Investigação - Trabalhos em curso - nº 127, Abril de 2003

USING INSTANCE STATICTICS TO DETERMINE THE LOOKAHEAD PARAMETER VALUE IN THE ATC DISPATCH RULE

Making a good heuristic better

Jorge M. S. Valente

FACULDADE DE ECONOMIA

UNIVERSIDADE DO PORTO

www.fep.up.pt

Using instance statistics to determine the lookahead parameter value in the ATC dispatch rule: Making a good heuristic better

Jorge M. S. Valente
Faculdade de Economia do Porto
Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
e-mail: jvalente@fep.up.pt

April 25, 2003

Abstract

The Apparent Tardiness Cost (ATC) heuristic is one of the best performing dispatch rules for the weighted tardiness scheduling problem. This heuristic uses a lookahead parameter whose value must be specified. In this paper we develop a function that maps some instance statistics into an appropriate value for the lookahead parameter. This function is compared with some fixed values that have been previously used. The computational results show that the ATC heuristic performs better when the lookahead parameter value is determined by the proposed function.

Keywords: scheduling, weighted tardiness, dispatch rule, instance statistics

Resumo

A *Apparent Tardiness Cost* (ATC) é uma das melhores heurísticas de construção para o problema de sequenciamento que visa minimizar a *weighted tardiness*. Esta heurística utiliza um parâmetro de pesquisa cujo valor é necessário especificar. Neste artigo é desenvolvida uma função que determina um valor apropriado para este parâmetro com base em determinadas estatísticas da instância em causa. Esta função é comparada com alguns valores fixos habitualmente utilizados. Os resultados computacionais mostram que a função proposta permite melhorar o desempenho da heurística ATC.

Palavras-chave: sequenciamento, problema *weighted tardiness*, heurística de construção, estatísticas da instância

1 Introduction

In this paper we consider the single machine total weighted tardiness scheduling problem. This problem can be stated as follows. A set of n independent jobs $\{J_1, J_2, \cdots, J_n\}$ has to be scheduled without preemptions on a single machine that can handle only one job at a time. The machine and the jobs are assumed to be continuously available from time zero onwards. Job $J_j, j = 1, 2, \cdots, n$, requires a processing time p_j and has a due date d_j and a positive weight or penalty w_j. For any given schedule, the tardiness of J_j can be defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of J_j. The objective is then to find the schedule that minimizes the total weighted tardiness $\sum_{j=1}^{n} w_j T_j$.

Lenstra, Rinnooy Kan and Brucker [5] show that the total weighted tardiness problem is strongly NP-hard. Several exact approaches used in solving this problem have been surveyed and tested by Abdul-Razaq, Potts and Van Wassenhove [1]. Hoogeveen and Van de Velde [3] show that better Lagrangean lower bounds can be obtained by reformulating the problem using slack variables. Akturk and Yildirim [2] present a new dominance rule that can be used to eliminate nodes in an exact procedure, as well as to improve both lower and upper bounding schemes. Several heuristics and dispatch rules have also been proposed. The Apparent Tardiness Cost (ATC) dispatch rule was developed by Rachamadugu and Morton [7], and several computational studies ([6], [8]) have shown that it's one of the best construction heuristics available for the weighted tardiness problem.

The effectiveness of the ATC heuristic depends on the value of a lookahead parameter. In previous studies, this parameter is set at a fixed value. In this paper we propose using certain instance statistics, or factors, to determine an appropriate value for the lookahead parameter. The computational results show that this approach improves the performance of the ATC heuristic.

The remainder of the paper is organized as follows. In section 2 we describe the instance factors and the experiments performed to determine the function that maps those instance statistics into a value for the lookahead parameter. Computational results are reported in section 3. Finally, some concluding remarks are given in section 4.

2 A function for determining the lookahead parameter value

The ATC heuristic uses the following priority index $I_j(t)$ to determine the job J_j to be scheduled at any instant t when the machine becomes available:

$$I_j(t) = \frac{w_j}{p_j} \exp\left(-\frac{[d_j - t - p_j]^+}{kp}\right),$$

where p is the average processing time and k is the lookahead parameter. The value of k should be releated to the number of competing critical and near-critical jobs.

