Gridification of Genetic Algorithm with Reduced Communication for the Job Shop Scheduling Problem[1]

Marco Antonio Cruz-Chávez[1], Abelardo Rodríguez-León[2], Erika Yesenia Ávila-Melgar[1], Fredy Juárez-Pérez[1], Martín H. Cruz-Rosales[4], Rafael Rivera-López[2]

[1]CIICAp, [3]FCAeI, [4]FC, Autonomous University of Morelos State
Av. Universidad 1001. Col. Chamilpa, C.P 62209. Cuernavaca, Morelos, México.
{mcruz, erikay, juarezfredy, mcr}@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 conFigureure 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.

Keywords: *evolutionary algorithm, grid environment, Job Shop Scheduling Problem, Message Passing Interface Library*

1. Introduction

The Job Shop Scheduling Problem (JSSP) deals with the resource assignment in manufacturing industries. The goal of JSSP is to schedule the job operations in a number of machines. Each job consists of a sequence of operations with a predefined precedence order. Each machine can process only one operation at a time.
The Job Shop Scheduling Problem attempts to find the "Makespan," which is the schedule that requires the minimum amount of time. An initial schedule is obtained for a given set of *n* jobs to be scheduled on *m* machines.

According to the complexity theory JSSP is a complex problem; it is classified as NP-Complete [1] and it is widely studied.

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

[1] This work was supported by projects FOMIX-CONACYT MOR-2009-C02-120102 and CONSOLIDACIÓN 2008-18-07-10-197.

applicable. In order to deal with NP-Complete problems, the meta-heuristics [2], [3] have been successfully applied.

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, which includes evolutionary and simulated annealing algorithms, and that has the capability to completely exploit and explore the solution space, thereby providing for an escape from local optimum solutions.

Simulated Annealing is a heuristic that attempts to simulate the process undergone by misplaced atoms in a metal when it is heated and then slowly cooled. There is a mathematical relation that explains the convergence of SA to the problem's global optimum when the temperature is decreased infinitely slowly [4].

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 and parallel processing. Distributed processing allows the use of more computational resources located in different geographic places. Parallel processing allows dividing computational tasks among the available processors. The execution of a polynomial time algorithm can be more efficient with the use of techniques like distributed and parallel processing.

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 the execution of parallel programs by means of a Message Passing Interface library (MPI library) [16].

The principal contribution of this work is the development of a Parallel Hybrid Evolutionary Algorithm in a Grid Environment with Reduced Communication to solve the Job Shop Scheduling Problem. The developed algorithm is not dependent of latency when it runs in the created MiniGrid, because there is reduced communication between Grid nodes.

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, bandwidth, 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 [5]. 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.

$$\mathbf{Min}\left[\frac{\max}{j\in\mho}\left(s_j + p_j\right)\right] \quad \mathbf{(1)}$$

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

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

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

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 in Grid Environment

Evolutionary Algorithms are gaining popularity due to their capabilities in handling several real world problems involving complexity, noisy environments, imprecision, uncertainty, and vagueness [6].

Simulated Annealing is a programming method that has been successfully applied to many combinatorial optimization problems.

In order to avoid getting stuck in local optimum solutions, a hybrid evolutionary algorithm is implemented. This hybrid algorithm consists of a combination of both methods, evolutionary and simulated annealing.

Evolutionary algorithms completely explore the solution space while simulated annealing exploits it. The hybrid evolutionary algorithm can therefore completely exploit and explore the solution space, thereby providing escape from local optimum solutions.

