ALGORITHMS TO SOLVE THE DYNAMIC WEIGHTED TARDINESS PROBLEM

Lasso M., Pandolfi D., de San Pedro M., Villagra A., Vilanova G.
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; gvilanov@satlink.com
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 static scheduling problems it is assumed that jobs are ready at zero time or before processing begins. In dynamic scheduling problems a job arrival can be given at any instant in the time interval between zero and a limit established by its processing time, ensuring to accomplish it before the due date deadline. In the cases where the arrivals are near to zero the problem comes closer to the static problem, otherwise the problem becomes more restrictive.
This paper proposes two approaches for resolution of the dynamic problem of Total Weighted Tardiness for a single machine environment. The first approach uses, as a list of dispatching priorities a schedule, which an evolutionary algorithm found as the best for a similar static problem: same job features, processing time, due dates and weights. The second approach uses as a dispatching priority a schedule created by a robust non-evolutionary heuristic. The details of implementation of the proposed algorithms and results for a group of selected instances are discussed in this work.

[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. Introducción

Manufacturing organizations are frequently subject to several sorts 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 [11]. Particularly in *Single Machine Scheduling Problems* (SMSP) a set of jobs should be planned on a machine, each job is processed one at a time, conflicting objectives should be completed 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 jobs are known before scheduling starts, while in *dynamic* problems, job release times are not fixed at a single point, jobs arrive at different times.

Evolutionary algorithms have been successfully applied to solve scheduling problems [12,13,16,17]. Current trends in evolutionary algorithms make use of multiparent [4, 5, 6] and multirecombined approaches [7, 8, 9]. 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 Weighted Tardiness Problem [10,18] 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 heuristic are techniques that provide a reasonably good solution in relatively short time.

In this paper we propose the 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 unpredicted job arrival times. Two approaches are implemented using the outcome of different algorithms for the static case. The first uses as a dispatching rule the order dictated by the schedule provided by an evolutionary algorithm while the second uses the outcome of the R&M and Covert heuristics.

2. Dynamic Scheduling for Single Machine Problems

In the weighted tardiness single machine problem *n* jobs should be planned without interruption. For each job *j* (j = 1,...,n) with processing time *pj* and due date *dj*, exists a penalty *wj* for each tardy unit. The objective of this problem is to find a sequence that minimizes:

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

where the tardiness of a job, is given by $T_j = \max\{C_{j-1} + p_j - d_j, 0\}$. Even with this simple formulation, this model leads to an optimization problem that is NP-Hard [14].

In the static weighted tardiness problem all jobs 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 unpredicted times. We restrict our study to this kind of dynamism.

However, once the jobs arrive to the system for their processing producing 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 approach that provide good solutions can be a suitable surrogate for the cases of dynamism. We can

consider the static weighted tardiness problem, as a relaxation of the dynamic problem, where all arrival times are equal to zero, that is *rj=0* for all *j*.

3. Dynamic Scheduling Algorithms based on Static Scheduling.

This work proposes two approaches to solve the dynamic single machine scheduling problem. In both approaches, arrivals times *rj* are generated randomly for each task of the selected instances. The *rj* values are created once for all, considering two types of arrivals: early and late arrivals.
The first approach called *WT-Dyna-S* is based on the knowledge provided by, an evolutionary algorithm (MCMP-SRI) or by different dispatching heuristics used previously to solve static cases. Here, we use as a dispatching rule the job order provided by a total schedule generated by an evolutionary algorithm (*WT-Dyna-S-EA*), or by conventional heuristics (*WT-Dyna-S-R&M*) and (*WT-Dyna-S-Covert*). To schedule a job, an arrival queue is created with those jobs whose *rj* are smaller or equal than the time *t* when the machine is available for the processing **(1)**. From the waiting queue, the job that appears first in the schedule ordering is selected to be allocated first **(3)**, that is the one with higher priority, in the schedule of dispatching priority **(2)** (*see Figure 1)*. Once a job is planned, it is removed from the queue. This process is repeated each time when the resource is available and while there are jobs in the waiting queue.

