SOLUTIONS TO THE DYNAMIC AVERAGE TARDINESS PROBLEM IN SINGLE MACHINE ENVIRONMENTS

De San Pedro M., Lasso M., Villagra A., Pandolfi D.
Proyecto UNPA-29/B032[1]
División Tecnología
Unidad Académica Caleta Olivia
Universidad Nacional de La Patagonia Austral
Ruta 3 Acceso Norte s/n
(9011) Caleta Olivia – Santa Cruz - Argentina
e-mail: {mlasso,dpandolfi,edesanpedro,avillagra}@uaco.unpa.edu.ar
Phone/Fax : +54 0297 4854888

Gallard R.
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)[2]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
(5700) - San Luis -Argentina
e-mail: rgallard@unsl.edu.ar
Phone: +54 2652 420823
Fax : +54 2652 430224

Abstract

In dynamic scheduling arrival times as well, as some or all job attributes are unknown in advance. Dynamism can be classified as partial or total. In simplest partially dynamic problems the only unknown attribute of a job is its arrival time r_j. A job arrival can be given at any instant in the time interval between zero and a limit established by its processing time, in order to ensure finishing it before the due date deadline. In the cases where the arrivals are near to zero the problem becomes closer to the static problem, otherwise the problem becomes more restrictive. In totally dynamics problems, other job attributes such as processing time p_j, due date d_j, and tardiness penalty w_j, are also unknown.

This paper proposes different approaches for resolution of (partial and total) Dynamic Average Tardiness problems in a single machine environment. The first approach uses, as a list of dispatching priorities a final (total) schedule, found as the best by another method for a similar static problem: same job features, processing time, and due dates. The second approach uses as a dispatching priority the order imposed by a partial schedule created by another heuristic, at each decision point. The details of implementation of the proposed algorithms and results for a group of selected instances are discussed in this work.

Keywords: Evolutionary Scheduling, Average Tardiness, Dynamic scheduling, conventional heuristics.

[1] The Research Group is supported by the Universidad Nacional de La Patagonia Austral.
[2] The LIDIC is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

1. Introduction

Manufacturing organizations are frequently subject to several sort of changes, such as new job releases, machine breakdowns, job cancellation and due date or time processing changes. Due to their dynamic nature, real scheduling problems are computationally complex and the time required to compute an optimal solution increases exponentially with the size of the problem [15]. Particularly in *Single Machine Scheduling Problems* (SMSP) a set of jobs should be planned on a single machine, each job is processed one at a time, conflicting objectives should be accomplished and frequently, a number of restrictions should be satisfied. The study of these problems is very important because good solutions provide a support to manage and model the behaviour of more complex systems. In these systems it is important to understand the working of their components, and quite often the single-machine problem appears as an elementary component in a larger scheduling problem [3]. SMSP can be classified as *static* or *dynamic*. In *static* problems, all job attributes are known before scheduling starts, while *dynamic* problems, ranges from *partially dynamic* where job release times are unknown to *totally dynamic* where all job properties are unknown.

Evolutionary algorithms have been successfully applied to solve scheduling problems [9, 13, 16]. Current trends in evolutionary algorithms make use of multiparent [4, 5] and multirecombined approaches [6, 7, 8]. The latter, known as MCMP (Multiple-Crossovers-on-Multiple-Parents), allows a better balance between exploration and exploitation of the search space. A new variant of this approach applied to static scheduling problems [10,11, 12] is known as MCMP-SRI. Here, an individual selected from the old population and designated as the *stud* (S), provides to the multirecombination process good features of the evolved population while a set of random immigrants (RI) provides genetic diversity to avoid premature convergence. Dispatching rules are techniques that provide a reasonably good solution in relatively short time. An elementary rule is a function of attributes of jobs and machines. An attribute can be any property associated with a job or a machine and can also be constant or variable, depending on time [15]. A composition of a dispatching rule is a ranking expression that combines a number of elementary dispatching rules.

In this paper we propose two approaches to lead with partial and total dynamism. For partial dynamism, to decide which job to process *Dyna-S* make use of schedules previously found for the static case as a clue to build good schedules for the dynamic problem, where changes are related only to the unpredictable job arrival times. For total dynamism *Dyna-H* establishes the priority of jobs in the waiting queue, each time the resource becomes available, resorting to different heuristics.