We now describe the factors or statistics that characterize an instance and may affect the choice of k. The instance size n and variability of the processing times p_j and the penalties w_j may influence the most effective value of k. The remaining two factors, which are associated with the due dates, are the tardiness factor TF and the range of due dates RDD. The factor TF can be defined as $TF = 1 - \left(\overline{d}/C_{\max}\right)$, where $\overline{d}$ is the average of the due dates and $C_{\max}$ is the makespan. If the tardiness factor is high, the average due date will be low, and most jobs will likely be tardy. Conversely, when TF is low, most jobs should be completed on time. The factor RDD, which is a measure of the due dates dispersion around their average, is defined as $(d_{\max} - d_{\min})/C_{\max}$, where $d_{\max}$ and $d_{\min}$ are, respectively, the maximum and the minimum value of the due dates.

The experiments performed to determine the mapping function were similar to those used by Lee, Bhaskaran and Pinedo [4] for the weighted tardiness problem with sequence-dependent setups. A set of problems with 25, 50, 100, 250, 500 and 1000 jobs was randomly generated as follows. For each job J_j an integer processing time p_j and an integer penalty w_j were generated from one of the two uniform distributions [1, 10] and [1, 100], to create low and high variability, respectively. For each job J_j, an integer due date d_j is generated from the uniform distribution $[C_{\max}(1 - TF - RDD/2), C_{\max}(1 - TF + RDD/2)]$, where TF was set at 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0, and RDD was set at 0.2, 0.4, 0.6 and 0.8. The values of the factors involved in the instance generation process are summarized in Table 1. For each combination of instance size, processing time and penalty variability, TF and RDD, 20 instances were randomly generated.

An initial test was first performed to determine, for each factor combination, the range where the best values of k were concentrated. A more detailed test was

Factors	Settings
Number of jobs	25, 50, 100, 250, 500, 1000
Processing time and penalty variability	[1, 10], [1, 100]
Tardiness factor	0.0, 0.2, 0.4, 0.6, 0.8, 1.0
Range of due dates	0.2, 0.4, 0.6, 0.8

Table 1: Experimental design

then performed on these ranges. In this test, we considered values of the lookahead parameter ranging from the lower to the upper limit of the range, with 0.2 increments, and computed the total weighted tardiness for each instance and each value of the lookahead parameter. For each instance we then identified all values of the parameter k which led to objective function values which were less than or equal to $(1 + \alpha) Z$, where Z is the minimum of the computed objective function values and $\alpha > 0$. The α is introduced so that a certain proportion of the lookahead parameter values that lead to an objective function value near the minimum are considered, instead of only the value (or values) that lead to the best schedule. The value of α was selected so that, for each factor combination, an average of three values of k were selected per instance. The method chosen for determining the value of α automatically compensates for the different variability the objective function value may exhibit for different factor combinations. The average of the lookahead parameter values selected for each instance is taken as the best estimate of k for that instance. The best estimate of the parameter k for a given factor combination is then given by the average of the best estimates for all twenty instances sharing that factor combination.

From the results of these experiments we could conclude the following. The processing time and penalty variability does not have a significant impact on k. The lookahead parameter is sensitive to the remaining instance factors (instance size, TF and RDD). The value of k is non-decreasing in the instance size, non-increasing in RDD and symmetric around $TF = 0.5$. These factors, however, are interrelated, since their effect on k depends on the values of the other factors. A simple formula would then fail to capture the variety of effects and interactions among the factors, and we therefore decided to develop separate formulas for different TF values. This approach yields a mapping function that is more accurate, even though it's also somewhat more cumbersome. In table 2 we present formulas that determine the value of k as a function of n and RDD for TF values of 0.0, 0.2 and 0.4. Given the symmetry in the lookahead parameter values, any $TF > 0.5$ is converted to an

equivalent value of $1 - TF$. Linear interpolation is then used to determine k for other TF values. When the tardiness factor is equal to 0.2 or 0.4, we present the formulas for selected RDD values. The lookahead parameter is linear (or piecewise linear) in the RDD factor for these two TF values, but the straight line expression is unwieldy, so we chose to present these simpler expressions. The experiments also showed that the best values of k are rarely below 0.6. Therefore, if the function ever returns a value lower than 0.6, k is set to 0.6 instead.