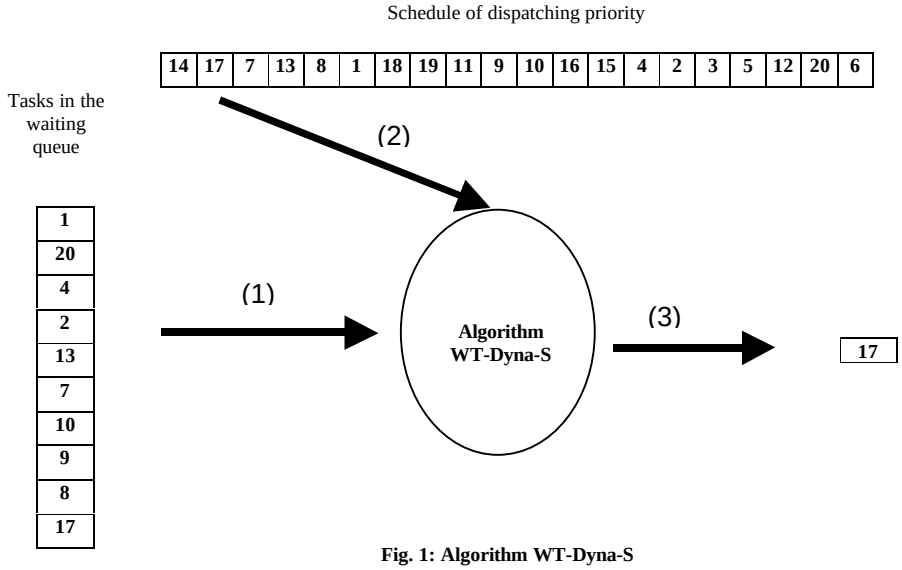

Fig. 1: Algorithm WT-Dyna-S

In the second approach, called *WT-Dyna-H*, jobs in the waiting queue are planned according to some dispatching rule which generates a partial schedule. Here, we use R&M and Covert heuristics (*WT-Dyna-H-R&M* and *WT-Dyna-H-Covert*). Again, a waiting queue is generated with those jobs whose *rj* are smaller or equal than the time *t* when the machine is available for the processing **(1)**. Now we run the heuristic to produce a partial schedule (reordering the available jobs) **(2)**. The algorithm uses this list of dispatching priorities to schedule the next job **(3)** by choosing the job in the first position of this list **(4)**. (*see Figure 2)* Once the job is planned, it is removed from the queue. This process is repeated each time when the resource is available and while there are jobs in the waiting queue.

The heuristics used are the following [15]:

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

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

here $Sj = [dj - (pj + Ch)]$ 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 times 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.

The Covert rule. This heuristic provides a schedule according to the following expression.

$$p_j = (w_j / p_j)\{1 - (S_j)^+ / kp_j\}^+$$

Under this heuristic the WSPT (Weighted Shortest Processing Time first) rule is modified by a slack factor, and processing times different than the job being considered are not taking into account. In the case of a single machine environment, the Covert rule works similar to R&M.

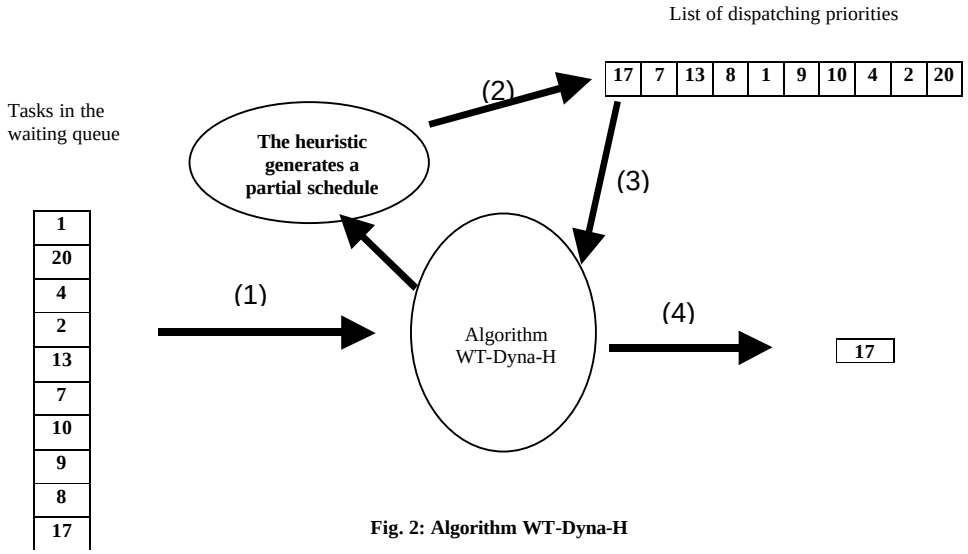

Fig. 2: Algorithm WT-Dyna-H

4. Experimental Tests and Results

The evolutionary algorithms were tested for 15 selected instances of 40-jobs problem size, extracted from the OR-library benchmarks [1, 2]. Two types of random arrivals for each instance were generated. In the first, arrivals are produced in the interval [0,(dj-pj)/2], that is with an early arrival approach. In the second, they are created in the interval [dj-pj)/2,(dj-pj)] with a later arrival approach.