2. Dynamic scheduling for Single Machine Problems

We consider an environment where n jobs should be planned without interruption on a single machine that can handle no more than one job at a time. For each job j ($j = 1,...,n$) it is associated an arrival time r_j, a processing time p_j and a due date d_j. Our problem is to find a processing order of the jobs with minimum average tardiness, defined as follows.

$$\frac{1}{n}\sum_{j=1}^{n} T_j$$

The problem has received considerable attention by different researchers. For many years its computational complexity remained open until established as NP-Hard in 1989 [16]. In the *static* case all jobs and their properties are simultaneously available for processing in time zero, which represents in most cases a not very common situation. In scheduling problems involved with real production, the environments are *dynamic*, at least in the sense that jobs arrivals can occur at unpredictable times. However, once the jobs arrive to the system generating a waiting queue, this can be considered as a static case for the determination of the next task to be allocated to the machine. Due to this characteristic, the study of the static case is important since the method that provide good solutions can be a suitable surrogate for the cases of dynamism. We can consider the static weighted tardiness problem, where all arrival times are equal to zero, that is $r_j = 0$ for all j, as a relaxation of the dynamic problem.

3. Typical approaches to the static average tardiness problem

Dispatching heuristics are methods that allow deciding which job should be processed next. To do this, a rule assigns a priority index to every job and the one with the highest priority is selected. There are different heuristics [9] for the Static Average Tardiness problem whose principal property is not only the quality of the results, but also to provide a job ordering (schedule) close to the optimal sequence. The following dispatching rules and heuristics were selected to determine priorities, build schedules and contrast their outcomes with those obtained by the evolutionary algorithms [14].

SPT (Shortest Processing Time first) the job with the shortest processing time is selected first and in the final schedule jobs are ordered satisfying: $p_1 \leq p_2 \leq \ \ldots \leq p_n$.

EDD (Earliest Due Date first) the job with earliest due date is selected first and in the final schedule jobs are ordered satisfying: $d_1 \leq d_2 \leq \ \ldots \leq d_n$.

Hodgson Algorithm: This heuristic provides a schedule according to the following procedure,

Step 1: Order the activities in EDD order.
Step 2: If there are no tardy jobs, stop; this is the optimal solution.
Step 3: Find the first tardy job, say k, in the sequence.
Step 4: Move the single job j ($1 \leq j \leq k$) with the longest processing time to the end of the sequence.
Step 5: Revise the completion times and return to step 2.

This algorithm is optimal for a related objective (unweighted number of tardy jobs) and can behave well for some instances of average tardiness.

Rachamadagu and Morton Heuristic (R&M). This heuristic provides a schedule according to the following expression.

$$\pi_j = (w_j / p_j)[\exp\{-(S_j)^+ / kp_{av}\}]$$

with $Sj = [dj - (pj + Ch)]$ is the slack of job j at time Ch, where Ch is the total processing time of the jobs already scheduled, k is a parameter of the method (usually k =2.0) and p_{av} is the average processing time of jobs competing for top priority. In the *R&M* heuristic, also called the *Apparent*

Tardiness Cost heuristic, jobs are scheduled one at a time and every time a machine becomes free a ranking index is computed for each remaining job. The job with the highest-ranking index is then, selected to be processed next.

4. Algorithms for Average Tardiness Dynamic Scheduling

For the *partially dynamic case* we propose *Dyna-S* which is based on the knowledge provided by, an evolutionary algorithm or by different dispatching heuristics used previously to solve static cases. The *Dyna-S* algorithm uses as a dispatching rule the job order provided by a total schedule *S* generated by an evolutionary algorithm (*MCMP-SRI*), or by conventional heuristics (*EDD, SPT, Hodgson* and *R&M*, as shown before). To schedule a job, an arrival queue is created with those jobs whose r_j are earlier or equal than the time *t* when the machine is available for processing. From that waiting queue, the job that appears first in the ordering of the total schedule *S,* is selected to be allocated next. Once a job is planned, it is removed from the queue. This process is repeated each time when the resource becomes available and while there are jobs in the waiting queue.

