# A Comparison between Centralized and Decentralized Genetic Algorithms for the Identical Parallel Machines Scheduling

**Susana C. Esquivel**
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis
esquivel@unsl.edu.ar

and

**Claudia R. Gatica**
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis
crgatica@unsl.edu.ar

**Abstract**

Identical parallel machines problems ($P_m$) involve task assignments to the system's resources (a machine bank in parallel). The basic model consists of *m* machines and *n* tasks. The tasks are assigned according to the availability of the resources, following some allocation rule. In this work, the minimization of some objectives related to the due dates such as the maximum tardiness ($T_{max}$) and the average tardiness ($T_{avg}$) were dealt with centralized and decentralized evolutive algorithms (EAs). In order to test our algorithms we used standard benchmarks. The main goal of this research was determinate the quality of the results obtained with a centralized GA and three decentralized GAs used to solve parallel machines scheduling problems. The results were compared using the ANOVA statistic method.

**Keywords:** Parallel machines scheduling, centralized evolutive algorithms, decentralized evolutive algorithms.

## 1 INTRODUCTION

Unrestrict parallel machines scheduling problems are common in production systems. In such systems it is usual to force the minimization of the objectives based on the due dates such as the *maximum tardiness ($T_{max}$)* and the *average tardiness ($T_{avg}$)*.

For these scheduling problems, the literature offers a set of dispatch rules and heuristics to provide reasonably good solutions in short times. Some heuristics behave better than others, depending on the particular instance of the problem. The EAs have been successfully applied to solve such scheduling problems as in [4], [6], [7], [10], [11] and [12]. The EAs are blind search population-based algorithms and their performance can be enhanced applying different techniques. Decentralizing the population is one of them. The Mallba library [1] was used to implement four EAs. One centralized (one population) and other three decentralized (population split in subpopulations or islands).

The Mallba project is an integrated way to develop a skeleton library for combinatorial optimization that includes exact, heuristic and hybrid methods, which can be dealt with parallelism not only in a friendly way with the user but also in an efficient one. One of the main characteristics of Mallba is the easiness to change from a sequential optimization implementation to a parallel one.

The skeletons are based on the separation of two concepts: the problem to be solved and the general resolution method to be used. The skeletons can be seen as generic templates that only need to be instanciated with the characteristics of the problem in order to solve it. All the features related to the method of selected generic resolution and their interaction with the problem itself, are implemented by the skeleton, while the particular characteristics of the problem must be provided by the user.

In the following section the scheduling problem is briefly described. The implemented EAs are described in section 3, the experiment designs and the parameter settings are detailed in section 4. In section 5 we show and discuss the results. Finally, in section 6 the conclusions are presented.

## 2 UNRESTRICT PARALLEL MACHINES SCHEDULING PROBLEM

The scheduling problem that we are tackling can be described as follows: The *n* tasks are processed without interruption on some of the *m* identical parallel machines of the system ($P_m$) and each machine can process not more than one task at a time. The job *j* *(j=1,2,...n)* is made available for processing in zero time. It requires a positive and uninterrupted time for its processing $p_j$ and it has a due date $d_j$ in which the job could be ideally finished. For a given processing order of jobs, the earliest completion time $C_j$ and the maximum delay time $T_j = \{C_j - d_j, 0\}$ of the job *j* can be easily calculated. The problem is to find a processing order of the jobs with minimum objective values. The objectives to be minimized are:

$$MaximumTardiness : T_{max} = max_j(T_j)$$

$$AverageTardiness : T_{avg} = \frac{1}{n}\sum_{j=1}^{n} T_j$$

It is of theoretical and practical interest to get the solutions of these problems. Problems related to the due dates called considerable attention from researchers in the area. Most of the problems were unsolved for many years due to their computational complexity and were left so as an open subject until they were designated as NP-Hards [2].

