An ACO approach for the Parallel Machines Scheduling Problem

Claudia R. Gatica, Susana C. Esquivel, and Guillermo Leguizamón
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis
Ejército de Los Andes 950, (5700) San Luis, Argentina
e-mail: {crgatica, esquivel, legui}@unsl.edu.ar

Abstract

The parallel machines scheduling problem (P_m) comprises the allocation of jobs on the system's resources, i.e., a group of machines in parallel. The basic model consists of *m* identical machines and *n* jobs. The jobs are assigned according to resource availability following some allocation rule. In this work, we apply the Ant Colony Optimization (ACO) metaheuristic which includes in the construction solution process different specific heuristic to solve P_m for the minimization *Maximum Tardiness* (T_{max}). We also present a comparison of previous results obtained by a simple genetic algorithm (GAs) and an evidence of an improved performance of the ACO metaheuristic on this particular scheduling problem.

Keywords: parallel machine scheduling, maximum tardiness, ant colony optimization algorithms, specific heuristic problem information.

1 INTRODUCTION

The parallel machines scheduling problems are representative of many real world problem. In such systems it is usual to drive the minimization of the objectives based on the due dates, such as the *Maximum Tardiness* (T_{max}). To provide reasonably good solutions in a very short time, the scheduling literature ([12] and [16]) offers a set of dispatching rules and heuristics. Depending on the particular instance of the problem, we see how some heuristics behave better than others. Evolutionary Algorithms have been successfully applied to solve scheduling problems (See [3], [4], and [5]). Similarly, the Ant Colony Optimization (ACO) metaheuristic has also been applied to solve scheduling problems such as in [1], [2], [8], [10], [13], [14] and [15]. However, there have been found too few works related, in particular, to Parallel Machines Scheduling Problem (P_m).

In this work we applied an Ant Colony System (ACS) [10], an advanced algorithm derived from the ACO metaheuristic, to P_m. It is important to note that the pheromone trail and heuristic information are the driving forces in ant algorithms to efficiently explore the search space. In the particular case of the heuristic information, different rules can be incorporated according to the problem under consideration. For the scheduling problem studied in this work, the ACS was implemented by considering different heuristic values, which are: Earliest Due Date (EDD), Shortest Processing Time (SPT), Longest Processing Time (LPT), and Least Slack (SLACK). In addition, we compared the results with a previous work by Ferretti et al. [3].

The remainder of the paper is organized as follows. Section 2, the Parallel Machines Scheduling Problem is formally introduced. In Section 3 we describe the Ant Colony Optimization metaheuristic and present the ACS algorithm for P_m. The experimental study, and the analysis and statistic results are presented respectivelly in Sections 4 and 5. Finally, in Section 6 the conclusions are presented.

2 PARALLEL MACHINES SCHEDULING PROBLEM

The formal notation used in the literature [12] for the scheduling problem that we are dealing is a triplet: ($P_m \mid\mid T_{max}$). The first field describes the machine environment, the second one contains the constrains, and the third one provides the objective function. This scheduling problem can be stated as follows: there are *n* jobs to be processed without interruption on some of the *m* identical machines belonging to the system (P_m); each machine can process not more than one job at a time. Job *j (j=1,2,...n)* is made available for the processing at time zero, it requires an uninterrupted positive processing time p_j on a machine and it has a due date d_j by which it should ideally be finished. For a given processing order of the jobs (schedule), 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 estimated. The problem consists in finding a optimum schedule objective value. The objective to be minimized is:

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

These problems related to the due dates have received considerable attention from a practical and theorical point of view, besides they have considered as NP-Hard when $2 \leq m \leq n$, see in the literature [12].