In our experiments we contrasted two versions of the *Dyna-S* algorithm:

- *Dyna-S-Heuristic* uses as a dispatching rule the job order provided, in the total static schedule, by the best performance heuristic.
- *Dyna-S-EA* uses as a dispatching rule the job order provided, in the total static schedule, by *MCMP-SRI.*

For the totally dynamic case we propose *Dyna-H,* which applies different heuristics to determine which job to schedule next. Essentially in *Dyna-H*, jobs in the waiting queue are planned according to some dispatching rule, which generates a partial schedule. Again, a waiting queue is generated with those jobs whose r_j are smaller or equal than the time *t* when the machine is available for processing. Now we produce a partial schedule (reordering the available jobs). The algorithm uses this partial schedule as a list of dispatching priorities to schedule the next job by choosing the job in the first position of this list. Once the job is planned, it is removed from the queue. This process is repeated each time when the resource becomes available and while there are jobs in the waiting queue.

Dyna-H was conceived in two different versions:

- *Dyna-H-Heuristic,* using one of the above listed conventional heuristics.
- *Dyna-H-EA* using a MCMP-SRI hibridized approach which is combined with EDD (earliest due date) in the case of jobs with zero tardiness or with R&M in the case of tardy jobs.

The last approach *Dyna-H-EA* deserves further explanation. To determine a partial schedule, which minimizes average tardiness for the jobs in the waiting queue *Dyna-H-EA*, depending on the queue length, uses an EA (if the queue length is greater than 5) or an enumerative algorithm (if the queue length is less than or equal to 5). Furthermore problem-specific-knowledge provided by EDD or R&M, is inserted in the algorithm. For example, in the case that a partial schedule containing only non-tardy jobs exists, then EDD can provide this schedule. Otherwise, if at the head of the schedule tardy and non-tardy jobs exists, then R&M determines which of the non-tardy jobs should be scheduled first.

More in detail *Dyna-H-EA*, works as follows. Each time a new job arrives, it is inserted in a queue waiting for the resource availability. When the machine becomes available, the algorithm checks, by means of EDD, if a partial schedule with average tardiness zero exists. In that case the job at the head of this partial schedule is assigned to the machine. Otherwise, depending on the waiting-queue length an enumerative or an evolutionary algorithm is run to produce a partial schedule. Then the job at the head of this schedule is scheduled first. In this last case, sometimes the partial schedule produced has the following characteristics: an average tardiness greater than zero and many jobs at the head with tardiness equal to zero. As a change in the relative order of these jobs does not change the total average tardiness of this *partial* schedule, but can alter the objective value of the *final* (total) schedule, the question is which of the prospective candidates should be processed first. As R&M works with a non-lineal slack factor, which measures the time needed for a job to be tardy, it makes a further discrimination between jobs which helps in building a good final schedule. This is the heuristic that *Dyna-H-EA* uses for this special but not infrequent case.

5. Experimental Tests and Results

As it is not usual to find published benchmarks for the Average Tardiness problem we built our own test suite with data (p_j ,d_j) extracted from 20 selected instances of the OR-library benchmarks for the weighted tardiness problem, with 40-jobs problem size, [1,2]. This data was the input for dispatching rules, conventional heuristics and our proposed EA, (MCMP-SRI). Two types of random arrivals for each instance were generated: early, that is in the interval [0,(*dj-pj*)/2], and late that is in the interval [(*dj-pj*)/2,(*dj-pj*)]. Many series of runs were performed for each algorithm on each instance. By using the static case as a relaxation of the dynamic case, we provided as upper bounds results from a best performer EA, obtained in previous works [14]. As two algorithms were designed, to compare their performance we established a percentile difference with the "best of two" performer defined as follows:

DTbest = ((Best – Best $_{instance}$)/ Best $_{instance}$)100
It is the percentile difference between the best individual provided by the considered algorithm and the best individual provided by the algorithm with best performance (for a particular instance).

The following tables summarize results and are organized as follows. The first column identifies the instance, the second column indicates the upper bound, the third column indicates the heuristic "associated" with the heuristic version of the algorithm, fourth and fifth columns indicate mean values for the best obtained objective values under each version of the algorithm, and the last two columns indicate the *DTbest* value for each version. At the bottom of the tables, average, minimun and maximum *DTbest* values are indicated.

