Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 10 (8): 699-703, 2010
ISSN 1812-5654

Migration Algorithm of Particle Swarm Optimization For a Scheduling Problem

S. Hernane, Y. Hernane and M. Benyettou
Simulations and Modelling of Industrial Systems Laboratory,
Department of Computer Science, Faculty of Science, University of Science and Technology of Oran,
Mohammed Boudiaf USTO, P.O. Box 1505, El Menaour, Oran, Algeria

Abstract: We aim in this study to measure the performance of a distributed algorithm of Particle Swarm Optimization. The PSO is a bio-inspired algorithm founded on the cooperative behaviour of agents and is known as a tool to address difficult problems in numerous and divers fields. Like evolutionary algorithms, PSO offer practical approach to solve complex problems of realistic scale and gave results at least satisfactory. In addition, the performance of production systems is related to the scheduling of work on the one hand and to the assignment of this work of the various machines of the system on the other hand. The problem is noted Np-complete. Nevertheless, it remains that this algorithms require large computational demand in terms of CPU time and memory. Also, it is possible to improve solutions quality in various manners. In this research, we study the adequacy of a parallel distributed PSO algorithm for a scheduling problem in hybrid flow-shop (FSH) systems. We use a fault-tolerant environment by exploiting the computing power of a high-performance cluster with homogeneous processors. For this purpose, we study a parallel distributed model of PSO algorithm on a high-performance cluster with homogeneous processors. Experimental tests are compared with those obtained by the parallel genetic algorithms with migration.

Key words: Swarm intelligence, PSO, migration, metaheuristic, parallel virtual machine

INTRODUCTION

Combinatorial optimization problems are expressed by a cost function with or without constraints to be minimized or maximize on a set of definitions finished or countable. Metaheuristics bring approximate solutions to large problem instances and require an intensive scientific computation. However, their resolutions are limited by available resources capacities. Thus, suitable distributed parallel models decrease the computing time and improve the quality of obtained solutions. Also, the exploitation of several workstations is an opportunity for parallel computing but new problems are posed (resources heterogeneity, failure of node). These new challenges are overcome with the use of high performance clusters and grids computing. Grids computing are an emerging computing model that provides the ability to perform higher throughput computing by taking advantage of many networked computers to model a virtual computer architecture that is able to distribute process execution across a parallel infrastructure. Grids computing are often confused with cluster computing. The key difference is that a cluster is a single set of compute nodes sitting in one location, while a grid is composed of many clusters and other kinds of resources (Parsopoulos *et al*., 2004).

In this study, we analyse a parallel distributed model performance of a bio-inspired algorithm: the Parallel Swarm Optimization metaheuristic. The application solves a scheduling problem of tasks in a hybrid flow shop production system. We deploy a cluster of nodes based on resources manager (Condor) (Pruyne *et al*., 1995) coupled with Parallel Virtual Machine communication libraries (Requilé, 1995).

THE PSO ALGORITHM

In PSO algorithm, each particle is characterized by:

- Position and velocity
- The objective function cost for its current position or that acquired previously: pbest
- The knowledge of its neighbors
- The best previous and current position acquired among all the particles in the swarm gbest

The PSO is initialized with a set of random particles (solutions) and then searches for optima by updating generations (Hu *et al*., 2003).

Corresponding Author: Soumeya Hernane, Simulations and Modelling of Industrial Systems Laboratory,
Department of Computer Science, Faculty of Science, University of Science and Technology of Oran,
Mohammed Boudiaf USTO, P.O. Box 1505, El Menaour, Oran, Algeria

```
1 FOR EACH PARTICLE DO
        INITIALISATION
END

2 FOR EACH PARTICLE DO
        CALCUATE FITNESS VALUE;
        IF THE FITNESS VALUE IS BETTER
        THEM THE BEST FITNESS VALUE
        (PBEST)
        THEN SET CURRENT VALUE AS THE
        NEW PBEST
END

CHOOSE THE PARTICLE WITH THE BEST
FITNESS VALUE OF ALL THE PARTICLES AS
THE GBEST.

3 FOR EACH PARTICLE DO
CALCULATE PARTICLE VELOCITY
ACCORDING EQUATION (1)
UPDATE PARTICLE POSITION ACCORDING
EQUATION (2)
END
```

Fig. 1: Pseudo code of PSO algorithm

In each step, the particle is updated and makes a compromise between three possible choices (Fig. 1):

- To follow its own way
- To return towards its best obtained position
- To move towards the best obtained position of the swarm

Each particle, updates velocity and position according pbest and gbest.