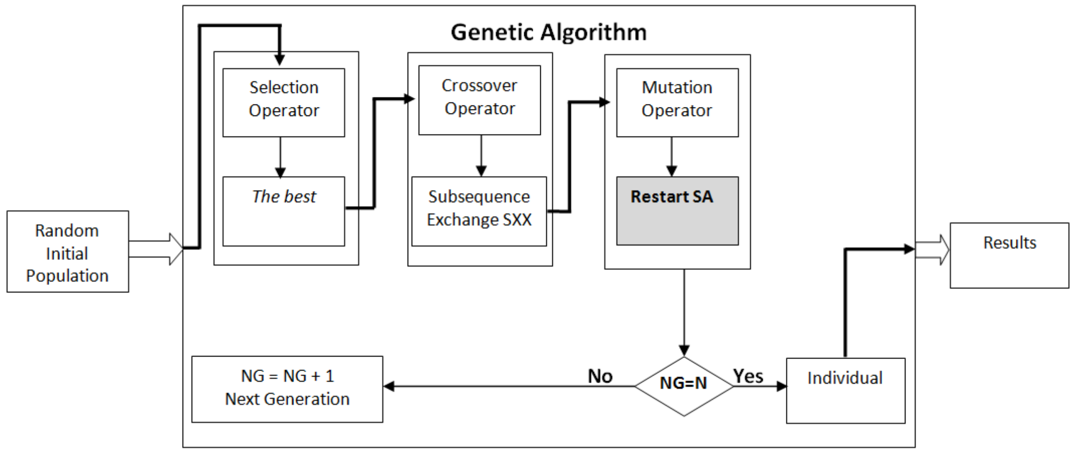


Figure. 1. Solution Methodology for the JSSP

Figureure 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. It is known that the initial feasible population, which is not necessarily the best one, finds good individuals that can then be used in future generations.

The selection operator used is called "*the best*" (elitism) [7]. It consists of taking the best individuals from the population, based on 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.

The crossover operator Subsequence Exchange (*SXX*) [8] allows exploration of the solution space. It consists of searching for exchangeable subsequence pairs of genes in an individual, and interchanging them to produce an individual with a new sequence.

The iterative mutation operator is defined by Simulated Annealing with Restart (SAR) [9]. It consists of executing a set of SA, as shown in Figureure 2. Each simulated annealing that is executed involves the iteration of the SAR algorithm. Each repetition begins using a different schedule. The idea of beginning with different schedules for each repetition of SAR originated from experimental tests that were conducted in the experimental tests, it was detected that even though SA allows the search for a global optimum, at some point, low temperatures eliminate this possibility and the search finds a local optimum. More solution variability is obtained by beginning with multiple schedules, as opposed to beginning with a single schedule. The restart of the simulated annealing algorithm leaves from a different point in the solution space each time it is repeated.

Restarting the algorithm enables a search for global optimums by using a new schedule in each repetition of SAR. This allows a different part of the solution space to be explored when SAR is at a local optimum. This not only increases the probability of finding a global optimum, but also increases the time of the search. In the SAR algorithm, at the beginning of each simulated annealing, a UB (Upper Bound) is established in order to randomly obtain the schedule with which to start the process.

The iterative operator used to mutate optimizes the fitness function applying a mutation in each individual. It helps to find the best individual by using an iterated local search. The previously described procedure is repeated for each generation of the algorithm.

```
1. Given initial iteration $k=0$, initial values of $S_f$, $T_f$, $\beta$
2. Beginning of annealing $k=k+1$:
  3. $S=S_0 \leq Upper\ Bound$, $T=T_0$, initial $S_c$
  4. While the final temperature Tf is not reached,
    5. While equilibrium is not reached:
      •Generate a state $S\prime$ by means of a perturbation in $S$
      •if $f(S\prime)-f(S) \leq 0$ then $S=S\prime$
      •if $f(S\prime)-f(S) > 0$ the state is accepted with
              the probability: $P_{accept} = e^{-\left(\frac{f(S\prime)-f(S)}{T}\right)}$
      •With a randomly generated number $\alpha$
                  evenly distributed between $(0,1)$
      •if $\alpha < P_{accept}$ then $S=S\prime$
      •if $S<S_c$ then $S_c=S$
      •If the equilibrium does not exist, return to 5
   $T=T*\alpha$
  The best configuration is stored, if $S_c<S_f$ then $S_f=S_c$
  If $T \geq T_f$, return to 4
If $k<maxiter$, return to 2 to begin a new annealing
The solution is $S_f$
```

Figure. 2. Simulated Annealing with Restart Procedure, taken from [9]

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 main idea of the 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.

The parallel hybrid evolutionary algorithm implemented uses the master-slave paradigm, which maintains the sequence of the original algorithm [10]. The original population is divided into subpopulations. Each subpopulation is assigned to a processor. Each processor is responsible for managing and executing the heuristic. Using the assigned generation, the best individuals are sent to other processors.