5.1 Partial dynamism

In partial dynamism the heuristic "associated" with *Dyna-S-Heuristic* is the heuristic which provided the best total schedule for the static case. Tables 1 and 2 show the results obtained under each approach (early and late arrivals) for the selected instances. For each algorithm the minimum Average Tardiness values (*Best*) and the corresponding *DTbest* values are recorded. Boldfaced values indicate the best performer(s) algorithm(s) for each instance.

Case 1: Partial Dynamism with early arrival of jobs

Instance	Upper Bound	Best			DTbest	
		Dyna-S-Heuristic		Dyna-S-EA	Dyna-S-Heuristic	Dyna-S-EA
1	11.98	**EDD**	13.05	***11.98***	8.93	0.00
6	73.15	**HDS**	83.60	***73.15***	14.29	0.00
11	191.30	**HDS**	203.07	**196.90**	3.13	0.00
19	509.25	**SPT**	542.85	**515.13**	5.38	0.00
21	522.50	**SPT**	525.20	***522.50***	0.52	0.00
31	71.32	**RM**	84.95	**74.45**	14.10	0.00
36	183.18	**HDS**	199.88	**184.38**	8.41	0.00
41	374.45	**HDS**	400.70	***374.45***	7.01	0.00
46	369.35	**SPT**	373.58	***369.35***	1.15	0.00
56	16.17	**RM**	30.28	**21.92**	38.14	0.00
61	150.48	**HDS**	171.57	**154.68**	10.92	0.00
66	395.90	**SPT**	453.98	***395.90***	14.67	0.00
71	449.23	**SPT**	468.42	***449.23***	4.27	0.00
81	3.20	**EDD**	4.85	***3.20***	51.56	0.00
86	81.88	**EDD**	127.35	***81.88***	55.53	0.00
91	329.95	**RM**	383.67	***329.95***	16.28	0.00
96	639.65	**SPT**	657.75	***639.65***	2.83	0.00
111	210.80	**RM**	275.02	***210.80***	30.46	0.00
116	242.90	**RM**	319.12	***242.90***	31.38	0.00
121	576.57	**SPT**	598.03	***576.57***	3.72	0.00
				Avg	**16.13**	**0.00**
				Min	**0.52**	**0.00**
				Max	**55.53**	**0.00**

Table 1. Partial dynamism. Best and DTbest values for each algorithm with early arrival of jobs

For early arrivals, table 1 indicates that *Dyna-S-EA* is the best performer with a mean average *DTbest* of 0.00, meaning that it was the best algorithm for every instance, while *Dyna-S-Heuristic* shows mean values of 16.13, 0.52, and 55.53 for average, minimum and maximum *DTbest*, respectively. Indicated by boldfaced-italic values, we see that in 15 out of the 20 instances *Dyna-S-EA* reaches the upper bound.

Case 2: Partial Dynamism with late arrival of jobs

Instance	Upper Bound	Best			DTbest	
		Dyna-S-Heuristic		Dyna-S-EA	Dyna-S-Heuristic	Dyna-S-EA
1	11.98	**EDD**	**87.35**	91.57	0.00	4.83
6	73.15	**HDS**	**287.48**	331.42	0.00	15.28
11	191.30	**HDS**	**331.08**	385.73	0.00	16.51
19	509.25	**SPT**	829.50	**605.72**	36.94	0.00
21	522.50	**SPT**	796.88	***522.50***	52.51	0.00
31	71.32	**RM**	**355.37**	360.53	0.00	1.45
36	183.18	**HDS**	**383.30**	413.52	0.00	7.88
41	374.45	**HDS**	383.12	**378.95**	1.10	0.00
46	369.35	**SPT**	370.50	**369.53**	0.26	0.00
56	16.17	**RM**	**219.50**	249.60	0.00	13.71
61	150.48	**HDS**	**212.23**	236.32	0.00	11.35
66	395.90	**SPT**	**921.62**	929.58	0.00	0.86
71	449.23	**SPT**	453.38	***449.23***	0.92	0.00
81	3.20	**EDD**	81.70	**75.33**	8.46	0.00
86	81.88	**EDD**	127.35	***81.88***	55.53	0.00
91	329.95	**RM**	381.55	**330.63**	15.40	0.00
96	639.65	**SPT**	647.48	***639.65***	1.22	0.00
111	210.80	**RM**	274.77	**211.68**	29.80	0.00
116	242.90	**RM**	317.15	***242.90***	30.57	0.00
121	576.57	**SPT**	588.65	**576.60**	2.09	0.00
				Avg	**11.74**	**3.59**
				Min	**0.00**	**0.00**
				Max	**55.53**	**16.51**