## 3 EVOLUTIONARY ALGORITHMS

A Genetic Algorithm (GA)[5] is a technique of the Evolutionary Computation inspired by the principles of natural selection for to search solutions in a search space. This technique uses a population of individuals which represent the space of possible solutions. The search process consists of applying stochastic operators such as selection, crossover, and mutation over a randomly generated population in order to create a new generation of individuals. Then, the original population of parents is substituted by the new population of children. What is intended in this process is to keep the best individuals, eliminating the worse ones. The individuals have associated a fitness value which gives a degree of benefit, usefulness or goodness to be maximized or minimized while exploring the search space. The complete search process is iteratively repeated until some termination criteria is reached, usually until a certain number of iterations is reached. The GAs have been successfully applied to solve scheduling problems. When the GAs are applied to such problems, they see the *schedules* as individuals. The capacity of each individual is given by its *fitness* value. The fitness of an individual is calculated using the associated value of the objective function. In this work we have used two objective functions to be minimized related to the due dates. The same were described in the previous section.

### 3.1 Centralized Genetic Algorithms

In this case, there is only one population or panmixia [5]. The group of individuals of the population is the same and any individual can be mated with any other individual of the population.

### 3.2 Decentralized Genetic Algorithms

In the decentralized algorithms the populations are structured in sub populations or in grid structures that define a neighborhood for any individual in the population [5]. Decentralizing the GA by structuring the population could have many advantages and one of them is obtaining a greater algorithmic performance bettering the diversity convergence. Nevertheless, using structured populations can also have some disadvantage such as a greater complexity in the implementation and analysis. There are two important population models:

1. *Coarse Grained or Distributed Evolutionary Algorithm (dEAs)*
   The population is partitioned in different sub populations or islands. Each island works independently and there exist interchanges of individuals between the islands with a certain given frequency. This is the model that we implemented in this work for two, four and eight islands.
2. *Fine Grained or Cellular Evolutionary Algorithm (cEAs)*
   In cEAs the individuals are located in a d-dimensional bull-like grid (where d=1, 2 and 3 is used in the practice). An individual is placed by a location in the grid, this location is often referred to as a cell and thus, this is what we call fine grained or cellular evolving algorithm. Every individual has a neighborhood, and an individual can only be mated for reproduction with other individual of its neighborhood. The main difference of a cellular EA with respect to a panmictic (centralized) one is its decentralized selection, since the reproductive loop is performed inside each of the numerous individual pools. In a cEA, one given individual has its own pool defined by neighboring individuals, and at the same time, one individual belongs to many pools. This structure with overlapped neighborhoods is used to provide a smooth diffusion of good solutions along the grid.

Next we show the pseudo-codes for the implemented algorithms.

```
t = 0
initialize population P(t)
evaluate indviduals in P(t)
while not end do
   t = t + 1
   select C(t) from P(t-1)
   recombine structures in C(t) forming C'(t)
   mutate structures in C'(t) forming C''(t)
   evaluate structures in C''(t)
   replace P(t) from C''(t) and/or P(t-1)
```

Figure 1: Centralized Genetic Algorithm Pseudo-Code.

```
 1 t = 0
 2 initialize P(t)
 3 evaluate structures in P(t)
 4 while not end do
 5    t = t + 1
 6    select C(t) from P(t-1)
 7    recombine structures in C(t) forming C'(t)
 8    mutate structures in C'(t) forming C''(t)
 9    comunication with neighborhoods
10    evaluate structures in C''(t)
11    replace P(t) from C''(t) and/or P(t-1)
```

Figure 2: Decentralized Genetic Algorithms Pseudo-Code.

The difference between the codes is given by the step 9 (in Figure 2). In this step the inter-population operators are applied. Inter-population operators are the operators that are applied between sub-populations. Currently, the skeleton only has implemented a single inter-population operator, the migration operator. In order to configure this operator, the user must define:

- operator number (0): Because is the single inter-operator.
- operator rate: Is the migration rate or number of generation between migrations.
- number of individuals: Is the number of individuals to send.
- selection method (and its parameters) of individual to send
- and replace method (and its parameters) of individual to send

The islands are organized in a unidirectional ring topology (default implemented in the skeleton).

