Proceedings of the Second EELA-2 Conference
R. Mayo et al. (Eds.)
CIEMAT 2009
© 2009 The authors. All rights reserved

Parallel Hybrid Evolutionary Algorithm in a Grid Environment for the Job Shop Scheduling Problem

Marco Antonio Cruz-Chávez[1], Abelardo Rodriguez-León[2], Erika Yesenia Ávila-Melgar[1], Fredy Juárez-Pérez[1], José Crispín Zavala-Díaz[3], Rafael Rivera-López[2]

[1]CIICAp, [3]FCAeI Autonomous University of Morelos State
Av. Universidad 1001. Col. Chamilpa, C.P. 62209. Cuernavaca, Morelos, México.
{mcruz, juarezfredy,erikay}@uaem.mx
[2]Technological Institute of Veracruz
Miguel Ángel de Quevedo 2779, Formando Hogar, 91860 Veracruz, Veracruz, México
{arleon, rrivera}@itver.edu.mx

Abstract

This paper presents a parallel hybrid evolutionary algorithm executed in a grid environment. The algorithm executes local searches using simulated annealing within a Genetic Algorithm to solve the job shop scheduling problem. Experimental results of the algorithm obtained in the "Tarantula MiniGrid" are shown. Tarantula was implemented by linking two clusters from different geographic locations in Mexico (Morelos-Veracruz). The technique used to link the two clusters and configure the Tarantula MiniGrid is described. The effects of latency in communication between the two clusters are discussed. It is shown that the evolutionary algorithm presented is more efficient working in Grid environments because it can carry out major exploration and exploitation of the solution space.

1. Introduction

The Job Shop Scheduling Problem (JSSP) deals with the resource assignment in manufacturing industries. According to the complexity theory, JSSP is a complex problem, and is classified as NP-Complete [2]. Throughout time, several methods have been proposed to solve complex problems. Finding the global optimum solution using an exact method would take years for an NP-Complete problem. Hence, finding solutions to complex problems with deterministic algorithms is not applicable. In order to deal with NP-Complete problems, the meta-heuristics [3], [4] have been successfully applied. The evolutionary algorithms are meta-heuristics that work efficiently by exploring the solution space to find the best solution. The drawback to these algorithms is that they generally do not completely exploit the solution space. One way to compensate for this problem is to codify a hybrid meta-heuristic that has the capability to completely exploit and explore the solution space, and thereby escape from local optimum solutions.

Although meta-heuristics are efficient techniques to deal with complex problems, generally the computational resources of a computer are not sufficient to obtain results in reasonable computational times with problems in the NP-complete class. It is necessary to make use of computational tools such as clusters or grids, which are a set of interconnected computers that enable distributed processing. Distributed processing allows the use of more computational resources, so that the execution of a polynomial time algorithm can be more efficient.

For the execution of the polynomial algorithm proposed in this paper, a proprietary high performance MiniGrid was created[1]. The MiniGrid was formed by linking two institutions located in different geographic places: the Autonomous University of Morelos State, located in Cuernavaca, Morelos, and the Technological Institute of Veracruz located in Veracruz, Veracruz. Each institution had its own cluster and they were linked to form the Tarantula MiniGrid. The experimental platform created allowed for the execution of parallel programs by means of a passing interface for messages, better known as an *MPI library* [4].

The principal contribution of this work is the presentation of a Parallel Hybrid Evolutionary Algorithm in a Grid Environment with low latency as used to solve the Job Shop Scheduling Problem.

The remainder of this paper is divided into the following sections: Section 2 explains the formation of the job shop scheduling problem. Section 3 presents the description of the hybrid evolutionary algorithm and the process of deployment in a parallel platform with MPI. Section 4 shows the test scenarios used to execute the parallel algorithm. Section 5 describes the experimental results of efficiency and latency of the parallel hybrid evolutionary algorithm execution in a grid environment. Finally, Section 6 presents the conclusions of this work.