Table 2. Partial dynamism. Best and DTbest values for each algorithm with late arrival of jobs

For late arrivals, we can see in table 2, that *Dyna-S-EA* is again the best performer with 3.59, 0.00, and 16.51 for average, minimum and maximum *DTbest* mean values, respectively, contrasting with *Dyna-S-Heuristic* which shows 11.74, 0.00, and 55.53 for the corresponding values. We also see that in 5 out of the 20 instances *Dyna-S-EA* reaches the upper bound, while *Dyna-S-Heuristic* never reaches the upper bound.

5.2 Total dynamism

In total dynamism the heuristic "associated" with *Dyna-H-Heuristic* is the heuristic that, used to determine which job to schedule next, provided the best final schedule. Tables 3 and 4 show the results obtained under each arrival situation (early and late). Minimum Average Tardiness values (Best) and the corresponding *DTbest* values are recorded. Boldfaced values indicate the best performer(s) for each instance.

Case 3: Total Dynamism with early arrival of jobs

Instance	Upper Bound	Best			DTbest	
		Dyna-H-Heuristic		Dyna-H-EA	Dyna-H-Heuristic	Dyna-H-EA
1	11.98	**EDD**	13.05	***11.98***	8.93	0.00
6	73.15	**SPT**	93.32	**76.27**	22.35	0.00
11	191.30	**SPT**	214.27	**195.55**	9.57	0.00
19	509.25	**SPT**	542.85	**516.63**	5.08	0.00
21	522.50	**SPT**	525.20	**523.44**	0.34	0.00
31	71.32	**RM**	95.80	**74.33**	28.88	0.00
36	183.18	**SPT**	231.50	**188.64**	22.72	0.00
41	374.45	**SPT**	416.45	**377.04**	10.45	0.00
46	369.35	**SPT**	373.58	**371.06**	0.68	0.00
56	16.17	**EDD**	31.32	**16.77**	86.76	0.00
61	150.48	**SPT**	238.58	**156.99**	51.97	0.00
66	395.90	**SPT**	453.08	**397.97**	13.85	0.00
71	449.23	**SPT**	466.70	**450.51**	3.59	0.00
81	3.20	**EDD**	4.85	**3.40**	42.65	0.00
86	81.88	**EDD**	127.35	**85.68**	48.63	0.00
91	329.95	**SPT**	399.60	**332.65**	20.13	0.00
96	639.65	**SPT**	657.75	**640.69**	2.66	0.00
111	210.80	**RM**	331.60	**214.61**	54.51	0.00
116	242.90	**SPT**	312.92	**244.18**	28.15	0.00
121	576.57	**SPT**	598.03	**577.67**	3.52	0.00
				Avg	**23.27**	**0.00**
				Min	**0.34**	**0.00**
				Max	**86.76**	**0.00**

Table 3. Total dynamism. Best and DTbest values for each algorithm with early arrival of jobs

For early arrivals, in table 3 it is shown that *Dyna-H-EA* is the best performer with a mean average *DTbest* of 0.0, reaching the upper bound in one occasion and being near of them in the remaining instances. *Dyna-H-Heuristic* shows mean values of 23.27, 0.34, and 86.76 for average, minimum and maximum *DTbest*, respectively.