TF	RDD	k
0.0	all	0.6
0.2	0.2	$0.115n^{0.8}$
	≥ 0.4	$(0.22 - 0.175\,(RDD - 0.4))\ln(n)$
0.4	0.2	$0.07n$
	0.8	$0.31n^{0.35}$

Table 2: Mapping function

3 Computational results

In this section we present the results from the computational tests. The set of test problems was generated as described in the previous section. The algorithms were coded in Visual C++ 6.0 and executed on a Pentium IV-1700 personal computer. Previous studies have indicated that k should be set at a value between 0.5 and 2.0. Therefore, we compared the function-based version of the ATC heuristic with the fixed values 0.5, 1.0, 1.5 and 2.0. In the following, and for each combination of instance size and processing time and penalty variability, we only present the results for the best fixed value (best k). Computational times will not be presented, since the increased computational effort required by the mapping function could hardly be noticed, even for the largest instances.

In table 3 we present the average objective function values (mean ofv) and the relative difference in average weighted tardiness, calculated as $\frac{F-K}{K} * 100$, where F and K are the average weighted tardiness values of the function and fixed value versions, respectively. Table 4 gives the number of instances for which the function-based version performs better (<), equal (=) or worse (>) than the fixed value version. We also performed a test to determine if the difference between the two versions is statistically significant. Given that the heuristics were used on exactly

the same problems, a paired-samples test is appropriate. Since some of the hypothesis of the paired-samples t-test were not met, the non-parametric Wilcoxon test was selected. Table 4 also includes the significance (sig.) values of this test, i.e., the confidence level values above which the equal distribution hypothesis is to be rejected. From the results presented in these two tables, we can conclude that the function-based version outperforms its fixed value counterpart, since the average objective function value is lower and it provides better results for most of the test instances. The Wilcoxon test values also indicate that the differences in distribution between the two versions is statistically significant. The relative improvement over the fixed value version increases with the instance size and the processing time and penalty variability.

	var. 1-10			var. 1-100		
	mean ofv			mean ofv		
n	function	best k	% ofv ch.	function	best k	% ofv ch.
25	1759	1764	-0,27	138001	138119	-0,09
50	6532	6582	-0,75	501234	505127	-0,77
100	25114	25520	-1,59	1895957	1930072	-1,77
250	152441	156304	-2,47	11630510	11986324	-2,97
500	600742	618276	-2,84	46154087	47787819	-3,42
1000	2382689	2454444	-2,92	182334086	189148723	-3,60

Table 3: Average weighted tardiness and relative difference

	var. 1-10				var. 1-100			
	function vs best k				function vs best k			
n	<	=	>	sig.	<	=	>	sig.
25	163	263	54	0,000	187	215	78	0,000
50	250	152	78	0,000	241	149	90	0,000
100	300	130	50	0,000	310	131	39	0,000
250	304	137	39	0,000	302	142	36	0,000
500	302	133	45	0,000	312	131	37	0,000
1000	305	135	40	0,000	307	130	43	0,000

Table 4: Comparison of heuristic average weighted tardiness values and statistical test

In table 5 we present the effect of TF and RDD on the relative difference in average weighted tardiness for instances with 100 jobs and low processing time and penalty variability. The results for other instances are similar to those presented. Both versions of the heuristic generate a schedule with no tardy jobs for all instances

with $TF = 0.0$ or $TF = 0.2$ and $RDD = 0.6; 0.8$. The weighted tardiness problem is easier when TF is equal to 0.0 (1.0), since most jobs will be on time (tardy). For these instances, there is usually little room for improvement, as can be seen from the results in table 5. As the tardiness factor comes closer to 0.5, the problem becomes harder, and the improvement provided by the function version increases significantly, particularly for a range of due dates of 0.2 or 0.4.