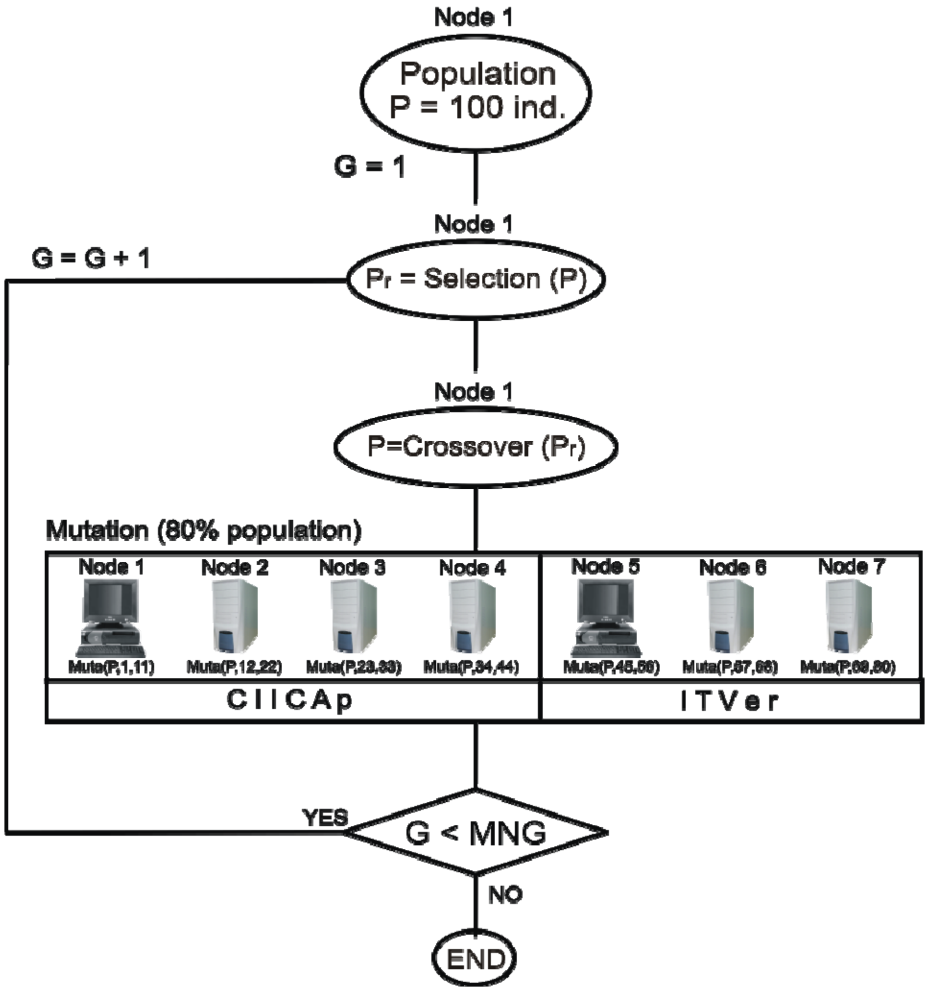


Figure. 3. Hybrid Evolutionary Algorithm Execution for the JSSP in MiniGrid UAEM-ITVer

Figureure 3 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, is created. For example, for a population of 100 with an 80% mutation rate, and 20 processors, each processor would mutate 4 individuals, as it is shown in Figureure 3. This algorithm has Reduced Communication because not exist communication between slave nodes. Communication between master node and slave nodes only exist. This makes that latency in MiniGrid have very little impact in the Hybrid Evolutionary Algorithm efficiency. The maximum execution time of the algorithm appends in slave nodes that execute the Simulated Annealing with Restart Procedure.

The MPI library was used to send messages for the parallel hybrid evolutionary algorithm. The MPI library enables the transference 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 create a conversion procedure, because the developed 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, if it is done manually, is complex and tedious.

Currently, an automatic tool called *Automap* [11] 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 clusters of computers (HPC), each computer belonging to a different domain. The clusters are integrated as a single parallel machine. The 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 conFigureuration of the following software and services on Red Hat Enterprise Linux version 4. The procedure consists of the following steps: 1) ConFigureuring the private network, 2) Setting up a network file system using Network File System (NFS), 3) Setting up the Network Information Service (NIS), 4) Setting up public keys to enable the execution of remote jobs, 5) Setting up the MPI for use of the OpenMPI or MPICH, which is the open source implementation of MPI-1 standard and MPI-2, 6) ConFigureuring Ganglia [12] for real time monitoring of the cluster, 7) ConFigureuring Torque settings to manage the cluster resources, and 8) Setting up MAUI for planning jobs in the cluster.