Case 4: Total Dynamism with late arrival of jobs

Instance	Upper Bound	Best			DTbest	
		Dyna-H-Heuristic		Dyna-H-EA	Dyna-H-Heuristic	Dyna-H-EA
1	11.98	**SPT**	84.05	**59.33**	41.67	0.00
6	73.15	**SPT**	275.52	**268.57**	2.59	0.00
11	191.30	**SPT**	331.05	**321.08**	3.11	0.00
19	509.25	**SPT**	584.40	**575.60**	1.53	0.00
21	522.50	**SPT**	**523.25**	523.33	0.00	0.02
31	71.32	**SPT**	315.55	**275.98**	14.34	0.00
36	183.18	**SPT**	344.48	**315.80**	9.08	0.00
41	374.45	**SPT**	400.70	**387.33**	3.45	0.00
46	369.35	**SPT**	370.50	**369.96**	0.15	0.00
56	16.17	**SPT**	194.15	**140.24**	38.44	0.00
61	150.48	**SPT**	234.43	**207.18**	13.15	0.00
66	395.90	**SPT**	**922.75**	937.31	0.00	1.58
71	449.23	**SPT**	453.38	**449.59**	0.84	0.00
81	3.20	**EDD**	81.70	**48.45**	68.63	0.00
86	81.88	**EDD**	127.35	***81.88***	55.53	0.00
91	329.95	**SPT**	356.75	**331.35**	7.67	0.00
96	639.65	**SPT**	647.47	**640.35**	1.11	0.00
111	210.80	**SPT**	268.52	**211.70**	26.84	0.00
116	242.90	**SPT**	269.33	**242.91**	10.88	0.00
121	576.57	**SPT**	588.65	**576.95**	2.03	0.00
				Avg	**15.05**	**0.08**
				Min	**0.00**	**0.00**
				Max	**68.63**	**1.58**

Table 4. Total dynamism. Best and DTbest values for each algorithm with late arrival of jobs

For late arrivals, table 4 indicates that *Dyna-H-EA* is the best performer with mean values of 0.08, 0.00, and 1.58 for average, minimum and maximum *DTbest*, respectively, reaching the upper bound in one occasion and being near of them in the eight of the remaining instances. *Dyna-H-Heuristic* shows mean values of 15.05, 0.00, and 68.63 for average, minimum and maximum *DTbest*, respectively.

6. Conclusions

The static scheduling problem minimizing average tardiness for single machine environments, is by itself a difficult problem and some conventional and evolutionary heuristics were developed to provide optimal or quasi-optimal solutions. For this objective we could not find OR-library or other well-known published benchmarks, to compare experimental results. More difficult is to find them even in the simplest form of dynamism. Consequently, we built our own test suite with data (p_j ,d_j) extracted from 20 selected instances of the OR-library benchmarks for a related objective

(weighted tardiness), with 40-jobs problem size. This data was the input for dispatching rules, conventional heuristics and evolutionary algorithms.

The present work showed different approaches to face partial and total dynamism.

For the partial dynamism case we proposed the *Dyna-S* algorithm in two different versions. *Dyna-S* is based on the knowledge provided by, a conventional (*EDD, SPT, Hodgson*) or an evolutionary (*MCMP-SRI*) heuristic used previously to solve static cases. The job order provided by a total schedule is used then as a dispatching rule. In other words the algorithm uses the outcome of evolutionary or conventional heuristics, which provide quasi-optimal solutions to a similar static case for the whole set of jobs to be scheduled. This outcome, used as a surrogate, is the element to help decision establishing an static priority of dispatch once for all, which is used each time the machine is available.

For the total dynamism case *Dyna-H* was proposed. The algorithm applies different heuristics to dynamically determine which job to schedule when the machine becomes available. The two versions shown here differ in the heuristic selected for decision: 1) some of the conventional heuristics, 2) a hybrid algorithm. Depending on the waiting-queue length and the nature of the partial schedule each time the machine becomes available, this hybrid algorithm uses an enumerative approach, EDD, R&M, or an evolutionary approach. *Dyna-H* schedules at each decision point, selecting the most suitable waiting job according to an ordering emanated from the heuristic applied.

The results obtained through this study can be summarized as follows:

- For any case of dynamism and arrival type the versions using the evolutionary approach (*Dyna-S-EA* and *Dyna-H-EA*) globally outperforms that versions using the conventional heuristics (*Dyna-S-Heuristic* and *Dyna-H-Heuristic*).