2.1 Conventional Heuristics to Scheduling Problems

Dispatching heuristics assign a priority index to every job in a waiting queue. The one with the highest priority is selected to be processed next. There are different heuristics (e.g., [12] and [16]) for the above mentioned problem whose principal property is not only the quality of the results but also to give a schedule to close the optimal sequencing. The following dispatching heuristics were selected to determine priorities, and they were used to build schedules by the Ant Colony System:

- *EDD* (Earliest Due Date first): the job with earliest due date is selected first and the final scheduled jobs are ordered satisfying:

$$d_1 \leq d_2 \leq ... \leq d_n.$$

- *SPT* (Shortest Processing Time first): the job with shortest processing time is selected first and the final scheduled jobs are ordered satisfying:

$$p_1 \leq p_2 \leq ... \leq p_n.$$

- *LPT* (Largest Processing Time first): the job with largest processing time is selected first and the final scheduled jobs are ordered satisfying:

$$p_n \leq p_{n-1} \leq ... \leq p_1.$$

- *SLACK* (Least slack): the job with the smallest difference between due date and processing time is selected first and the final scheduled jobs are ordered satisfying:

$$d_1 - p_1 \leq d_2 - p_2 \leq ... \leq d_n - p_n.$$

3 ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) is a metaheuristic in which a base process is the *solutions construction*, (see in the bibliography [11]). It manages a colony of ants that concurrently and asynchronously visit adjacent states of the considered problem by moving through neighbor nodes of the construction graph G. They move by applying a stochastic local decision policy, denoted by P_{ij}, which is descripted in section 3.1, it makes use of pheromone trails, denoted by (τ_{ij}) and heuristic information, denoted by (η_{ij}). In this way, ants increasingly build solutions to the optimization problem. Once an ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution that will be used by the *update pheromones* procedure to decide how much pheromone to deposit. *Update pheromones* this is the process by which the pheromone trails are modified. The trails value can either increase, as ants deposit pheromone on the components or connections they use, or decrease, due to pheromone evaporation. From a practical point of view, the deposit of new pheromone increases the probability for those components/connections, that were either used by many ants or that were used by at least one ant, and which produced a very good solution, to be used again by future ants. On the other hand, pheromone evaporation implements a useful form of *forgetting*: it avoids premature convergence of the algorithm toward a suboptimal region, therefore favoring the exploration of new areas of the search space.

The Ant Colony System (ACS), is an ACO algorithm introduced by Dorigo and Gambardella [10]. It uses a modified rule when an ant chooses the next travel node, it uses a best-so-far pheromone update rule but applies pheromone evaporation only to the trail that belongs to the solution components that are in best-so-far solution. It also uses a local pheromone update rule to decrease the pheromone values on visited solution components, in order-to encourage exploration. A general outline of the ACS is presented in Algorithm 1.

Algorithm 1 Pseudo-code for ACS

```
Initialize
for c=1 to Cycles-Number do
  for k=1 to Ants-Number do
    Construct-Ant-Solution (Local Update Pheromone)
    Save-Best-Solution
    Rank-Solution
    Global-Update-Pheromone
    Reallocation-Ants
  end for
end for
Print-Best-Solution
```

3.1 ACS for ($P_m||T_{max}$)

This section presents the description of the main components of the implemented ACS.

1. *Construction graph:* The ants perform random *walks* in a *construction graph* and these walks represent feasible solutions of the underlying combinatorial optimization problem. To construct a feasible solution the artificial ants successively choose jobs to be appended to the actual subsequence, until all jobs are scheduled. Each ant decides independently of each other which job j should be the i-th job in the sequence, and each ant generates a complete solution. A walk

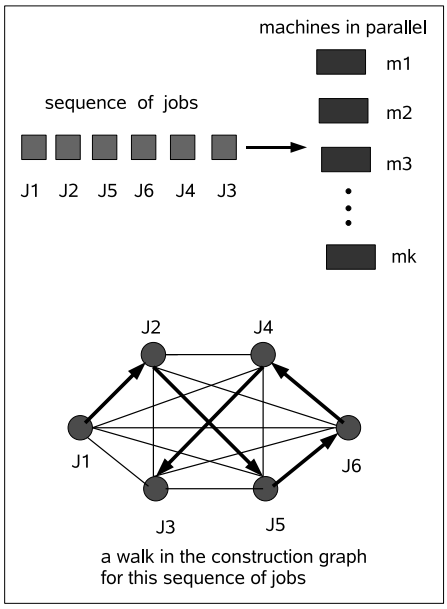

Figure 1: *Example of a construction graph for 6 jobs in m_k machines in parallel.*

consists of several "node-to-node" movements and these movements are performed on the basis of *transition probabilities*. The transition probability P_{ij} that job j be selected to be processed on position i in the sequence is formally given by:

$$P_{ij} = \begin{cases} \frac{\eta_{ij}\tau_{ij}^{\beta}}{\Sigma\eta_{ih}\tau_{ih}^{\beta}} & j\epsilon\Omega \\ 0 & otherwise \end{cases} \tag{1}$$

where Ω is the set of non-scheduled jobs, and h belongs to Ω, η_{ij} is the specific-problem heuristic information, and τ_{ij} the pheromone trails.