Table 1. Infrastructure of Hardware and Software for the Grid

GRID TARANTULA ---- SOFTWARE ---	
Red Hat Enterprise Linux 4, Compiler gcc version 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

The integration of the components showed in Table 1 is essential for the creation of high-performance clusters, Figureure 4. 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.

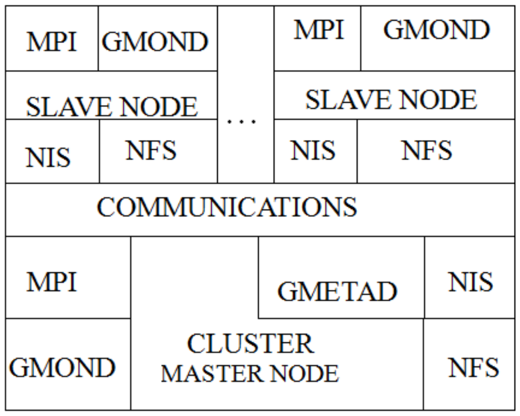


Figure. 4. High Performance Cluster Components

5. Experimental Results

The latency computation is based on the transference rates between 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 [13].

Figureure 5 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 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 transference is more efficient.

The size of the data to be sent throughout the network depends on the data structure size. An individual of the JSSP is represented by a data structure with a size of 12,800 bytes. Therefore, each processor of the cluster CIICAp sends 51,200 bytes.

The parallel hybrid evolutionary algorithm needs to send 204,800 bytes from the CIICAp cluster to the Nopal cluster during each iteration of the evolutionary algorithm. The data is sent through the network using Internet 2 technology.

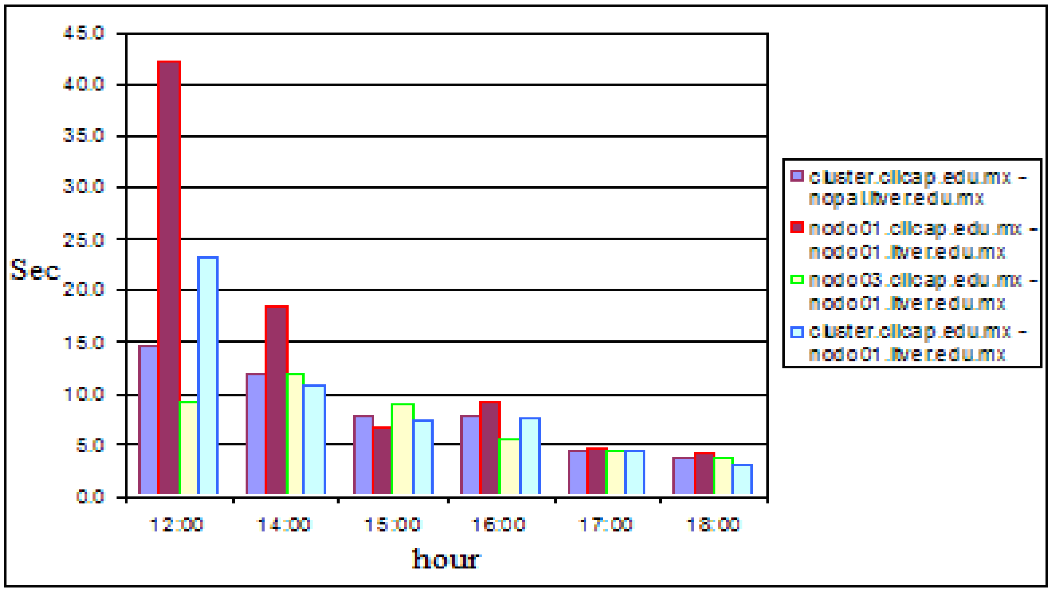


Figure.5. Tarantula MiniGrid Latency

In order to measure the efficiency of the developed algorithm, it was executed for two benchmarks taken from the OR Library [14]. The algorithm was executed from the master node of the CIICAp cluster.

The first executed benchmark was LA40; 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.

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 processors were in use was greater than the communication time between clusters (93.2% vs. 6.8%).