The velocity restriction constant V_{max} was also included in the algorithm. If the sum of the tree parts exceeds a constant value, then Particle's velocity is clamped to a maximum velocity V_{max} that is specified by the user. This mechanism prevents the phenomenon of swarm explosion (Li-Ping *et al*., 2005).

The pseudo code of the procedure is as follows (Kennedy *et al*., 1995):

PARALLEL DISTRIBUTED STRATEGY OF PSO ALGORITHM

Initial swarm is divided into sub-swarms and distributed to cluster's compute nodes (Fig. 2). Each processor executes the PSO algorithm independently and accelerates convergence to the global optima by sending their best solutions (fitness) at each interval of iterations to a nearby node in a ring topology. This approach was applied to a problem of multiobjective optimization

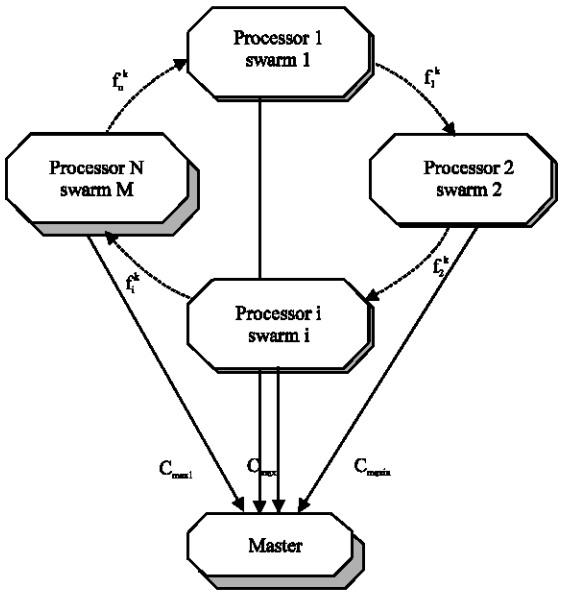

Fig. 2: Parallel Distributed PSO model

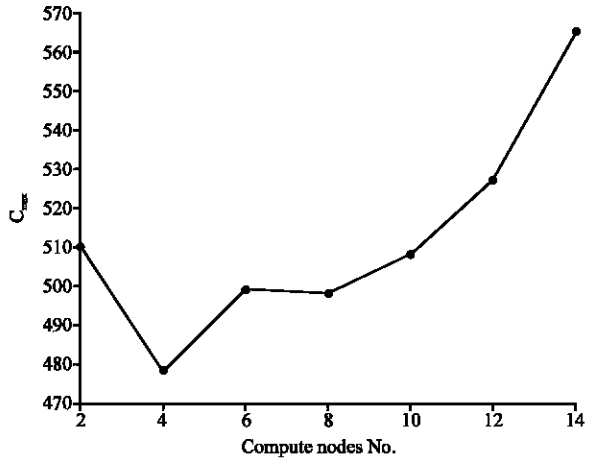

Fig. 3: Parallel distributed PSO: C_{max}

(Parsopoulos *et al*., 2004). In this study, we investigated this approach with an aim of measure parallel efficiency by studying the impact of the number of sub-swarms and particles on solution quality.

In order to minimize inter-nodal communication, which often forms the performance bottleneck on networked machines, we used a master/slave communication model with a minimal amount of communication. In our application, the master node is used exclusively for distributing sub-swarms and to coordinate the particle queue. The slaves executes PSO algorithm in parallel, migrate their best solution to a nearby slave and finally send the solution to the master (Fig. 3). We used a cluster of 14 nodes and PVM library

Table 1: The characteristics of the cluster

Characteristics	Description
CPU	3 Ghz
Memory	512
Cluster's size	2 to 14
Network	Fast ethernet 100 Mbits/sec
OS	LINUX
Compilator	Gcc
Communication	PVM
Workload manager	Condor

routines for sending and receiving messages. The main characteristics of our cluster are reported in Table 1.