For example, for an instance problem of 6 jobs and m_k machines, the construction graph can be seen as in Figure 1. The nodes in the graph represent jobs whereas the edges represent the possible walks the ants can follow. The solution $(J_1, J_2, J_5, J_6, J_4, J_3)$ is represented by edges in boldface starting in node J_1.

2. The formulas of the local and global pheromone update are:

   (a) Local Update Rule

   $$\tau_{ij} = (1-\phi)\tau_{ij} + \phi\tau_0 \tag{2}$$

   where ϕ is a weight, and τ_0 is a constant value.

   (b) Global Update Rule

   $$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{b}(t+1) \tag{3}$$

   b is the index of the best-so-far solution.

3. For selecting the next component j, ACS uses the next formula:

$$j = \begin{cases} arg.max\{\tau_{ij}\eta_j^{\beta}\} & q_0 \leq q \\ P_{ij} & otherwise \end{cases} \quad (4)$$

P_{ij} is the probability item selection given in equation 1.

4. *The objective function:*

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

where T_j = $\{C_j$ - $d_j, 0\}$ is the maximum delay time, the d_j is the due date of j job, and C_j the earliest completion time.

5. *Heuristic information* η

   (a) *Earliest Due Date (EDD)* based heuristic, where jobs are sorted and scheduled according to ascending due dates.

$$\eta_{EDD} = 1/d_j$$

   (b) *Shortest Processing Time (SPT)* based heuristic, where jobs are sorted and scheduled according to ascending shortest processing time.

$$\eta_{SPT} = 1/p_j$$

   (c) *Largest Processing Time (LPT)* based heuristic, where jobs are sorted and scheduled according to ascending largest processing time.

$$\eta_{LPT} = p_j$$

   (d) *Least Slack (SLACK)* based heuristic, where jobs are sorted and scheduled according to ascending smallest difference between due date and processing time.

$$\eta_{SLACK} = 1/(d_j - p_j)$$

To implement the ACS with different heuristics we use the Mallba project [6]. It is an integrated way to develop a skeleton library for combinatorial optimization that includes exact, heuristic, and hybrid methods. 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 instanced 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.

4 EXPERIMENTAL DESIGN

As it is not usual to find published benchmarks for the scheduling problems we worked on, we built our own test suite with data, based on selected data corresponding to weighed tardiness problems taken from the OR-Library [9]. For problems of 40 jobs, 20 instances are selected and for problems of 100 jobs, 20 instances are selected as well, each instance with the same identification number, although they are not the same problem. That is to say that we have a problem numbered 1 with

40 jobs and another one numbered 1 with 100 jobs, and so on. The numbers of the problems are not consecutive because each one was selected randomly from different groups. The tardiness factor is harder for those with the highest identification number. In the OR-Library 125 test instances are available for each problem size $n = 40$, $n = 50$ and $n = 100$.

The instances were randomly generated as follows: For each job j ($j = 1, ..., n$), an integer processing time $p\{j\}$ was generated from the uniform distribution [1,100] and integer processing weight $w\{j\}$ was generated from the uniform distribution $[1, 10]$. Instance classes of varying hardness were generated by using different uniform distributions for generating the due dates. For a given relative range of due dates RDD ($RDD = 0.2, 0.4, 0.6, 0.8, 1.0$) and a given average tardiness factor TF ($TF = 0.2, 0.4, 0.6, 0.8, 1.0$), an integer due date $d(j)$ for each job j was randomly generated from the uniform distribution $[P(1 - TF - RDD/2), P(1 - TF + RDD/2)]$, where $P = SUM\{j = 1, ..., n\}p(j)$. Five instances were generated for each of the 25 pairs of values of RDD and TF, yielding 125 instances for each value of n. These data were the input for dispatching rules and conventional heuristics, and for the implemented ACS algorithm.