Table 2. Input Data for the Hybrid Genetic Algorithm for the LA40

Variable	**LA40**
Initial Temperature for Simulated Annealing	25
Final Temperature for Simulated Annealing	1.0
Coefficient Temperature for Simulated Annealing	0.99
Markov Chain Length	210
Given Upper Bound for the JSSP	1222
Restarting number for each individual in Simulated Annealing	20
Good Population Selection (%)	40
Bad Population Selection (%)	5

Population to mutate (%)	80
Number of Generations for Genetic Algorithm	35

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 without increasing the required time to be executed.

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 4. Input Data for the Hybrid Genetic Algorithm for the YN1

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

The second benchmark executed was an instance of YN1 [8]. Table 4 presents the input data. The benchmark considers 20 jobs, 20 machines and 400 operations. The stop criteria were the number of generations created.

Table 5 presents the results obtained for the algorithm execution for each processor in the MiniGrid. The average time of execution in the nodes 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 here presented.

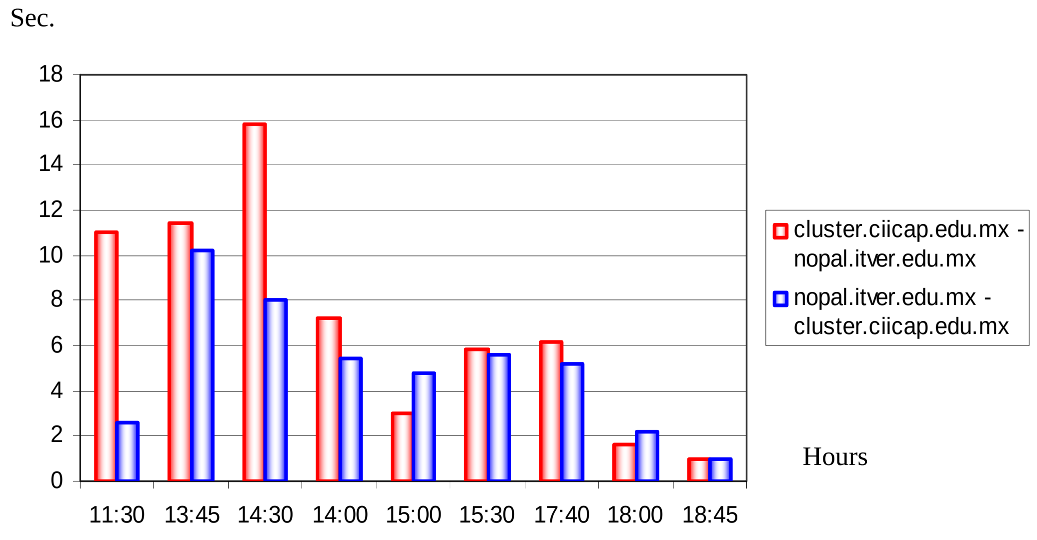
In table 4, it can be seen that the number of restarts for SA for each individual was 20. According to the restart number, in the worst case, the average execution time of SA in each processor for each generation was 4,567 seconds vs. 25 seconds for the worst latency registered in the MiniGrid. Based 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 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 5. 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

Finally, in order to obtain the response time, a file of 200 kb was sent from cluster CIICAp to cluster Nopal. The time was recorded after the file was written. These experiments were carried out for three consecutive days for between 9 and 18 hours. The time required for transfer is shown in Figureure 6. It can be seen that latency increases in the morning and decreases as the day progresses.