As five algorithms were designed, to compare their performance we established a percentile difference with the "best of five" 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 best performer for a particular instance.

Tables 1 and 2 show the results obtained under each approach (early and tardy arrivals) for the selected instances. For each algorithm the minimum WT values (Best) and the corresponding DTbest values are recorded. Boldfaced values indicate the best performer(s) for each instance.

Instance	WT-Dyna-S-EA		WT-Dyna-S-R&M		WT-Dyna-S-Covert		WT-Dyna-H-R&M		WT-Dyna-H-Covert	
	Best	**DTbest**	**Best**	**DTbest**	**Best**	**DTbest**	**best**	**DTbest**	**Best**	**DTbest**
wt40-1	**913**	0.00	**913**	0.00	4297	370.65	1291	41.40	29714	3154.55
wt40-11	**17465**	0.00	17525	0.34	28132	61.08	53312	205.25	162198	828.70
wt40-19	**80899**	0.00	81048	0.18	81122	0.28	109087	34.84	179319	121.66
wt40-21	**77774**	0.00	77793	0.02	77802	0.04	85954	10.52	98690	26.89
wt40-26	**108**	0.00	**108**	0.00	198	83.33	252	133.33	25218	23250.00
wt40-31	**6575**	0.00	**6575**	0.00	16187	146.19	16792	155.39	98688	1400.96
wt40-41	**57640**	0.00	58430	1.37	58525	1.54	124883	116.66	189997	229.63
wt40-46	**64451**	0.00	64498	0.07	64505	0.08	82316	27.72	95256	47.80
wt40-56	**2099**	0.00	2565	22.20	3648	73.80	11779	461.17	40838	1845.59
wt40-66	**65386**	0.00	65525	0.21	65503	0.18	113733	73.94	156916	139.98
wt40-71	**90486**	0.00	**90486**	0.00	**90486**	0.00	120695	33.39	131729	45.58
wt40-91	**47683**	0.00	47771	0.18	48337	1.37	123201	158.38	174854	266.70
wt40-96	**126048**	0.00	126056	0.01	**126048**	0.00	173741	37.84	186045	47.60
wt40-116	**46770**	0.00	47151	0.81	48140	2.93	89009	90.31	125986	169.37
wt40-121	**122266**	0.00	122536	0.22	122995	0.60	177722	45.36	192915	57.78
Promedio		**0.00**		**1.71**		**49.47**		**108.37**		**2108.85**

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

Instances	WT-Dyna-S-EA		WT-Dyna-S-R&M		WT-Dyna-S-Covert		WT-Dyna-H-R&M		WT-Dyna-H-Covert	
	Best	**DTbest**	**Best**	**DTbest**	**Best**	**DTbest**	**Best**	**DTbest**	**Best**	**DTbest**
wt40-1	18449	2.87	18079	0.80	37660	109.98	**17935**	0.00	68839	283.82
wt40-11	48428	2.62	**47192**	0.00	76093	61.24	89245	89.11	220827	367.93
wt40-19	100780	2.00	**98802**	0.00	99585	0.79	120279	21.74	201767	104.21
wt40-21	**77774**	0.00	77786	0.02	77802	0.04	85947	10.51	98690	26.89
wt40-26	45955	2.53	45557	1.64	84087	87.60	**44823**	0.00	108634	142.36
wt40-31	**62003**	0.00	**62003**	0.00	98813	59.37	79957	28.96	181559	192.82
wt40-41	65368	5.57	**61922**	0.00	62900	1.58	130211	110.28	191288	208.92
wt40-46	**64484**	0.00	64491	0.01	64505	0.03	82316	27.65	95256	47.72
wt40-56	38365	13.63	**33764**	0.00	48850	44.68	63598	88.36	108843	222.36
wt40-66	192990	0.21	**192590**	0.00	193374	0.41	239265	24.24	274588	42.58
wt40-71	**90486**	0.00	**90486**	0.00	**90486**	0.00	120695	33.39	131729	45.58
wt40-91	**47893**	0.00	47972	0.16	48802	1.90	123954	158.81	175379	266.19
wt40-96	**126048**	0.00	126056	0.01	**126048**	0.00	173051	37.29	185755	47.37
wt40-116	**46770**	0.00	47151	0.81	48140	2.93	87780	87.68	125986	169.37
wt40-121	**122383**	0.00	122723	0.28	123182	0.65	177722	45.22	192089	56.96
Promedio		**1.96**		**0.25**		**24.75**		**50.88**		**148.34**