2. Job Shop Scheduling Problem

The JSSP is an attempt to establish a scheduling of machines in a manufacturing shop to realize a set of jobs [1]. Each job requires a set of operations to be carried out. The solution to the problem is an optimum sequence of jobs to be executed by each machine, respecting precedence constraints while executing the operations in each job.

$$\text{Min}\left[\max_{j\in O}\left(s_j + p_j\right)\right] \tag{1}$$

$$\forall\ j \in O \qquad s_j \geq 0 \tag{2}$$

$$\forall\ i, j \in O,\ (i \prec j) \in J_k \qquad s_i + p_i \leq s_j \tag{3}$$

$$\forall\ i, j \in O,\ \left(i, j \in M_k\right) \qquad s_i + p_i \leq s_j \vee s_j + p_j \leq s_i \tag{4}$$

Equation (1) is the objective function related to the maximum completion time needed to finish the last operation realized in the manufacturing workshop. Constraint (2) states that the start time of each operation *j* should be greater or equal to zero. Constraint (3) represents the existing precedence between two operations of the same job. Constraint (4) defines the resource capacity between pairs of operations executed by the same machine.

3. Hybrid Evolutionary Algorithm Deployed in Grid Environment

Hybrid evolutionary algorithms are gaining popularity due to their capabilities in handling several real world problems involving complexity, noisy environments, imprecision, uncertainty, and vagueness [5]. Figure 1 shows the solution methodology used to solve the job shop scheduling problem with the proposed hybrid evolutionary algorithm. It presents a genetic algorithm containing a randomly generated feasible population, a selection operator "*the best*" (elitism) [13], a crossover operator Subsequence Exchange (*SXX*) [12] and the iterative mutation operator *SA* [6]. According to their fitness, the operator "the best" selects an average of the best individual values from a population and recombines them, thus producing offspring that will compose the next generation.

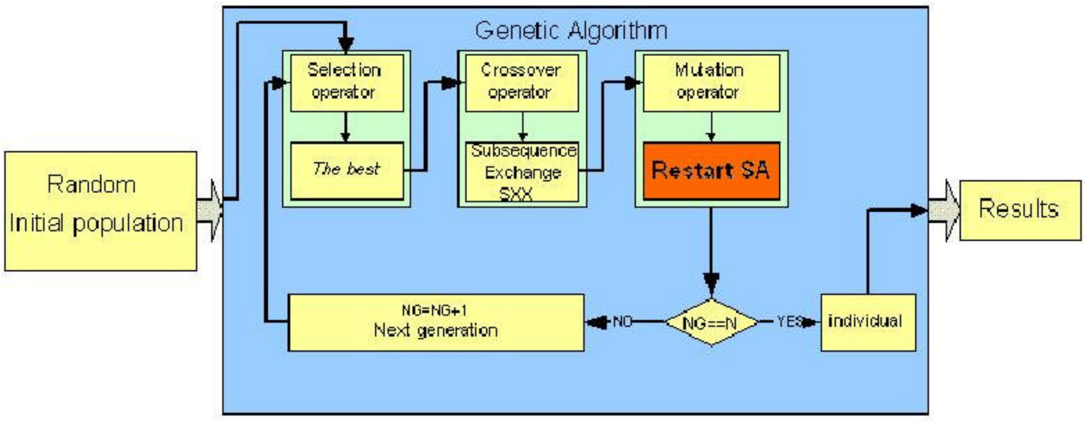

Figure 1. Solution Methodology for the JSSP

Application of a crossover operator allows for exploration of the solution space. An iterative operator is used to mutate [6]. It optimizes the fitness function, applying a mutation in each individual. It helps to find the best individual by using an iterated local search with the Simulated Annealing process (SA). The previously described procedure is repeated for each generation of the population.