To evaluate the dispatching rules and the conventional heuristics we used PARSIFAL [16], a software package provided by Morton and Pentico to solve different scheduling problems by means of different heuristics, as for example *EDD* and *SPT* used in this work.

To compare the ACS algorithm with four heuristics, the following relevant performance variables were chosen from previous works [3]:

- $Ebest = ((\textit{best value} - \textit{opt-val})/\textit{opt-val}) * 100$: it is the percentage error of the best found solution when compared with the known or estimated (upper bound) optimum value *opt-val*. It gives a measure on how far the best solution is from that *opt-val*. When this value is negative, it means that the *opt-val* has been improved.

- *Mean Ebest (MEbest)*: it is the mean value of *Ebest* throughout all runs.

- *Mean Best ($\mu Best$)*: it is the mean objective value obtained from the best found solutions throughout all runs.

- *Hit Ratio*: it is the percentage of runs where the ACS reaches or improves the known or estimated optimum value.

The initial phase of the experiments consisted in establishing the best results from dispatching rules and conventional heuristics to use them as upper bounds for the objective function values. Also, the best parameter values for the ACS were obtained after performing a set of previous experiments and some from the related literature [7], and they are presented in Table 1.

In all the experiments, we used the same maximum number of evaluations, in order to compare the different variants of the implemented ACS, and also to compare them with the results obtained by the simple genetic algorithm reported in a previous work [3]. We took a maximum number of 140,000 evaluations, and used *elitism*. We performed several experiments with T_{max}, for three systems of parallel machines scheduling problems:

- 20 instances of 40 jobs and $p = 2$ processors.

- 20 instances of 40 jobs and $p = 5$ processors.

- 20 instances of 100 jobs and $p = 5$ processors.

Parameters	
Names	*Values*
numbers of runs	10 and 30
number of steps	1000
colony size	140
α	1
β	5 and 10
ρ	0.5 and 0.9
q_0	0.9
ϕ	0.02 and 0.05
t_0	0.5
ξ	0.02 and 0.05

Table 1: *The Parameters values*

5 ANALYSIS OF RESULTS

In this section we present the results obtained by the implemented ACS algorithm with four different heuristics. Table 2 shows the *MEbest* values obtained of T_{max} objective, for the first system problem, table 3 for the second problem, and table 4 for the third problem. Here we shaded cells with the intention of showing the following: darker cells represent best values. That is, darker cells reach higher performance of the ACS with the respective heuristic value. In turn, to indicate statistical significant differences we used the different shades. Similarly, we used the same shade to indicate any statistical significant differences. In order to accomplish this, we took one instance at a time and we applied an ANOVA the *MEbest* values of four heuristics with a confidence level of 95 percent, doing the same for all the instances of the three problems.

Seeing the different shades in tables 2, 3, and 4, we can notice that the *EDD* heuristic is the darkest in all the problems and instances, followed by *SLACK* heuristic in the second place, and *SPT* in the third place. The *LPT* heuristic is the lightest in all instances of the three problems, meaning that it was the worst performance. We can also see that the heuristics *SLACK* is not the best when the problem instances have a higher numeration. From all this, we may say that the heuristic information related to due date as *EDD* and *SLACK* incorporate more information to search process of the ACS algorithms and this is lower average percentage error in the data *MEbest*.

Besides, we present tables 5, 6, and 7, in which it can be observed the columns *Opt-val* (best known value, obtained by Parsifal), $\mu Best$, and *HRatio* are obtained by simple genetic algorithms *GAs* reported in [3], and by ACS algorithms using four heuristics for each problem studied.

In the tables 5 and 6, we can see the $\mu Best$, values obtained by *EDD* heuristic were the best minimun values, and the *Hit Ratio* values were the higher. However, for the instance number 66, 91, 111, 116, and 121, the *Hit Ratio* is zero, except en table 5 for instance number 116. That means the optimum value never has been reached. For the other hand, the *Hit Ratio* values of *GAs* are zero for 111 and 116 instances in the three problems.

