Acta Cybernetica **19** (2009) 125–133.

Parameter Learning Online Algorithm for Multiprocessor Scheduling with Rejection*

Tamás Németh† and Csanád Imreh†

Abstract

In multiprocessor scheduling with rejection the jobs are characterized by a processing time and a penalty and it is possible to reject the jobs. The goal is to minimize the makespan of the schedule for the accepted jobs plus the sum of the penalties of the rejected jobs. In this paper we present a new online algorithm for the problem. Our algorithm is a parameter learning extension of the total reject penalty algorithm. The efficiency of the algorithm is investigated by an experimental analysis.

Keywords: online algorithms, scheduling, experimental analysis

1 Introduction

In this paper we develop a new algorithm for the solution of the online scheduling with rejection problem on identical machines. The algorithm is based on the idea of learning the parameter of the Reject Total Penalty (RTP) algorithm. We measure the efficiency of the new algorithm by an experimental analysis.

The problem of scheduling with rejection is defined in [2]. In this model, it is possible to reject the jobs. The jobs are characterized by a processing time and a penalty. The goal is to minimize the makespan of the schedule for the accepted jobs plus the sum of the penalties of the rejected jobs. In the online case a 2.618-competitive algorithm is given for arbitrary number of machines. This algorithm is called Reject Total Penalty (RTP). One basic idea in scheduling with rejection is to compare the penalty and the load (processing time divided by the number of machines) of the job, and reject the job in the case when the penalty is smaller. This greedy algorithm can make a bad decision when the number of machines is large and this makes possible to appear large jobs with small loads. RTP handles these jobs more carefully. We give the detailed definition in the next section. In [2] a further, 1.618-competitive algorithm is presented in the case of 2 machines. Matching lower bounds are also given. In the offline case an FPTAS is presented for fixed number

*This research has been supported by the Hungarian National Foundation for Scientific Research, Grant F048587.

†Institute of Informatics, University of Szeged E-mail: {tnemeth,cimreh}@inf.u-szeged.hu

DOI: 10.14232/actacyb.19.1.2009.8

of machines, and a PTAS in the case where the number of machines is part of the input. The preemptive version of online scheduling with rejection is studied in [14], a generalized version of the reject total penalty algorithm is analyzed, and it is proved that this generalized algorithm is 2.387-competitive for arbitrary number of machines. A general lower bound of 2.124, and a lower bound of 2.33 for the class of obliviously scheduling algorithms (the accepted jobs are scheduled without knowledge of the rejection penalties) are also proved. In [7] the offline scheduling problem with rejection is investigated in some more complex machine models. In [8] an FPTAS is given for scheduling with rejection on related parallel machines. A further extension of the problem where the machines have cost and the algorithm has to purchase the machines before using them is investigated in [6] and [13]. A general machine scheduling problem on two sets of machines which generalizes the problem of scheduling with rejection is presented in [11].

Typically, the quality of an online algorithm is judged using competitive analysis. An online scheduling algorithm is C-competitive if the algorithm cost is never more than C times the optimal cost. On can find many details about competitive analysis in [4], [9] and [12]. Considering the competitive ratio the problem of scheduling with rejection on identical machines is completely solved in the general case, where no further restrictions are given on the jobs, algorithm RTP is an optimal online algorithm in the sense that it achieves the smallest possible competitive ratio. On the other hand in some cases the algorithm which has the best competitive ratio does not work well in average cases or on real data sets. In the area of online scheduling algorithms only a few papers present experimental studies, such papers are, for example, the next ones: In [1] the algorithms for online multiprocessor scheduling to minimize the makespan are investigated. In [3] a multicriteria version of the scheduling problem is studied. In [5] online scheduling with release dates to minimize the weighted total completion time is investigated.

The paper is organized as follows. In the next section we introduce the basic notations and recall the most important results which are used in the paper. In Section 3 we present the developed parameter learning algorithm. Section 4 contains the description of the experimental analysis, and the evaluation of the results. In Section 5 we summarize the results and list some further open problems related to the paper.

2 Notations and preliminaries

In the problem considered there are m identical machines. Each job j has a processing time denoted by p_j and a penalty denoted by w_j. For an arbitrary list J of jobs and an algorithm A, we denote by $A(J)$ the cost of the schedule produced by algorithm A on list J.

As a subroutine we will use an online scheduling algorithm. Several algorithms are developed for the online scheduling problem on n identical machines (see the survey [15]), we will use the classical, greedy online scheduling algorithm LIST ([10]). This algorithm always schedules greedily the arriving job on the least loaded

machine.