	RDD			
TF	0.2	0.4	0.6	0.8
0.0	—	—	—	—
0.2	-11,04	-29,11	—	—
0.4	-25,95	-11,30	-1,24	-5,91
0.6	-15,34	-10,12	-3,94	0,22
0.8	-2,13	0,05	-0,45	-0,29
1.0	-0,10	-0,11	-0,14	-0,12

Table 5: Relative difference in average weighted tardiness for instances with 100 jobs and low processing time and penalty variability

We also compared the function and fixed value versions on instances with TF and RDD values of 0.1, 0.3, 0.5, 0.7 and 0.9. The results for these instances are presented in tables 6 and 7. The function-based heuristic once again outperforms the fixed value approach, and the relative improvement is even higher for this second set of instances. These results confirm the validity of the proposed mapping function (and the linear behaviour it assumes between TF values), and illustrate its improved performance over a wide range of test instances.

	var. 1-10			var. 1-100		
	mean ofv			mean ofv		
n	function	best k	% ofv ch.	function	best k	% ofv ch.
25	1456	1460	-0,31	106896	107154	-0,24
50	5187	5230	-0,82	398953	401601	-0,66
100	19897	20422	-2,57	1426281	1464787	-2,63
250	119601	124866	-4,22	8994431	9470246	-5,02
500	470984	497043	-5,24	35676414	38045419	-6,23
1000	1880019	1992808	-5,66	140870384	151089892	-6,76

Table 6: Average weighted tardiness and relative difference (second set of instances)

	var. 1-10 function vs best k				var. 1-100 function vs best k			
n	<	=	>	sig.	<	=	>	sig.
25	209	209	82	0,000	196	221	83	0,000
50	240	165	95	0,000	233	161	106	0,000
100	299	129	72	0,000	294	123	83	0,000
250	303	120	77	0,000	311	119	70	0,000
500	309	120	71	0,000	312	120	68	0,000
1000	314	120	66	0,000	317	120	63	0,000

Table 7: Comparison of heuristic average weighted tardiness values and statistical test (second set of instances)

4 Conclusion

The ATC heuristic is one of the best dispatch rules available for the weighted tardiness scheduling problem, as several studies have shown. This heuristic uses a lookahead parameter whose value must be specified. Previously, this parameter had been set at a fixed value. We developed a function that maps some instance statistics into an appropriate value for the lookahead parameter. This approach was compared with some fixed values that have been recommended. The computational results showed that using the mapping function improves the performance of the heuristic over a wide range of instances.

References

[1] Abdul-Razaq, T. S., Potts, C. N., and Van Wassenhove, L. N. A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics 26* (1990), 235–253.

[2] Akturk, M. S., and Yildirim, M. B. A new lower bounding scheme for the total weighted tardiness problem. *Computers and Operations Research 25* (1998), 265–278.

[3] Hoogeveen, J. A., and Van de Velde, S. L. Stronger lagrangian bounds by use of slack variables: Applications to machine scheduling problems. *Mathematical Programming 70* (1995), 173–190.

[4] Lee, Y. H., Bhaskaran, K., and Pinedo, M. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions 29* (1997), 45–52.

[5] Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P. Complexity of machine scheduling problems. *Annals of Discrete Mathematics 1* (1977), 343–362.

[6] Potts, C. N., and Van Wassenhove, L. N. Single machine tardiness sequencing heuristics. *IIE Transactions 23* (1991), 346–354.

[7] Rachamadugu, R. V., and Morton, T. E. Myopic heuristics for the single machine weighted tardiness problem. Working Paper 28-81-82, Graduate School of Industrial Administration, Carnegie-Mellon University, 1981.

[8] Volgenant, A., and Teerhuis, E. Improved heuristics for the n-job single-machine weighted tardiness problem. *Computers and Operations Research 26* (1999), 35–44.