In table 7 the minimun $\mu Best$ values were obtained by *GAs*, even the instances with higher identification, but the *Hit Ratio* were better for *EDD* and *SLACK* heuristics.

We could say that the performance of ACS algorithms were the best using the heuristics *EDD* and *SLACK* is that because the due date values used for heuristic also are related to the objective function. It is important to note that this relationship between the heuristics and objective function can improve the process of searching for ACS algorithms. However, the best $\mu Best$ values for instances of higher

Maximun Tardiness: MEbest				
Inst.	*n=40 y p=2*			
N	*EDD*	*SPT*	*LPT*	*SLACK*
1	-8,085	2,128	111,915	-8,08
6	-0,835	1,002	40,902	-0,835
11	-5,849	-4,151	20,849	-5,849
19	-0,246	0,369	9,521	-0,246
21	-1,506	-1,506	8,494	-1,084
26	-50,909	201,818	721,818	-50,909
31	-4,049	27,530	122,267	-4,049
36	-1,956	8,861	44,879	-1,956
41	-0,703	9,766	39,609	-0,703
46	-2,339	-2,984	17,581	8,226
56	-7,287	53,036	261,538	-7,287
61	0,332	37,874	176,246	0,332
66	0,459	23,395	74,954	0,459
71	-0,859	1,094	38,281	0,391
86	-7,302	45,233	222,718	-7,302
91	6,696	29,911	111,161	16,183
96	-0,130	10,735	29,343	1,301
111	9,408	85,584	226,555	13,050
116	-2,923	47,385	184,000	134,154
121	5,175	14,336	51,049	23,636
AVG	**-3,645**	**29,571**	**125,684**	**5,471**

Table 2: *The MEbest values for four different Heuristics*

identification of the three problems were the obtained by *GAs* algorithms.

6 CONCLUSIONS

In this work we have presented an ACS algorithm with four different heuristics implemented: 1) Earliest Due Date (EDD), 2) Shortest Processing Time (SPT), 3) Largest Processing Time (LPT), and 4) Least Slack (Slack). The experiments have been based upon studying 20 instances, for each three of the problems, where the parallel identical machine scheduling problems ($P_m \mid\mid T_{max}$) were intented to be minimized. The obtained results were analized statisticaly by ANOVA test which was used to measure the differences beetwen the means of four data groups belonging to each problem.

We see that a best performance was achieved when the *ACS* algorithm is used with information heuristic which more has related to the problem, such as *EDD* and *SLACK*. However, this results could be compared with simple *GAs* as well, but we can not say the same for advanced *GAs*. Clearly, more experiments are necessary because none of the algorithms found the optimum values for all problem instances, such as instances 66, 91, 111, 116, and 121. In a future work, we plan to incorporate local search to the *ACS* using hybridization techniques, and to use other set of instance problems.

References

[1] Bauer A., Bernd B., Hartl R., and Strauss C. An ant colony optimization approach for the single machine total tardiness problem. *IEEE*, pages 1445–1450, 1999.

[2] Bauer A., Bernd B., Hartl R., and Strauss C. Minimizing total tardiness on a single machine using ant colony optimization. *Central European Journal of Operations Research*, 8(2):125–141, 2000.

Maximun Tardiness: MEbest				
Inst.	*n=40 y p=5*			
N	*EDD*	*SPT*	*LPT*	*SLACK*
1	-15,845	14,789	67,254	-19,014
6	-5,828	3,067	28,988	-7,055
11	-9,381	-3,363	13,274	-10,000
19	-3,294	1,412	5,118	-3,529
21	-1,395	0,349	4,360	-0,291
26	-30,000	156,00	334,00	-35,000
31	-13,665	16,460	73,603	-14,441
36	-8,130	4,370	29,980	-8,130
41	-1,866	7,015	32,015	-1,269
46	0,763	-0,382	11,069	1,069
56	-19,182	78,616	189,623	-20,755
61	-4,206	35,007	121,981	-9,227
66	1,532	15,726	50,726	6,855
71	-0,752	6,466	32,331	3,910
86	-10,526	40,747	182,513	-7,131
91	4,038	31,539	76,442	7,019
96	-1,302	6,627	18,935	-1,065
111	22,175	79,113	204,006	34,478
116	25,149	59,375	179,762	122,619
121	6,646	14,557	36,709	2,848
AVG	**-3,253**	**28,374**	**84,634**	**2,095**

Table 3: *The MEbest values for four different Heuristics*

[3] Ferretti E. and Esquivel S. A comparison of simple and multirecombinated evolutionary algorithms with and without problem specific knowledge insertion, for parrallel machines scheduling. *International Transaction on Computer Science and Engineering*, 3(1):207–221, 2005.