Considering the problem of multiprocessor scheduling with rejection, it is a straightforward idea to reject the jobs where the penalty is smaller than the load. The following greedy algorithm gives this solution for the problem.

Algorithm Greedy

If $w_j \leq p_j/m$ is valid for job j then reject it otherwise accept and schedule it by LIST.

Unfortunately Greedy makes bad decisions in some cases. If only one job arrives with a large processing time M and penalty $M/m + \varepsilon$, then Greedy accepts and schedules it, and this shows that it is not better than m competitive. Therefore its competitive ratio is not constant.

In [2] an algorithm called RTP is defined which achieves a constant competitive ratio. It is a refined version of Greedy, it also rejects some large jobs with $w_j > p_j/m$, these jobs are collected in set R. We can define this algorithm as follows.

Algorithm RTP(α)

- 1. *Initialization.* Let $R := \emptyset$.
- 2. When job j arrives
  - (i) If $w_j \leq \frac{p_j}{m}$, then reject.
  - (ii) Let $r = \sum_{i \in R} w_i + w_j$. If $r \leq \alpha \cdot p_j$, then reject job j, and set $R = R \cup \{j\}$.
  - (iii) Otherwise, accept j and schedule it by LIST

In [2] the following statement is proved about the competitive ratio of RTP.

Proposition 1. *RTP is* $(3+\sqrt{(}5))/2$ *competitive, with parameter* $\alpha = (\sqrt{(}5)-1)/2$

In the same paper it is proved that no algorithm with better competitive ratio exists.

Proposition 2. *There exists no online algorithm that is* β*-competitive for some constant* $\beta < (3 + \sqrt{5})/2$ *and for all* m.

3 Parameter learning algorithm

Proposition 1 and Proposition 2 solves the problem of multiprocessor scheduling with rejection on identical machines for the general case as far as the competitive analysis is concerned. $RTP((\sqrt{5}-1)/2)$ is the best possible online algorithm for the solution of the problem. On the other hand sometimes the algorithms which have better competitive ratio show worst performance in average case on real or randomly generated inputs. In the area of online scheduling such example can be found in [1]

where the online algorithms for multiprocessor scheduling to minimize makespan are analysed by an experimental analysis and it is shown that the simple LIST algorithm has better performance than some of the more complicated algorithm with smaller competitive ratio.

In the case of scheduling with rejection the parameter $\alpha = (\sqrt{5}-1)/2$ is the value which minimizes the competitive ratio, thus it is a natural question whether using some other parameter can improve the average case. In this section we present a new algorithm PAROLE (Parameter Online Learning) which tries to learn the best parameter during its execution. The algorithm works in phases, after each phase it chooses a new parameter based on the known part of the input. First we define a frame algorithm which uses the selection of the new parameter as a subrutin, then we define the subrutin which finds the new parameter. We defined the phases by the number of arriving jobs, the phase is finished after 250 jobs.

Algorithm PAROLE (PHASE i)

- At the beginning of phase i, use algorithm CHOOSE to find a new parameter α_i.
- Perform $RTP(\alpha_i)$ on the arrived part of the input. Change set R.
- Use $RTP(\alpha_i)$ for the jobs arriving during the phase.

It is a straightforward idea to use the value α_i where the cost $RTP(\alpha_i)(I)$ is minimal for the known part of input. Unfortunately it seems to be difficult to find the optimal value of the parameter. $RTP(\alpha)(I)$ is neither monotone nor continuous function of α. It is a step function, but it may have many pieces. Our conjecture is that it is an NP-hard problem to find the optimal value of α, but it seems to be difficult to prove that. Therefore we used the following sampling algorithm to find the new value of the parameter. The algorithm uses the previous value of the parameter denoted by α^*.

Algorithm CHOOSE

- Generate one element from the intervals $[(i-1)/10, i/10]$ by uniform distribution for $i = 1, \ldots, 10$. Denote this set by S_1. Consider the value α from S_1 where $RTP(\alpha)$ has the smallest cost on I. Denote it by $\bar{\alpha}$.
- Generate one element from the intervals $[\alpha^\star - i/100, \alpha^\star - (i-1)/100]$, $[\alpha^\star + (i-1)/100, \alpha^\star + i/100]$, $[\bar{\alpha} - i/100, \bar{\alpha} - (i-1)/100]$, $[\bar{\alpha} + (i-1)/100, \bar{\alpha} + i/100]$ for $i = 1, \ldots 10$ by uniform distribution. Denote the set of the generated elements by S_2. Return the value α from $S_2 \cup \{\alpha^*\} \cup \{\bar{\alpha}\}$ where $RTP(\alpha)$ has the smallest cost on I.