EXPERIMENTAL TESTS AND RESULTS

Serial execution algorithm results are shown by Table 2:

In this case, the objective function corresponds to C_{max}.

The performance of a parallel program is related to execution time. By measuring how long the parallel program needs to run to solve our problem, we can directly measure its effectiveness. There are two important performance metrics: speedup and parallel effectiveness.

Speedup is the ratio of the serial program execution time T_s to the parallel execution time T_p on N processors. Parallel efficiency is the ratio between the speedup and the number N of processors. Parallel effectiveness is an indication of scalability. Ideal Speedup should equal the number of processors with a maximum of Parallel efficiency of 1 (100%).

Our infrastructure is optimized for large sized problem instances. The swarm is initialized with a population of 2048 particles and then divided into sub-swarms as follow:

- 2 sub-swarms of 1024 particles
- 4 sub-swarms of 512 particles
- 8 sub-swarms of 256 particles
- 146 sub-swarms of 146 particles

The swarm is divided 4 times. At each time, a cluster of 2, 4, 8 and 14 nodes evaluates the parallel implementation. These experiments aim to show if it is better to choose a parallel optimization with a significant number of sub-swarms with reduced sizes rather than some sub-swarms with a large population. For each case, 10 independent experiments have been performed. The C_{max} cost is computed; Speedup and the parallel effectiveness are measured.

According to the numerical results, it seems that using 2 swarms or more improves the performance of parallel algorithm in some cases (Fig 4). This improvement

Table 2: Serial execution algorithm results

Resutls	Values
Tasks	20
CPU time	19 sec
C_{max}	568

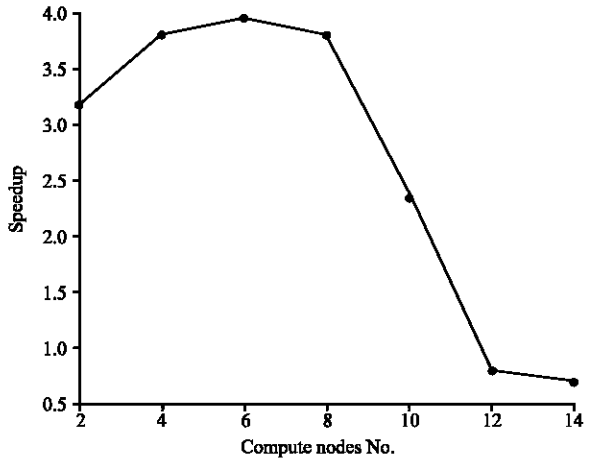

Fig. 4: Parallel distributed PSO: speedup

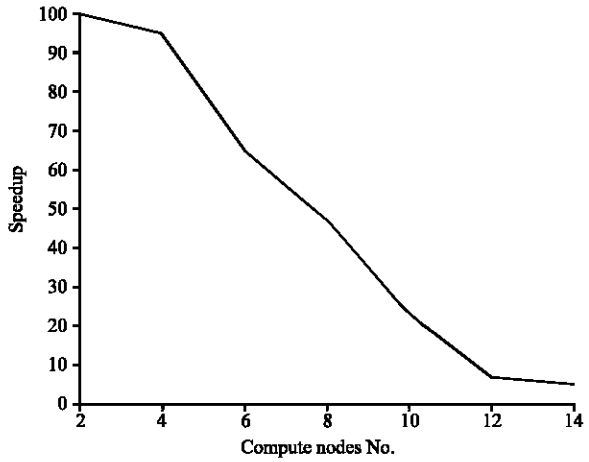

Fig. 5: Parallel distributed PSO: effectiveness

reaches the optimum (maximum) when using 4 sub-swarms of 512 particles. This implies that cooperation between sub-swarms by sending and receiving the C_{max} cost with regular intervals has advantages.