The hybrid evolutionary algorithm is implemented by using distributed processing and parallel techniques. The MiniGrid consists of two clusters. Each cluster contains some nodes, which have one or more processors. The processors are involved in the implementation of parallel algorithms using the Interface Passing Messages library (MPI). The parallel hybrid evolutionary algorithm implemented uses the master-slave paradigm, which maintains the sequence of the original algorithm [7].

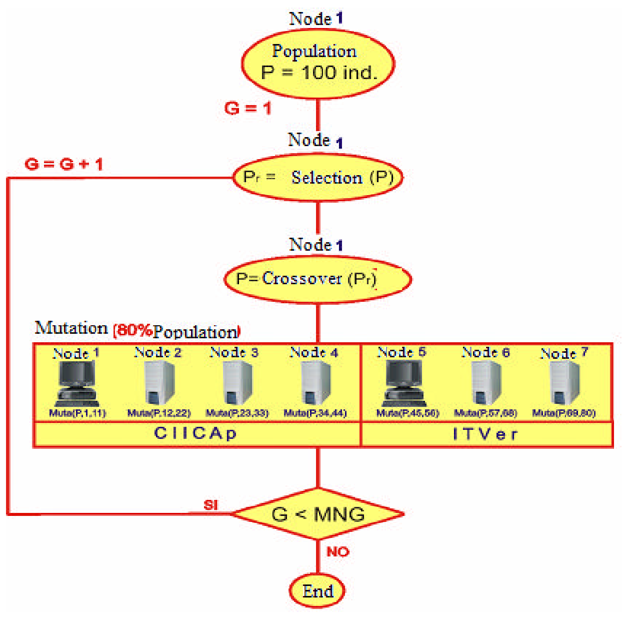

Figure 2. Hybrid evolutionary algorithm execution for the JSSP in MiniGrid UAEM-ITVer

The main idea of the proposed parallelization of the evolutionary algorithm is to first allocate tasks to processors with lower communication requirements. The tasks are divided proportionally according to the available processors. Figure 2 shows the use of nodes in the MiniGrid for the algorithm. First, an initial feasible population is created in a master node. The master node carries out selection and crossover operations for genetic algorithms; it then sends a sub-population to the slaves' nodes on the MiniGrid. Based on experimental tests, the mutation operator has been identified as the most time consuming procedure. To decrease the execution time, the mutation function is divided proportionally according to the available nodes in the MiniGrid. That means that a process number is generated based on the number of processors available in the MiniGrid (1:1). Each processor executes a set of iterations using the Restart SA technique (iterative mutation). A Restart SA is executed for each individual of a sub-population that is to be mutated. The slave nodes return the newly mutated individuals to the master. Next, the master node collects all the individuals mutated by the slaves' nodes and uses them to create a new population. This process continues until an optimal solution is found or a given number of generations are created. For example, for a population of 100 with an 80% mutation rate, and 20 processors, each processor would mutate 4 individuals, as shown in Figure 2.

The MPI library was used for sending messages for the parallel hybrid evolutionary algorithm. The MPI library enables the transfer of complex data types. However, the MPI library has no mechanism to transfer dynamic data types. It only recognizes the primitive data types used in C language. For every primitive data type in C language as int, char, long, double, and so on, there is an equivalent MPI data type: MPI_INT, MPI_CHAR, MPI_LONG, MPI_DOUBLE. When using dynamic data types, there is not an equivalent MPI data type, so it is necessary to develop a conversion procedure, because the hybrid algorithm uses dynamic data types.

The conversion procedure necessary to transfer data is known in some platforms as serialization and de-serialization. It consists of serializing the dynamic data structure, sending it, receiving it, and reconstituting it. The conversion process is complex and tedious. Currently, an automatic tool called *Automap* [10] is proposed to do the conversion automatically. Once the data is converted to MPI data types, it is easy to send dynamic data structures through the grid.