Figureure 6. Latency when Transfering a File

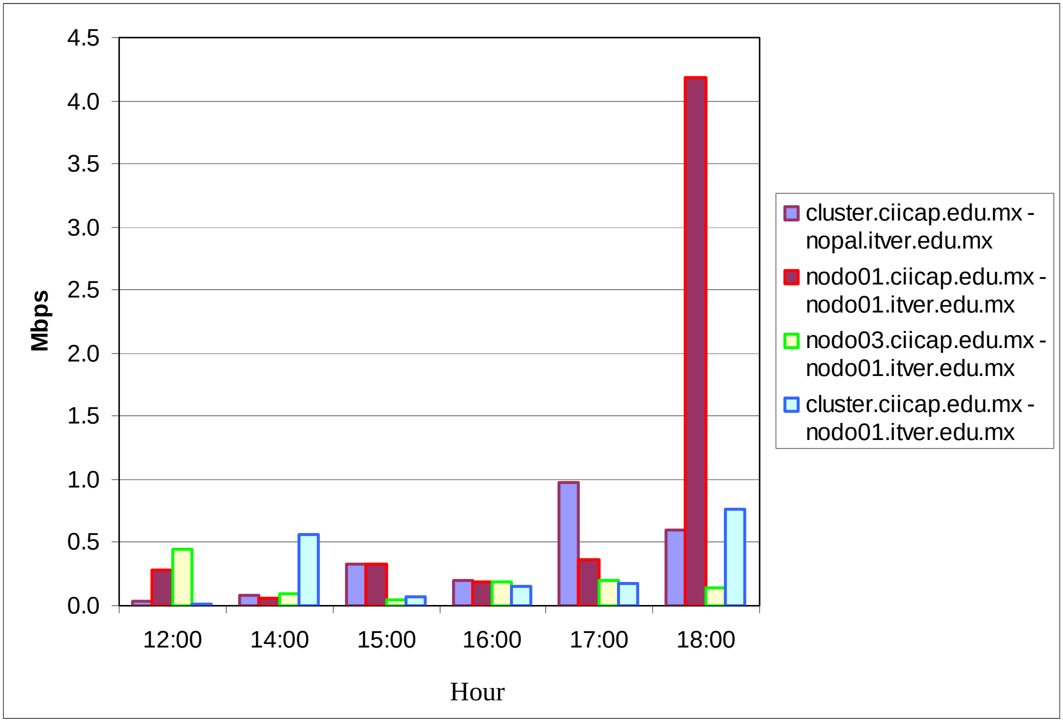
5.1 Computing bandwidth in the Grid

When implementing parallel applications, it is important to know the throughput of the network in order to compute the bandwidth created in the Mini-Grid.

The computations were done taking into consideration the virtual private network, the master nodes, and the slaves' nodes. Various computations were performed during the course of a single day. In addition, calculations were performed for three days.

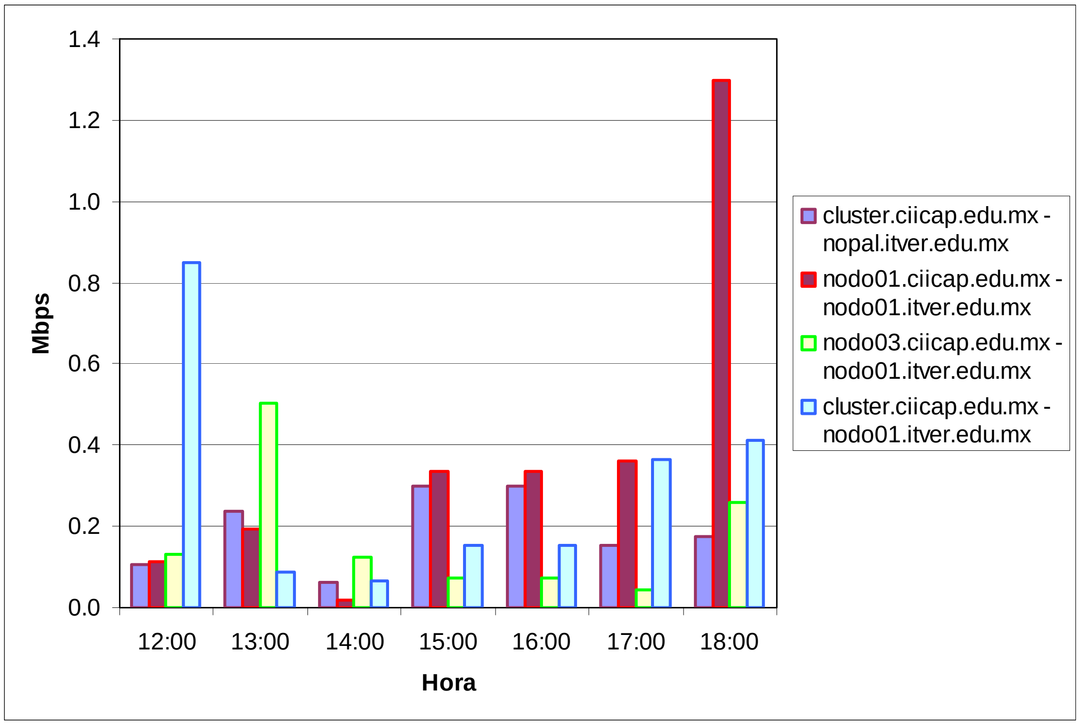
In order to get an approximation of the throughput [15], the ping command was used. The ping command is an excellent tool for testing the network connectivity. The destiny node sends a package, requiring the receptor to answer with a package of the same size. The time employed to send a package from source to destiny node and vice versa is known as Round Trip Time (RTT).

