algorithm parameters to the slaves. These parameters include number of iterations, inertia weight, communication period,

 Copyright 2020.All rights reserved.

population size and the acceleration coefficients.
Step 2: Each sub-swarm evolves separately and obtains its pbest and gbest.
Step 3: Then, all the other sub-swarms called slaves communicate their pbest information to the master node. This process occurs at a certain communication period.
Step 4: The master determines the gbest and communicates this information to all the slaves.
Step 5: Each sub-swarm updates its velocity and position.
Step 6: Again each slave communicates the information about its pbest to the master and the master determines the new gbest.
Step 7: The process continues until the termination criteria is achieved.

Broadcast PPSO:
In the model of broadcast PPSO all the sub-swarms communicate and execute in parallel. Information of each is broadcasted to all the sub-swarms.
Broadcast PPSO Algorithm [1]
Step 1: The master determines and shares all the algorithm parameters to the slaves. These parameters include number of iterations, inertia weight, communication period, population size and the acceleration coefficients.
Step 2: Each sub-swarm evolves separately and obtains its pbest and gbest.
Step 3: Then, all the sub-swarms share their pbest position information to obtain gbest of the swarm at a certain communication period.
Step 4: With updated pbest and gbest, sub-swarms update their positions and velocities.
Step 5: Step 3 is repeated.
Step 6: The process continues until the termination criteria is achieved

Neighborhood Parallel Model (NPM)
PSO algorithm a parallelization based on NPM model consist of parallel computing and dynamic neighborhood. The NPM algorithm starts by a division of the global search space into sub-spaces (which will represent the particles' neighborhoods), then a random initialization of the particles in the search space is made [8].

NPM algorithm [8]:
Step 1: Create the global search space and divide it into sub spaces according to a value of step.
Step 2: Generate randomly a set of particles by attributing their positions and speeds.
Step 3: Each sub set of particles is attributed to one of the created thread.
Step 4: Each thread evaluates the velocity and the position of all its own particles.
Step 5: Wait for all threads and update neighborhoods.
Step 6: If the stopping criterion is satisfied, stop, otherwise go to step.

Multi-PSO Parallel Model (MPM)
The MPM model is based on a set of PSO algorithms launched in parallel in search of the optimum. The global search space is divided into subspaces using a value of "step" attributed to each axis of the search space depending upon the value of objective function. Then assign each subspace a group of particles. All portions of the search space will be explored and exploited After dividing the global search space into sub-spaces, and initializing them by particles, the processing of each subspace is attributed to a "thread" [8].

MPM Algorithm [8]:
Step 1: Create the global search space and divide it into sub search spaces.
Step 2: Attribute particles for each sub search space by generating their positions, velocities and communication topology.
Step 3: Create PSO Thread* by sub search space.
Step 4: Evaluate the best solution for each PSO Thread.
*PSO Thread steps:
Step 1: Evaluate each particle's fitness.
Step 2: For each particle, if its fitness is smaller than its previous best (Pb) fitness, update Pb.
Step 3: For each particle, if its fitness is smaller than the best one (Pg) of all the particles, update Pg.
Step 4: Move all particles according to the formula (1) and (2).
Step 5: It the stopping criterion is satisfied, then stop, else go to Step 1.

IV. CONCLUSION

Large-size real-world complex problems have raised the demand of parallel computing techniques. Now day's researchers have obtained significant attention for solving complex applications using parallelization techniques with different strategy with different models. This paper discussed different method and strategy used for parallelization techniques with PSO. In feature researcher can also solve multi objective problems using different strategy of parallelization.

REFERENCES

[1] Soniya Lalwani and Harish Sharma,"A Survey on Parallel Particle Swarm Optimization Algorithms" Arabian Journal for Science and Engineering, https://doi.org/10.1007/s13369-018-03713-6J
[2] Narjess DALI and Sadok BOUAMAMA "GPU-PSO : Parallel Particle Swarm Optimization approaches on Graphical Processing Unit for Constraint Reasoning: Case of Max-CSPs",19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems,Procedia Computer Science 60 (2015) 1070 – 1080
[3] Saptarshi Sengupta,"Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives",Mach. Learn. Knowl. Extr. 2019, 1, 157–191; doi:10.3390/make1010010
[4] Yudong Zhang,"A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its

 Copyright 2020.All rights reserved.

Applications",Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2015, Article ID 931256, 38 pages http://dx.doi.org/10.1155/2015/931256

[5] nVIDIA.: nVIDIA CUDA Programming Guide v.2.3. nVIDIACorporation, Santa Clara (2009)

[6] S. V. Kamble,"Hybrid Multi-Objective Particle Swarm Optimization for Flexible Job Shop Scheduling Problem",I.J. Intelligent Systems and Applications, 2015, 04, 54-61,DOI: 10.5815/ijisa.2015.04.08

[7] Fernandez-Villaverdey, J.; Zarruk-Valenciaz, D.: A Practical Guide to Parallelization in Economics. University of Pennsylvania, Philadelphia (2018)

[8] Maria Zemzami and Norelislam El Hami "A comparative study of three new parallel models based on the PSO algorithm",Int. J. Simul. Multidisci. Des. Optim. 11, 5 (2020),https://doi.org/10.1051/smdo/2019022

[9] James Kennedy and Russell Eberhart,"Particle Swarm Optimization",0-7803-2768-3/95/$4.00 0 1995 IEEE

[10] Madhuri, A., Deep, K. " A state-of-the-art review of population based parallel meta-heuristics" In:World Congress on Nature and Biologically Inspired Computing, pp. 1604–1607 (2009)

[11] The Julia Language. https://docs.julialang.org/en/stable/manual/parallel-computing. Julia Parallel Computing (2018)

[12] Umbarkar A.J, "Review of parallel genetic algorithm based on computing paradigm and diversity in search space" ICTACT J. Soft Comput. 3(4), 615–622 (2013)

 Copyright 2020.All rights reserved.