4. Test Scenario

The MiniGrid is formed by joining two high-performance cluster computers (HPC), each computer belonging to a different domain. The clusters are integrated as a single parallel machine. They need for integration of the HPC is due to their distant geographic locations. The integration allows for local management of both clusters despite the fact that they are independent entities.

The basic components in the construction of a high performance cluster with support for MPI programming were based on the installation and configuration of the following software and services on Red Hat Enterprise Linux 4. 1) Configuration of the private network. 2) Set up of a network file system using Network File System (NFS). 3) Set up of the Network Information Service (NIS). 4) Set up of public keys to enable the execution of remote jobs. 5) Set up of the MPI for use of OpenMPI or MPICH, which is the open source implementation of MPI-1 standard and MPI-2. 6) Configuration of Ganglia [11] for real time monitoring of the cluster. 7) Configuration of Torque settings to manage the cluster resources. 8) Set up of MAUI for planning jobs in the cluster.

The integration of the components shown in Table 1 is essential for the creation of high-performance clusters. The Mini-Grid consists of two or more high-performance clusters, which are geographically remote. An important element of the MiniGrid is a link that joins the two clusters by a *Virtual Private Network* (VPN, network-to-network). For the MiniGrid implemented in this study, an Open VPN was used instead of routers (via hardware). The joined clusters were in different subnets.

Table 1. Infrastructure of Hardware and Software for the Grid

GRID TARANTULA ---- SOFTWARE ---	
Red Hat Enterprise Linux 4, Compiler gcc versión 3.4.3, OpenMPI 1.2.8, MPICH2-1.0.8, Ganglia 3.0.6 NIS ypserv-2.13-5, NFS nfs-utils-1.0.6-46, OpenVPN, Torque + Maui.	
CLUSTER CIICAP	**CLUSTER ITVER**
Switch Cisco C2960 24/10/100	Switch 3com 8/10/100
Master node Pentium 4, 2793 MHz, 512 MB RAM, 80 GB HD, 2 network cards 10/100 Mb/s	Master node Pentium 4 Dual Core, 3200 Mhz, 1 GB RAM, 80 GC HD, 1 network card 10/100 Mb/s
9 slave nodes Intel® Celeron® Dual Core, 2000MHz, 2 GB RAM, 160GB HD, 1 network card 10/100 Mb/s	2 slave nodes Pentium 4 Dual Core, 3201 Mhz, 1 GB RAM, 80 GB HD,1 network card 10/100 Mb/s

5. Experimental Results

The latency computation is based on the transfer rates between the two clusters: cluster.ciicap.edu.mx and nopal.itver.edu.mx. The process used to compute latency in the two clusters is described in detail in [9].

Figure 3 shows the latency obtained for the Tarantula MiniGrid throughout the day, from 12 AM to 6 PM. The results shown are the average values over a period of five days. It can be observed that the latency decreases as the day progresses. At 12 hours, higher latencies are reported, measuring between 45 and 40 seconds. At 18 hours, a latency of less than 0.4 seconds is observed. The latency tends to decrease as the day progresses. This indicates that in the afternoon the data transfer is more efficient.