[4] Ferretti E. and Esquivel S. An efficient approach of simple and multirecombinated genetic algorithms for parallel machine scheduling. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 1340–1347, Scotland, UK, September 2005. IEEE Center.

[5] Ferretti E. and Esquivel S. Knowledge insertion: An efficient approach to simple genetic algorithms for unrestricted for parallel equal machines scheduling. In *GECCO'05*, pages 1587–1588, Washington DC, USA, 2005.

[6] Alba Enrique, Luque Gabriel, Garcia Nieto Jose, Ordónez Guillermo, and Leguizamón Guillermo. MALLBA: A software library to design efficient optimisation algorithms. *International Journal of Innovative Computing and Applications*, 1(1):74 – 85, 2007.

[7] Botee M. Hozefa and Eric Bonabeau. Evolving ant colony optimization. *Complex Systems*, pages 149–159, 1998.

[8] Heinonen J. and Pettersson F. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. *Applied Mathematics and Computation*, 2006.

[9] OR library Beasley J. http://people.brunel.ac.uk/mastjjb/info.html.

[10] Dorigo M. and Gambardella L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transaction Evolutionary Comput.*, 1(1):53–66, 1997.

[11] Dorigo M. and Stülzle T. *Ant Colony Optimization*. Massachusetts Institute of Tecnology, 2004.

[12] Pinedo M. *Scheduling: Theory, Algorithms and System*. Prentice Hall, 1995.

Maximun Tardiness: MEbest				
Inst.	*n=100 y p=5*			
N	*EDD*	*SPT*	*LPT*	*SLACK*
1	-7,288	27,797	134,407	-8,475
6	-6,190	1,310	29,583	-6,012
11	-2,328	4,504	23,473	-2,786
19	-0,376	5,430	24,409	-0,511
21	0,420	1,660	8,015	3,531
26	-38,095	254,167	934,524	-47,024
31	-1,780	45,170	141,356	-4,237
36	-2,217	13,066	81,274	-2,736
41	-2,264	12,911	46,119	-2,453
46	1,288	6,572	17,795	12,227
56	-5,522	158,657	338,657	-9,403
61	-0,920	89,939	146,564	-0,982
66	3,648	38,730	80,738	22,377
71	4,319	24,817	36,649	9,293
86	0,887	150,645	263,952	2,016
91	25,650	53,049	117,040	28,655
96	11,692	31,077	51,692	26,308
111	63,099	173,028	240,282	60,916
116	22,716	57,974	113,147	54,741
121	9,183	41,830	67,157	56,699
AVG	**3,796**	**59,617**	**144,842**	**9,607**

Table 4: *The MEbest values for four different Heuristics*

[13] Bank Markus and Honing Udo. An aco-based approach for scheduling task graphs with communication costs. *International Conference on Parallel Processing*, pages 623–629, 2005.

[14] Chandrasekharan Rajendran and Hans Ziegler. Ant colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, pages 426–438, 2004.

[15] Saravana Sankar, Ponnambalam S.G., Rathinavel V., and Visveshvaren M.S. Scheduling in parallel machine schop: An ant colony optimization approach. *Industrial Technology, 2005, ICIT 2005 IEEE International Conference*, pages 276–280, 2005.

[16] Morton T. and Pentico D. *Heuristic Scheduling Systems*. John Wiley and Sons, New York, 1993.