## 4 EXPERIMENTAL DESIGN

*Problem*: In this work we treat the identical parallel machine scheduling problem for 100 jobs and 5 machines.
*Benchmarks*: As is not usual to find published benchmarks for the scheduling problem we worked on, we take a test suite defined in a previous work [9]. In brief, data of twenty problem instances from of the OR library [8] were selected. In these instances the problem number is not consecutive because each one was selected randomly from different groups. In each group the tardiness factor which is an instance parameter that controls the amount of tardy jobs, is harder for those with a higher identifier number. That means that higher identifier number of instances involves greather amount of tardy jobs. Then the benchmark values were established by applying different conventional heuristics using the PARSIFAL software package provided by Morton and Pentico [3].
*Common Parameter Settings*: The representation of the solutions or schedules used in the implemented GAs was the permutation of integer numbers where each gene indicates the index of a task or job. The PMX was used as the crossover operator and the SWAP was used as the mutation operator. The parameter values used for all GAs were: 30 independent runs for each instance, with a crossover

probability of 0.65 and with a mutation probability of 0.05, the selection method used for parents and offspring was the ranking selection. All experiments were performed on Pentium 4 at 2.66 GHz processor using Susex Linux Operating System. To be able to compare the four algorithms and divide the experiments, we establish a maximum number of iterations which is of 150,000. Consequently, for each algorithm the setting was:

- *a*) popsize=128 offspring=128 iterations=1172 islands=1 (centralized)
- *b*) popsize=64 offspring=64 iterations=1172 islands=2 (decentralized)
- *c*) popsize=32 offspring=32 iterations=1172 islands=4 (decentralized)
- *d*) popsize=16 offspring=16 iterations=1172 islands=8 (decentralized)

*Particular Parameter Settings*: Aditionally for the decentralized case we use the synchronized mode, migration ratio = 10, number of migrating individuals = 1. The individual select to migrate was the one which had the best ranking (sender island) and the individual with the worse rank was replaced (receiver population). The three dEAs algorithms ran over 1 processor with 2, 4 and 8 process, respectively.
*Performance Metric*: To compare the performance of the four algorithms we used the ANOVA statistic test. We have taken a metric, *Best Found*, given by the Mallba library. This is the best value of the objective function found in each run. For each of the 20 instances we have 30 values of the objective function for the four algorithms. Then we apply ANOVA of one factor method for analyzing the variance, that is the best value found for each algorithm (sometimes called F test). The F value measures the difference between the means of the four algorithms. We used a value alpha of 0.05 in order to indicate a 95% confidence level in the results. To determine whether F is significant, we assume as null hypothesis that the means of all the algorithms are the same and we reject the null hypothesis if $F > F_c$ with $F_c$ the critic value.

## 5 RESULTS

In Tables 1 and 2 we used the following notation: $N$ column indicates the instance number and the $Bench$ column shows the benchmark value for this instance. A yes value in an entry of column *Significant F* indicates that the null hyphotesis is false and in column *Algorithm with Minimum Mean* we shown only the algorithm that obtained the minimum mean value (which is more close to the benchmark value), while a no value indicates that the null hypothesis is true.

In the Table 1 we can see that for almost all instances the F value is not meaningful. This indicates that the behavior of all algorithms regarding the quality of the solutions for the *maximum tardiness problem* is similar. Only for instance 61 the decentralized algorithm with 4 islands obtains the best performance.