Reducing the swarm's size produces this degradation. For the case of 14 swarms, the size is equal 146. Thus, when used many sub-swarms with reduced amount of particles, the algorithm evolves within a restricted research space and that in spite of exchanges.

Concerning the parallel performance, speedup shows a good acceleration of the parallel algorithm, then performance degradation when using a cluster of 8 or 10 nodes (Fig. 5).

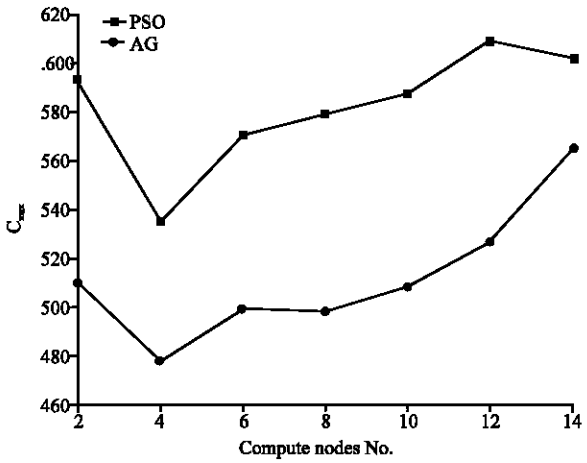

Fig. 6: Parallel distributed (PSO,GA): C_{max}

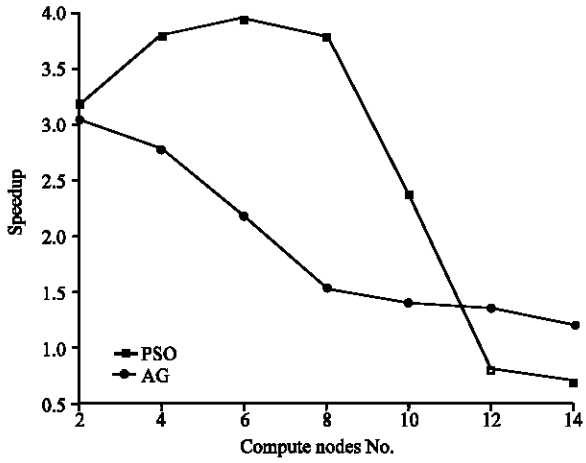

Fig. 7: Parallel distributed (PSO,GA): speedup

Although, the experimentations were accomplished within a high-performance cluster, we seen that parallelization becomes ineffective with 10 compute nodes. Indeed, when speedup is lower than 1, the computational time of the parallel algorithm exceeds that of the serial algorithm (Fig. 6).

The enhanced time is due to bottleneck. The main performance bottleneck is the communication latency between processors.

To compare these results with those with those obtained in the implementation of the parallel genetic algorithm model with migration in a ring topology (Hao *et al*., 1998) we applied this model to solve the same scheduling problem. Experiences were accomplished in a configuration of shared memory.

The parallel implementation resulted in an improvement of the performance. However, the number of threads remains limited within a shared memory.

So, we have adapted this schema to our hardware configuration. Thus, with a large population size; the genetic algorithm searches the solution space more thoroughly. In a master/slave paradigm, the Master initialises the known parameters of the algorithm: population size, crossover rate, mutation rate and migration frequency.

Then, like the previous schema, the master divides the population in multiple subpopulations in which each computational node carries out genetic operations on its own chromosome set and communicates with only the neighbours in a ring topology.

As can be seen, the obtained results using 2, 4, 6, 8, 10, 12 and 14 sub-populations follow the same reasoning. C_{max} grows gradually by reducing the size of the population (Fig. 7).