The procedure used here involved sending two packages of different sizes four times, and computing the media.
The transfer velocities are adjusted to the velocity of the VPN between the cluster.ciicap.edu.mx and nopal.itver.edu.mx nodes. Data is sent through the master nodes before it can be sent to the salves nodes connected to the VPN.
Figureure 7 presents experimental measurements of the bandwidth between the clusters CIICAp and Nopal using an Internet 2 connection. Specifically, measurements are taken between pairs of nodes; one corresponding to the CIICAp cluster and one corresponding to the Nopal cluster. It can be seen that the bandwidth measured at different times during the day oscillates between 0.1 and 1 Mbps. It can be noted that the bandwidth between two nodes almost never remains stable, and is inferior to the maximum reported bandwidth of Internet 2. It can be observed that bandwidth increases as the day progresses; probably because Internet 2 is used less in the evenings.

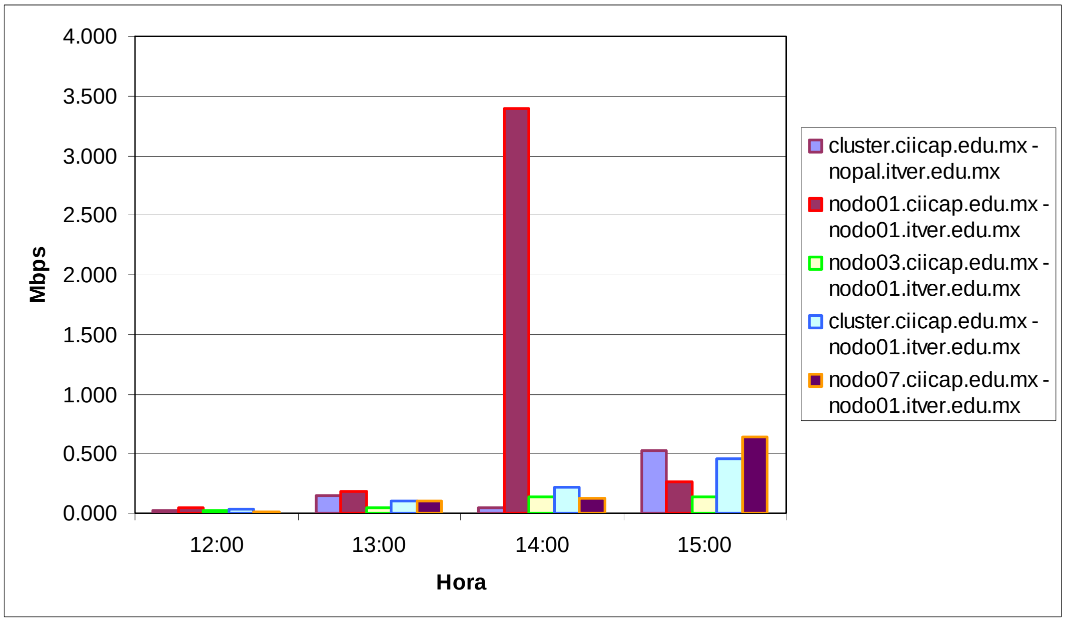


Figureure 7. Bandwidth MiniGrid CIICAp-Nopal Day 1

Figureure 8 presents experimental measurements of the bandwidth between the clusters CIICAp and Nopal using an Internet 2 connection. Specifically, measurements are taken between pairs of nodes; one corresponding to the CIICAp cluster and one corresponding to the Nopal cluster. It can be seen that bandwidth taken at different times during the day oscillates from 0.1 and 0.4 Mbps. It can be noted that the bandwidth between two nodes almost never remains stable and is inferior to the maximum reported bandwidth of Internet 2, which is 34 Mbps. It can be observed that bandwidth decreases from 12 to 14 hours and it increases from 14 to 18 hours.