Maximun Tardiness: n=40 y p=2											
Inst.	*Opt.*	*GAs*		*EDD*		*SPT*		*LPT*		*SLACK*	
N	*val.*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*
1	235	225.73	9	**216.0**	100	259.4	0.0	509.2	0.0	**216.0**	100
6	599	614.7	17	**594.0**	100	615.1	0.0	860.2	0.0	**594.0**	100
11	1060	1018.1	93	**998.0**	100	1025.6	100	1284.4	0.0	**998.0**	100
19	1628	1635.93	20	**1624.0**	100	1644.8	0.0	1785.0	0.0	**1624.0**	100
21	1660	**1639.67**	100	1646.2	90	1640.5	100	1801.9	0.0	1688.5	20
26	55	66	60	**27.0**	100	209.9	0.0	465.8	0.0	**27.0**	100
31	494	527.47	7	**474.0**	100	671.8	0.0	1126.4	0.0	**474.0**	100
36	869	920.23	7	**852.0**	100	979.0	0.0	1292.8	0.0	**852.0**	100
41	1280	1421.8	7	**1271.0**	100	1431.7	0.0	1814.0	0.0	1271.1	100
46	1240	**1224.0**	80	1230.1	80	1233.2	50	1458.8	0.0	1381.8	0.0
56	247	293.17	0.0	**229.0**	100	483.0	0.0	906.4	0.0	**229.0**	100
61	602	762.73	0.0	604.1	0.0	915.8	0.0	1675.2	0.0	**604.0**	0.0
66	1090	1244.43	0.0	**1236.3**	0.0	1386.0	0.0	1923.1	0.0	1272.2	0.0
71	1280	**1289.27**	53	1346.0	10	1350.1	0.0	1770.0	0.0	1398.8	0.0
86	493	535.43	17	**457.0**	100	810.4	0.0	1694.0	0.0	**457.0**	100
91	896	**1014.07**	3	1164.4	0.0	1272.8	0.0	1926.0	0.0	1192.1	0.0
96	1537	**1592.03**	13	1630.6	30	1747.6	0.0	2016.8	0.0	1615.6	0.0
111	659	**956.97**	0.0	1014.6	0.0	1311.1	0.0	2156.4	0.0	1206.7	0.0
116	650	**799.3**	0.0	900.3	20	1084.1	0.0	1866.4	0.0	1635.2	0.0
121	1430	**1486.07**	13	1676.6	0.0	1733.3	0.0	2160.4	0.0	1954.0	0.0
AVG	**900,2**	**963.36**	**29.0**	**959.56**	**61.5**	**1090.26**	**12.5**	**1524.66**	**0.0**	**1034.55**	**51.0**

Table 5: *The Opt-val are obtained by Parsifal, the $\mu Best$ and $HitRatio$ are obtained by simple genetic algorithms GAs and ACS algorithms using four different heuristics.*

Maximun Tardiness: n=40 y p=5											
Inst.	*Opt.*	*GAs*		*EDD*		*SPT*		*LPT*		*SLACK*	
N	*val.*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*
1	284	283.67	50	242.8	100	352.9	0.0	504.5	0.0	**232.5**	100
6	652	647.2	67	621.5	100	702.4	0.0	858.6	0.0	**608.0**	100
11	1130	1059.83	100	1030.9	100	1113.1	80	1288.0	0.0	**1019.9**	100
19	1700	1666.4	97	1656.0	100	1741.9	0.0	1796.8	0.0	**1648.0**	100
21	1720	**1681.37**	90	1734.6	10	1738.5	0.0	1795.0	0.0	1730.5	30
26	100	127.77	13	79.5	100	307.5	0.0	472.4	0.0	**71.1**	100
31	644	613.2	77	**560.2**	100	800.4	0.0	1132.0	0.0	**560.2**	100
36	984	1001.8	37	919.3	100	1090.9	0.0	1295.7	0.0	**908.1**	100
41	1340	1446.07	3	**1316.9**	100	1498.4	0.0	1823.0	0.0	1326.0	100
46	1310	**1270.43**	100	1346.6	0.0	1333.7	20	1459.0	0.0	1340.8	0.0
56	318	403.97	0.0	263.7	100	610.7	0.0	935.1	0.0	**255.1**	100
61	737	896.37	0.0	717.9	100	1040.2	0.0	1653.3	0.0	**684.7**	100
66	1240	**1363.9**	0.0	1385.7	0.0	1500.0	0.0	1953.5	0.0	1429.1	0.0
71	1330	**1352.97**	10	1416.5	10	1462.6	0.0	1764.5	0.0	1464.9	0.0
86	589	624.0	23	**541.6**	100	917.6	0.0	1711.0	0.0	564.4	100
91	1040	**1112.8**	17	1239.3	0.0	1423.7	0.0	1902.5	0.0	1308.2	0.0
96	1690	**1699.27**	50	1731.8	20	1844.5	0.0	2040.5	0.0	1766.1	10
111	699	**1036.73**	0.0	1131.1	0.0	1422.3	0.0	2135.0	0.0	1239.8	0.0
116	672	**943.57**	0.0	1050.7	0.0	1208.4	0.0	1887.0	0.0	1588.0	0.0
121	1580	**1639.67**	20	1778.9	0.0	1847.2	0.0	2162.0	0.0	1893.0	0.0
AVG	**987.95**	**1043.55**	**38.0**	**1038.275**	**57.0**	**1197.845**	**5.0**	**1528.47**	**0.0**	**1081.92**	**57.0**