For the *average tardiness problem* the results are different. In the Table 2 we can observe that for almost all instances the F value is meaningful. This indicates that the behavior of some of the algorithms is better, because its mean is the minimum when is compared with the mean of the each other algorithms. For this objective function there are, in almost all instances, more than one algorithm that obtains the better result, the letter in boldface indicates which is the one with a minor mean value showed in the corresponding column ($Mean$), but the values obtained by the other algorithms indicated in column *Algorithms with minimum means* are very similar. Also observing Table 2 in detail we can say with a 95% confidence that algorithms *a* and *b* have a similar performance. Nevertheless

| *$T_{max}$ para n=100 y m=5* | | | | | |
|---|---|---|---|---|---|
| *Inst. 100 jobs* | | *ANOVA Results with alpha=0.05* | | | |
| *N* | *Bench.* | *Significant F* | *Algorithm with Minimum Mean* | *Mean* | *Standard Deviation* |
| 1 | 590 | no | - | - | - |
| 6 | 1680 | no | - | - | - |
| 11 | 2620 | no | - | - | - |
| 19 | 3720 | no | - | - | - |
| 21 | 5240 | no | - | - | - |
| 26 | 168 | no | - | - | - |
| 31 | 1180 | no | - | - | - |
| 36 | 2120 | no | - | - | - |
| 41 | 3710 | no | - | - | - |
| 46 | 4580 | no | - | - | - |
| 56 | 670 | no | - | - | - |
| 61 | 1630 | yes | c | 2467.566 | 220.917 |
| 66 | 2440 | no | - | - | - |
| 71 | 3820 | no | - | - | - |
| 86 | 1240 | no | - | - | - |
| 91 | 2230 | no | - | - | - |
| 96 | 3250 | no | - | - | - |
| 111 | 1420 | no | - | - | - |
| 116 | 2320 | no | - | - | - |
| 121 | 3060 | no | - | - | - |

Table 1: *When F is significant, the model with minimum mean and their corresponding values of Mean and Standard Deviation are shown.*

the algorithm *a* was the best for the 80% of the instances, the algorithm *b* was the best for 65 %, algorithm *c* only work well for the 10% while the algorithm *d* has not a good performance for none.

From these results and considering the computational times show in Table 3, we can conclude that there is not advantage, for the particular problem under study, in to use decentralized algorithms because the quality of solutions found are very similar with a lower computational time for the centralized algorithm.

## 6 CONCLUSIONS

In this work we have presented four different Evolutive Algorithms: *a)* Single population or panmixia, *b)* population divided in two islands, *c)* population divided in four islands, and *d)* population divided in eight islands running over one processor. The experiments have been based upon studying 20 instances for 100 jobs and 5 parallel machine scheduling problems ($T_{max}, T_{avg}$). The obtained results were only analyzed regarding the quality of the solutions, using ANOVA test. This test, measures the differences in the population means of more than two groups. In this way we could tell that for the $T_{max}$ problem, all the algorithms had a similar performance. However, for the $T_{avg}$ problem, algorithms *a* and *b* were the best for around the 80% of the instances. The future research will focus: 1) to realize a set of new experiments but now running the decentralized algorithms over a cluster of computers and evaluating other important information given for other metrics, such as speedup and

| *$T_{avg}$ para n=100 y m=5* | | | | | |
|---|---|---|---|---|---|
| *Inst. 100 jobs* | | *ANOVA Results with alpha=0.05* | | | |
| *N* | *Bench.* | *Significant F* | *Algorithms with minimum Mean* | *Mean* | *Standard Deviation* |
| 1 | 37 | no | - | - | - |
| 6 | 215 | yes | b - **c** | 217.795 | 1.219 |
| 11 | 525 | yes | a - b - **c** | 529.931 | 1.893 |
| 19 | 1090 | yes | **a - b** | 1093.784 | 1.817 |
| 21 | 2070 | yes | a - **b** | 2077.933 | 0.976 |
| 26 | 3.7 | no | - | - | - |
| 31 | 145 | yes | **a - b** | 152.26 | 2.578 |
| 36 | 445 | yes | a - **b** | 460.013 | 3.837 |
| 41 | 1390 | yes | a | 1397.2626 | 1.945 |
| 46 | 1810 | yes | **a - b** | 1818.1103 | 1.048 |
| 56 | 73 | yes | a | 81.6166 | 2.983 |
| 61 | 493 | yes | b | 497.286 | 2.690 |
| 66 | 839 | yes | **a - b** | 844.848 | 3.793 |
| 71 | 1640 | yes | **a - b** | 1649.93133 | 1.241 |
| 86 | 370 | yes | a - **b** | 385.634 | 3.758 |
| 91 | 869 | yes | a | 881.992 | 1.909 |
| 96 | 1390 | yes | **a - b** | 1398.12933 | 1.772 |
| 111 | 592 | yes | **a - b** | 608.271 | 3.899 |
| 116 | 997 | yes | a | 1007.264 | 1.741 |
| 121 | 1270 | yes | **a - b** | 1282.234 | 2.531 |