Figureure 8. Bandwidth MiniGrid CIICAp-Nopal Day 2



Figureure 9. Bandwidth MiniGrid CIICAp-Nopal Day 3

Figureure 9, presents experimental measurements of the bandwidth between the clusters CIICAp and Nopal using an Internet 2 connection. Specifically, measurements are taken between pairs of nodes; one corresponding to the CIICAp cluster and one corresponding to the Nopal cluster. It can be seen that the bandwidth taken at different times during the day oscillates between 0.1 and 0.4 Mbps. It can be seen that bandwidth between two nodes almost never remains stable and is inferior to the maximum reported bandwidth of Internet 2, which is 34 Mbps. It can be observed that bandwidth increase as the day progresses.

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.

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.

References

[1]. C.H. Papadimitriou and K. Steiglitz, Combinatorial optimization: algorithms and complexity, Prentice Hall Inc., USA. ISBN 0-13-152462-3, 496 pp., 1982

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

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

[4]. Deekers, A., Aarts, E.: Global optimization and simulated annealing. Math. Program. 50, 367–393 (1991)

[5]. Roy and Sussman, Les problemes d'ordonnancement avec contraintes disjonctives, Note D.S. no 9 bis, SEMA, Paris, France, December 1964.

[6]. 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

[7] 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.

[8] 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.

[9]. 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.

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

[11]. 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 [12] Ganglia Monitoring System, Monitoring clusters and Grids since the year 2000 http://ganglia.info/, September 2009.

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

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

[14]. 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.

Systems: Workshops (July 04 - 07, 2000). ICPADS. IEEE Computer Society, Washington, DC, 83.

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

[16]. Message Passing Interface Forum, http://www.mpi-forum.org/docs/docs.html

Authors

Marco Antonio Cruz-Chávez received the Doctor degree in Computer Sciences from Tec de Monterrey in 2004. He works from 2004 like professor-researcher in the Research center in Engineering and Applied Sciences (CIICAP) of the Autonomous University of the Morelos State (UAEM). He is a National Researcher of Mexico (SNI). Leader of Project Grid Morelos, Computing Lab of High Performance. Manager High Performance Grid Morelos. Active Member of Joint Research Unit México, contact in UAEM. He has 19 international publications and 11 nationals, from the 2005 he is reviewer of International Journal of Production Re-search.

Abelardo Rodríguez-León. Received the B.E. degree in computer systems engineering from Technological Institute of Veracruz (ITVer), in 1989, the M.S. degree in computer sciences from Universidad Veracruzana, in 1992, and the Ph.D. degree in computer science, from Universidad Politécnica de Valencia, Spain in2007. He is currently a Professor in Computer Science in the Computer and Systems Department of the ITVer, in MEXICO. His research interests include grid computing, parallel programming and optimization.

Erika Yesenia Ávila-Melgar. Currently I am studying PhD. At Centre of Research in Engineering and Applied Sciences. I am interested in studying software optimization, parallel computing, clustering and, distributed systems. I am also interested in water distribution networks problem, which consist of finding the best way to convey water from sources to users. It is a problem classified in the NP-Complete class and it is been solved through evolutionary algorithm in grid environments.

Fredy Juárez-Pérez. Computer Systems Engineer by the Technological Institute of Ciudad Madero, México. Master in Computer Science by the National Center for Research and Technological Development, México. PhD student in Electrical Technology at the Center for Research in Engineering and Applied Sciences at the Autonomous University of Morelos, México. Manager Grid Computing Lab of High Performance. Manager High Performance Grid Morelos. Active Member of Joint Research Unit México. Responsible for the installation of the site with g-Lite for integration into the Grid (EELA2 and GISELA project). Areas of interest: Scheduling and Combinatorial Optimization. Cluster Computing and Grid Computing. Programming parallel and distributed memory Distributed and Web Systems.

Martín H. Cruz-Rosales. Ph.D. From Autonomous University of Morelos State. Professor Martin is a researcher and lecturer at the Faculty of Sciences of the Autonomous University of Morelos State. His research is in the area of combinatorial optimization. He has several publications and articles on scheduling algorithms and the he has also given several lectures and seminars.

Rafael Rivera-López received the B.E. degree in computer systems engineering from Technological Institute of Veracruz (ITVer), MEXICO, in 1989, and the M.S. degree in computer sciences from Technological Institute of Superior Studies of Monterrey (ITESM), Morelos, MEXICO, in 2000. He is working toward the Ph.D. degree in computer science, ITESM. He is currently a Professor in Computer Science in the Computer and Systems Department of the ITVer. His research interests include optimization, artificial intelligence and robotics.