Table 6: *The Opt-val are obtained by Parsifal. the $\mu Best$ and $HitRatio$ are obtained by simple genetic algorithms GAs and ACS algorithms using four different heuristics.*

Maximun Tardiness: n=100 y p=5											
Inst.	*Opt.*	*GAs*		*EDD*		*SPT*		*LPT*		*SLACK*	
N	*val.*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*	$\mu Best$	*HRatio*
1	590	742.67	3	555.3	100	833.6	0.0	1436.27	0.0	**547.53**	100
6	1680	1676.1	57	1588.17	100	1773.3	0.0	2271.87	0.0	**1583.40**	100
11	2620	2686.03	7	2569.6	100	2774.4	0.0	3249.73	0.0	**2556.13**	100
19	3720	3889.10	0.0	3720.3	40	3995.3	0.0	4635.03	0.0	**3704.87**	100
21	5240	5330.9	0.0	**5311.9**	0.0	5404.57	0.0	5676.17	0.0	5495.93	0.0
26	168	499.87	0.0	129.2	100	781.43	0.0	1805.97	0.0	**110.30**	100
31	1180	1515.30	0.0	1167.67	100	1829.93	0.0	2900.0	0.0	**1139.57**	100
36	2120	2462.23	0.0	2092.33	100	2598.17	0.0	3908.77	0.0	**2072.13**	100
41	3710	4087.43	0.0	3646.27	100	4349.17	0.0	5456.0	0.0	**3634.37**	100
46	4580	**4694.37**	7	4762.17	0.0	4963.07	0.0	5401.5	0.0	5188.67	0.0
56	670	1465.93	0.0	658.33	80	1928.67	0.0	2957.4	0.0	**613.0**	100
61	1630	2620.43	0.0	1646.27	13.33	3263.7	0.0	4065.4	0.0	**1631.03**	46.67
66	2440	3276.7	0.0	**3102.97**	0.0	3610.1	0.0	4435.87	0.0	3474.13	0.0
71	3820	**4436.1**	0.0	4499.4	0.0	4841.23	0.0	5238.33	0.0	4612.17	0.0
86	1240	2695.53	0.0	**1271.43**	0.0	3299.53	0.0	4598.13	0.0	1457.57	0.0
91	2230	**3096.47**	0.0	3285.33	0.0	3633.07	0.0	4853.83	0.0	3643.63	0.0
96	3250	**3933.7**	0.0	4040.2	0.0	4354.0	0.0	4938.17	0.0	4561.57	0.0
111	1420	3214.43	0.0	**3014.23**	0.0	4052.6	0.0	4864.57	0.0	3075.23	0.0
116	2320	**3390.9**	0.0	3632.87	0.0	3903.97	0.0	4959.33	0.0	4441.80	0.0
121	3060	4192.17	0.0	**4077.67**	0.0	4460.87	0.0	5122.0	0.0	4867.33	0.0
AVG	**2384.4**	**2595.28**	**3.0**	**2738.56**	**41.67**	**3332.53**	**0.0**	**4138.72**	**0.0**	**2920.52**	**47.33**

Table 7: *The Opt-val are obtained by Parsifal, the $\mu Best$ and $HitRatio$ are obtained by simple genetic algorithms GAs and ACS algorithms using four different heuristics.*