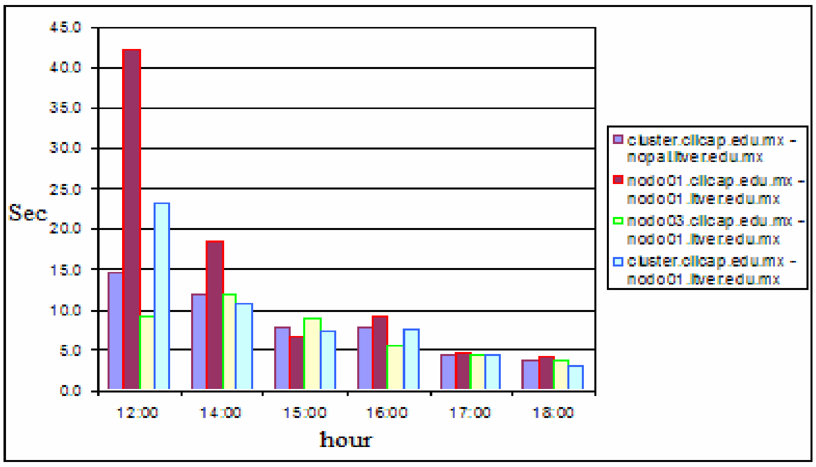

Figure 3. Tarantula MiniGrid Latency

In order to measure the efficiency of the hybrid Evolutionary Algorithm, it was executed for two benchmarks taken from the OR Library [8]. The algorithm was executed from the master node of the CIICAp cluster. The first executed benchmark was LA40 and it consisted of 15 jobs and 15 machines. The stop criteria were the number of generations created. The input data is presented in Table 2.

The obtained results of the algorithm execution in the MiniGrid are presented in Table 3. The average time of execution in the nodes (cores) of each cluster and the best makespan was presented after it finished 35 generations. Logical reasoning indicates that if the number of processors is increased then the population size could be increased, while requiring a similar amount of time. Increasing the number of processors would let the algorithm explore a larger portion of the solution space and find better solutions in less time. This shows the importance of having a large GRID that includes computational resources for high performance applications, such as the algorithm presented here.

Table 2. Input Data for the Hybrid Genetic Algorithm for the LA40 and YN1 Instance

Variable	**LA40**	**YN1**
Initial Temperature for Simulated Annealing	25	20
Final Temperature for Simulated Annealing	1.0	1.0
Coefficient Temperature for Simulated Annealing	0.99	0.955
Markov Chain Length	210	5000
Given Upper Bound for the JSSP	1222	885
Restarting number for each individual in Simulated Annealing	20	20
Good Population Selection (%)	40	40
Bad Population Selection (%)	5	5
Population to mutate (%)	80	80
Number of Generations for Genetic Algorithm	35	6

Table 3. Results of the Parallel Hybrid Evolutionary Algorithm for Solving LA40 Instance

Cluster	nodo0x.ciicap.edu.mx	Nodo0x.itver.edu.mx
Node	0,1,2,3,4,5,6,7,8,9	0,1,2
Time (sec)	12618	15413
Makespan	1234	1243

Table 2 indicates that the number of restarts for SA for each individual was 20. According to the restart number, in the worst case, the average execution time of the mutation in each processor for each generation was 345 seconds vs. 25 seconds for the worst latency registered in the MiniGrid. Based on this observation, it can be determined that latency is not significantly important because the time the processors were in use was greater than the communication time between clusters (93.2% vs. 6.8%).

The second executed benchmark was an instance of YN1 [8]. It had 20 jobs, 20 machines and 400 operations. Table 2 presents the input data and Table 5 shows the obtained results. The average latency detected was approximately 25 seconds. It indicates that, in the worst case, the total latency in execution time of the algorithm was 150 seconds according to the number of generations. Table 4 presents the results obtained for the algorithm execution for each processor in the MiniGrid. The average time of execution in the nodes (cores) of each cluster and the best makespan is presented after finishing 6 generations. If it were possible to increase the number of processors, then the population size could be increased, without a dramatic increase in time. Increasing the number of processors would allow the algorithm to explore a larger portion of the solution space and find a better solution in less time. This shows the importance of having a larger GRID that includes computational resources for high performance applications, such as the algorithm presented here. Table 2 indicates that the number of restarts for SA for each individual was 20. According to the restart number, in the worst case, the average execution time of SA in each processor for each generation was 4,567 seconds vs. 25 seconds for the worst latency registered in the MiniGrid. Based on this observation, it can be determined that latency is not significantly important because the time processors were in use was greater than communication time between clusters (95.5% vs. 0.5%).