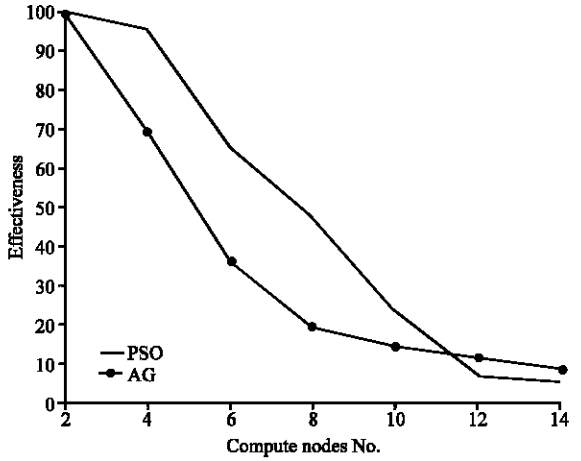

Fig. 8: Parallel distributed (PSO, GA): Parallel effectiveness

This model presents an acceptable acceleration (Fig. 8) and parallel effectiveness which decreases more slowly than in the PSO model but we notice that the parallel distributed PSO model gave better results.

CONCLUSION

This study has presented a parallel implementation of the Particle Swarm Optimization algorithm. The method was validated by an industrial scheduling problem. The PSO is an approach to problems whose solutions can be represented as a point in an n-dimensional solution space. Many improvements of original model PSO were proposed by adjusting parameters in the algorithm. However, Parallel optimization uses multiple computers or processors with an aim of to reduce the elapsed time. Then, a good acceleration can be obtained. Both single

node and parallel implementations of the algorithm have been developed and applied on a high-performance cluster with a fault-tolerant strategy to obtain an efficient and robust parallel scheme. Two widely used metrics have been used for the evaluation of the results and for comparisons with the corresponding results of the parallel distributed genetic algorithm approach. Through the results, we note that cooperation between computational nodes produce improvements but too many processors decreases the parallel efficiency. Also, the PSO model outperforms the genetic algorithm model. Lastly, we can say that distributed system technology, grid computing offers a number of potential uses and benefits for parallel and distributed algorithms and a wide range of computational problems. Nevertheless several parameters should be studied beforehand: the parallel model, the topology, the population's size as well as the infrastructure used play a dominate role in the solution quality and the parallel effectiveness. Future research includes parallelization of other another bio-inspired algorithm: ant-colony optimization.

ACKNOWLEDGMENTS

Our acknowledgments are intended to University of Science and Technology of Oran and in particular to Laboratory of Simulations and Modelling of Industrial Systems which has placed at our disposal the necessary material and a suitable environment for carrying out this study.

REFERENCES

Hao, C., N.S. Flann and D.W. Waston, 1998. Parallel genetic simulated annealing: massively parallel SIMD algorithm. IEEE Trans. Parallel Distributed Syst., 9: 126-136.

Hu, X., R.C. Eberhart and Y. Shi, 2003. Engineering optimization with particle swarm. Proceedings of the IEEE Swarm Intelligence Symposium 2003, April 24-26, IEEE Computer Society, Washington, DC, USA., pp: 53-57.

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proc. IEEE Int. Conf. Neural Networks, 4: 1942-1948.

Li-Ping, Z., Y. Huan-Jun and H. Shang-Xu, 2005. Optimal choice of parameters for particule swarm optimization. J. Zhejiang Univ. Sci. A, 6: 528-534.

Parsopoulos, K.E., D.E. Tasoulis and M.N. Vrahatis, 2004. Multi-objective optimization using parallel vector evaluated particle swarm optimization. Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, February 2004, ACTA Press, Innsbruck, Austria, pp: 823-828.

Pruyne, J. and M. Livny, 1995. Providing resource management services to parallel applications. Workshop on Job Scheduling Strategies for Parallel Processing. Proceedings of the International Parallel Processing Symposium (IPPS'95), April 15.

Requilé, G., 1995. PVM: Parallel virtual machine, les aspects communication. Laboratoire de Mécanique et Génie Civil-CNRS URA 1214-Université Montpellier, 1995. http://1995.jres.org/actes/appli4/1/requile.pdf.

Schi, Y. and R. Eberhart, 1998. Parameter selection in particle swarm optimization. Proceedings of the 7th Internationl Conference on Evolutionary Programming, March 25-27, Springer-Verlag London, UK., pp: 591-600.