At the light of these results future work will be devoted to larger problems, different job arrival distributions and variants of the *Dyna-S* and *Dyna-H* approaches.

7. Acknowledgements

We acknowledge the co-operation of the LIDIC for providing new ideas and constructive criticisms. Also to the Universidad Nacional de San Luis, the Universidad Nacional de La Patagonia Austral, and the ANPCYT from which we receive continuous support.

8. References

[1] Beasley J.E. “Common Due Date Scheduling”, OR Library, http://mscmga.ms.ic.ac.uk/

[2] Crauwels H.A.J., Potts C.N. and Van Wassenhove L.N. “Local search heuristics for the single machine total weighted tardiness scheduling problem”, Informs Journal on Computing 10, 341-350. 1998.

[3] Baker K. R., “Introduction to sequencing and scheduling” Willey New York 1974.

[4] Eiben A.E., Raué P.E., and Ruttkay Z., "Genetic algorithms with multi-parent recombination", Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, Springer-Verlag, 1994, number 866 in LNCS, pp. 78-87.

[5] Eiben A.E., Van Kemenade C.H.M., and Kok J.N., "Orgy in the computer: Multi-parent reproduction in genetic algorithms". Proceedings of the 3rd European Conference on Artificial Life, Springer-Verlag, 1995, number 929 in LNAI, pages 934-945.

[6] Esquivel S., Leiva A., Gallard R., "Multiple Crossover per Couple in Genetic Algorithms", Proceedings of the Fourth IEEE Conference on Evolutionary Computation (ICEC'97), Indianapolis, USA, April 1997, pp 103-106.

[7] Esquivel S., Leiva A., Gallard R., "Couple Fitness Based Selection with Multiple Crossover per Couple in Genetic Algorithms". Proceedings of the International Symposium on Engineering of Intelligent Systems (EIS´98), La Laguna, Tenerife, Spain, February 1998, pp 235-241.

[8] Esquivel S., Leiva H., Gallard R., "Multiple crossovers between multiple parents to improve search in evolutionary algorithms", Proceedings of the Congress on Evolutionary Computation (IEEE). Washington DC, 1999, pp 1589-1594.

[9] Morton T., Pentico D., "Heuristic scheduling systems", Wiley series in Engineering and technology management. John Wiley and Sons, INC, 1993.

[10] Pandolfi D., Vilanova G., De San Pedro M., Villagra A., Gallard R., "Adaptability of Multirecombined Evolutionary Algorithms in the single-machine common due date problem." Proceedings of the Multiconference on Systemics, Cybernetics and informatics. Orlando, Florida July 2001.

[11] Pandolfi D., Vilanova G., De San Pedro M., Villagra A., Gallard R. "Multirecombining studs and immigrants in evolutionary algorithm to face earliness-tardiness scheduling problems". Proceedings of the International Conference in Soft Computing. University of Paisley, Scotland, U.K., June 2001, pp.138

[12] Pandolfi D., De San Pedro M., Villagra A., Vilanova G., Gallard R.- "Studs mating immigrants in evolutionary algorithm to solve the earliness-tardiness scheduling problem" . In Cybernetics and Systems of Taylor and Francis Journal, Vol. 33 Nro. 4, pp 391-400 (U.K.) June 2002.

[13] Pandolfi D., De San Pedro M., Villagra A., Vilanova G., Gallard R. "Multirembining Random and Seeds with Studs in evolutionary algorithm to solve W-T Scheduling problems" In proceedings of ACIS International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA-02), pp 133,138, Foz Iguazú, Brasil 2002.

[14] Pandolfi D., Lasso M., De San Pedro M., Villagra A., Gallard R. – "Evolutionary algorithms to solve average tardiness problems in single machine environments"- Proceedings of the International Conference on Computer Science, Software Engineering Information Technology, e-bussness and Aplications (CSITeA03), pp 444-449, Rio de Janeiro, June 2003, Brazil.

[15] Pinedo M., "Scheduling: Theory, Algorithms and System." First edition Prentice Hall, 1995.

[16] Rachamadugu R.V., Morton T.E., "Myopic heuristics for the single machine weighted tardiness problem". GSIA, Carnigie Mellon University, Pittsburgh, PA. 1982., Working paper 30-82-83.