Table 2: *When F is significant, the algorithms with minimum mean value and their corresponding values of Mean and Standard Deviation are shown.*

| *Time Mean Best Found* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|
| *Tavg* | 7.9408E+11 | 4.0843E+12 | 6.1493E+12 | 1.1472E+13 |
| *Tmax* | 1.0918E+12 | 2.3403E+12 | 3.9659E+12 | 1.0135E+13 |

Table 3: *Average Time Mean Best Found over all the Instances in $\mu$ sec.*

efficiency and 2) implementing the other model of decentralized algorithm: cellular GA.

## 7 ACKNOWLEDGMENTS

We acknowledge the Universidad Nacional de San Luis and the ANPCYT from which we receive continuous support.

## 8 BIBLIOGRAPHY

1. http://neo.lcc.uma.es/mallba/mallba.html.

2. M. Pinedo.,*“Scheduling: Theory, Algorithms and System”.* Prentice Hall, 1995.

3. T. Morton and D. Pentico.,*"Heuristic Scheduling Systems"*. John Wiley and Sons, New York, 1993.

4. Yamaha T., et al.,*"A Genetic Algorithm Applicable to Large Scale Job Shop Problems"*. Parallel problem Solving from Nature II, 1992.

5. http://neo.lcc.uma.es/cEA-web/introduction.htm

6. Esquivel S. C., Gatica C. R., Gallard R. H.,*"A genetic approach using direct representation of solutions for the parallel task scheduling problem"*., Proceeding of V Congreso Argentino de Ciencias de la Computación, CACIC'99, CD-Rom, UNCPBA, Octubre 1999.

7. Esquivel S. C., Gatica C. R., Gallard R. H.,*"A Multirecombinative Evolutionary Approach to solve the Parallel Task Scheduling Problem"*. Proceeding of VI Congreso Argentino de Ciencias de la Computación, CACIC'2000, pp.1343, Universidad Nacional de la Patagonia San Juan Bosco,Ushuaia, Octubre 2000.

8. J. Beasley. Or library. http://people.brunel.ac.uk/ mastjjb/info.html.

9. Ferretti E., Gallard R.,*"Soluciones a problemas de planificación de tareas en ambientes de máquinas paralelas por medio de técnicas de Computación Evolutiva"*. Trabajo Final de Licenciatura en Cs. de la Computación, 2004.

10. Ferretti E., Esquivel S., *"A Comparison of Simple and Multirecombinated Evolutionary Algorithms with and without Problem Specific Knowledge Insertion for Parallellel Machines Scheduling"*. International Transaction on Computer Science and Engineering, Vol.3, No.1, pp. 207-221, ISSN: 1738-6438, April 2005.

11. Ferretti E., Esquivel S., *"An Efficient Approach of Simple and Multirecombinated Genetic Algorithms for Parallel Machines Scheduling"*. IEEE Congress on Evolutionary Computation (CEC), Edimburgo - Escocia, 2-5 de Septiembre de 2005, pp. 1340-1347, ISBN:0-7803-9363-5.

12. Ferretti E., Esquivel S., *"Knowledge Insertion: An Efficient Approach to Simple Genetic Algorithms for Unrestricted Parallel Equal Machines Scheduling"*. in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), ISBN: 1-59593-010-8, Vol.2, pp.1587-1588, Washington, USA, July 25-29 de 2005.

13. http://www.georgetown.edu/departments/psychology/researchmethods/statistics/anova.htm.