Table 2. Best and DTbest values for each algorithm with tardy arrival of jobs

Table 1 indicates that *WT-Dyna-S-EA* is the best performer for any of the considered instances. The values it obtained are the same as those reached by MCMP-SRI for the static case and match

those provided by the OR-library. In this case the result used as a surrogate gives an indication that the generated early arrival times do not influence on the schedules performance. According to the table summary, *WT-Dyna-S-R&M* is the second best performer and *WT-Dyna-H-Covert* is the worst. Furthermore, those methods based on a total schedule (*WT-Dyna-S*) perform better than those, which are building partial schedules at the time the machine is free (*WT-Dyna-S*).

Table 2 indicates that, in average, *WT-Dyna-S-R&M* is the best performer in the case of tardy job arrivals. *WT-Dyna-S-EA* is the second best performer and *WT-Dyna-H-Covert* is the worst one. Again, those methods based on a total schedule (*WT-Dyna-S*) perform better than those, which are building partial schedules at the time the machine is free (*WT-Dyna-S*).

It is important to remark that the *WT-Dyna-S* algorithm requires a lesser computational effort than the WT-Dyna-H algorithm, because the latter must repeatedly evaluate in each point of decision (machine free) the job to schedule from those that are in the waiting queue.

5. Conclusions

The static scheduling problem minimizing weighted tardiness for single machine environments, is a difficult problem by itself and some conventional and evolutionary heuristics were developed to provide optimal or quasi optimal solutions. The solutions provided could be contrasted against well-known OR-library benchmarks. This is not the case, even in the simplest for of dynamism where job arrivals are unpredicted. But usually this is the normal situation in a production line.

The present work showed two different approaches to face the problem. The first one uses the outcome of evolutionary and 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 a priority of dispatch once for all. The second approach schedules at each decision point, selecting the most suitable job according to an ordering emanated from a rapid conventional heuristic. The comparisons we have done indicate that the first approach provide better results in both considered situations, early and late arrivals. Also, these results are obtained with lesser computational effort because in the first approach a total schedule provided once for all is used, while in the second approach a re-scheduling is necessary at each decision point.

Future work will be devoted to larger problems and different job arrival distributions.

6. Acknowledgements

We acknowledge the co-operation of the project group 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.

7. References

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

[2] H.A.J.Crauwels, C.N.Potts and L.N.Van Wassenhove "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] A.E.Eiben, P.E.Raué, and Z.Ruttkay, “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] A.E. Eiben, C.H.M. Van Kemenade, and J.N. Kok, “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] A.E. Eiben and. Th. Bäck, “An empirical investigation of multi-parent recombination operators in evolution strategies”. Evolutionary Computation, 5(3):347-365, 1997.

[7] S. Esquivel, A. Leiva, R. Gallard, “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.

[8] S. Esquivel, A. Leiva, R. Gallard, “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.

[9] S. Esquivel, H. Leiva,.R. Gallard, “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.

[10] D. Pandolfi, M. De San Pedro, A. Villagra,G. Vilanova, R Gallard “Stud and Immigrants in multirecombined evolutionary algorithm to face Weighted Tardiness Scheduling problems” Proceedings 7mo Congreso Argentino de la Computación, El Calafate, Argentina, Otubre 2001 vol II pp 1251.

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

[12] D. Pandolfi, G. Vilanova, M. De San Pedro, A. Villagra, R Gallard “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., June2001, pp.138

[13] D. Pandolfi; G. Vilanova; M. De San Pedro; A. Villagra, R. Gallard: “Solving the Single-Machine Common Due Date Problem Via Stud and Inmigrants in Evolutionary Computation”, World Multiconference on Systemics, Cybernetics and Informatics, Orlando July 2001 pp 409-413

[14] M. Pinedo, “Scheduling: Theory, Algorithms and System.” First edition Prentice Hall, 1995.

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

[16] 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.

[17] Pandolfi D., Vilanova G., De San Pedro M., Villagra A., Gallard R., “Evolutionary Algorithms to minimize earliness-tardiness penalties from a common due date”. Proceedings of the Multiconference on Systemics, Cybernetics and informatics. Orlando, Florida July 2001.

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