Table 4. Results of the parallel hybrid evolutionary Algorithm for solving YN1 instance

Cluster	nodo0x.ciicap.edu.mx	Nodo0x.itver.edu.mx
Node	0,1,2,3,4,5,6,7,8,9	0,1,2
Time (sec)	19175	25617
Makespan	933	935

6. Conclusions

It can be concluded that the use of a Grid platform for this type of problem is highly recommendable for solving complex problems in a short time. The time needed to search and find solutions diminishes, as the number of available processors in the Grid increases and there is little communication between processors. The latency varies in the MiniGrid, the time tends to decline in the evenings, so it is recommendable to work in the afternoons when implementing algorithms that use long communication times.

In order to reduce communication between processors, it is necessary to define the structure of the algorithm in a way that allows it to work with a small latency in the MiniGrid. The latency does not considerably affect the parallelization because the large tasks of the algorithm mean the processors work for extended periods in order to complete execution of the program. Thus, there are minimal communications. The algorithm requires communication only to send and receive the sub-populations, and this prevents the latency from negatively influencing the execution of the algorithm. To increase efficiency of the algorithm in the grid environment, communication between Grid nodes should be minimized. The algorithm can be performed using all resources of the Grid presented in this work.

Acknowledgements

This work was supported by project CUDI-CONACYT 2007.

References

[1]. Roy and Sussman, Les problemes d’ordonnancement avec contraintes disjonctives, Note D.S. no 9 bis, SEMA, Paris, France, December 1964.
[2]. C.H. Papadimitriou and K. Steiglitz, Combinatorial optimization: algorithms and complexity, Prentice Hall Inc., USA. ISBN 0-13-152462-3, 496 pp., 1982

[3]. Díaz, A., Glover, F., Ghaziri, H.M., et al, Optimización Heurística y Redes Neuronales. Madrid, Paraninfo, (1996).

[4]. F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533-549, 1986.

[5]. Grosan, Crina, Abraham, Hybrid Evolutionary Algorithms, Series: Studies in Computational Intelligence, Vol. 75, Ajith; Ishibuchi, Hisao (Eds.), 2007, XVI, 404 p. 207 illus., Hardcover ISBN: 978-3-540-73296-9

[6]. M. A. Cruz-Chávez, J. Frausto-Solís, Simulated Annealing with Restart to Job Shop Scheduling Problem Using Upper Bounds, Lecture Notes in Artificial Intelligence, Springer Verlag Pub., Berlin Heidelberg, ISSN: 0302-9743, Vol. 3070, pp. 860 - 865, 2004.

[7]. Cantu-Paz, E., A Survey of Parallel Genetic Algorithms, Technical Report IlliGAL 97003, University of Illinois at Urbana-Champaign, 1997.

[8]. J. E. Beasley. OR-Library: Distributing test problems by electronic mail, Journal of the Operational Research Society, Vol. 41, No. 11, 1069-1072, 1990. Last update 2003.

[9]. J. D. Sloan, Network Troubleshooting Tools, ISBN 10: 0-596-00186-X, ORelly, pp. 364, 2001.

[10]. Michel, M. and Devaney, J. E. 2000. A Generalized Approach for Transferring Data-Types with Arbitrary Communication Libraries. In *Proceedings of the Seventh International Conference on Parallel and Distributed Systems: Workshops* (July 04 - 07, 2000). ICPADS. IEEE Computer Society, Washington, DC, 83.

[11] Ganglia Monitoring System, Monitoring clusters and Grids since the year 2000 http://ganglia.info/, September 2009.

[12] P. J. Zalzala, and Flemming. Zalsala, A.M.S. (Ali M.S.), ed., Genetic algorithms in engineering systems /Edited by A.M.S. Institution of Electrical Engineers, London, 1997.

[13] O. Al Jadaan, L. Rajamani, C. R. Rao, Improved Selection Operator for GA , Journal of Theoretical and Applied Information Technology, ISSN 1992-8645, ARPA Press, 269-277, 2008.