The basic idea of CHOOSE is to select a good parameter value. We investigate the neighborhoods of two candidates, one is the previous value of the parameter and a further one is a new value $\bar{\alpha}$ which is the best among several candidates selected independently on the previous value of the parameter.

4 Experimental analysis

4.1 Description of the performed tests

To investigate the performance of the new parameter learning algorithm we performed the following experimental study. We have implemented the algorithms presented above and analysed their behavior on randomly generated test cases and on real data.

During the analysis we investigated the following algorithms.

- the Greedy algorithm
- $RTP(\alpha)$ with $\alpha = (\sqrt{5}-1)/2$
- the parameter learning algorithm $PAROLE$.

To test the algorithms we considered the following classes of input. We used real data sets (Tests A and B) furthermore we used randomly generated inputs which were defined by the same distributions as in [1] and [3]. In each cases we used relatively large inputs in the tests, the reason is that for large inputs PAROLE has enough possibility to learn and use the best value of the parameter.

- Test A (real data): We collected the processing times of the tasks on the server www.szakoktatas.hu. The database contained around 100000 jobs. The average size of the jobs were 2597.1, the standard deviation was 20129.96. The smallest job had size 1, the largest had size 3134460. Therefore the jobs sizes followed the situation described in [1], there were very large jobs. On the server each process had a priority value which measured the importance of the task, this value is received by some properties of the jobs. The following properties were considered: the owner of the jobs (each user had a priority), the type of the job (the jobs are assigned to different classes by their type and each class has some priority value), some jobs had deadline (the deadline were also taken into account). We used a linear combination of these values to define the penalty. The average penalty was 272.83 the standard deviation was 229.32. The minimal penalty was 1, and the maximal was 2613.
- Test B (real data): We collected the processing times of the tasks on the server www.moravarosi.hu. The database contained around 200000 jobs and the jobs were larger than in the previous test. The average size of the jobs were 15608.96, the standard deviation was 120991.53. The smallest job had size 1, the largest had size 18839728. We used the priority to define the penalty in the same way as in Test A. We obtained larger penalties, the average penalty was 572.99, the standard deviation was 481.58. The minimal penalty was 1, the maximal one was 5487.
- Test C: In this case we generated the size of the jobs by exponential distribution. We used the parameter $\lambda = 1/1000$, therefore the expected value of the

size of the jobs were 1000, the variance was 1000000. We generated the penalties by exponential distribution as well, the used parameter was $\lambda = 1/100$, therefore the expected value of the penalty of the jobs were 100, the variance was 10000.

- Test D: In this case we generated the size of the jobs by hyperexponential distribution. We used two value $\lambda_1 = 1/800$, $p_1 = 1/2$, $\lambda_1 = 1/1200$, $p_2 = 1/2$. Therefore the expected value of the size of the jobs is $p_1/\lambda_1 + p_2/\lambda_2 = 1000$, the variance was 1080000. We generated the penalties by hyperexponential distribution as well, te used parameters were $\lambda_1 = 1/80$, $p_1 = 1/2$, $\lambda_1 = 1/120$, $p_2 = 1/2$, therefore the expected value of the penalty of the jobs was 100, the variance was 10800.

- Test E: In this case we generated the size of the jobs by Erlang-2 distribution. The used parameter was $\lambda = 500$, thus the expected size was 1000, the variance was 500000. We generated the penalties by Erlang-2 distribution as well, the used parameter was $\lambda = 1/50$, therefore the expected value of the penalty of the jobs was 100, the variance was 5000.

- Test F: In this case we generated the size of the jobs by bounded pareto distribution. We used the distribution $B(k, p, \alpha)$ where k (the bound on the smallest job size) was 1, p (the bound on the longest job size) was 20000, α is chosed to get the expected size 1000. We used the same distribution for penalties with upper bound 1000 and expected value 100.

In the case of the real data sets we used two subcases: the large input contained the full list of the jobs, in the small input we only used the first 20 percent of the jobs. We used 10 machines. The results are summarized in table 1. In the randomly generated tests we also investigated 2 subclasses. In the smaller size inputs 10000 jobs, in the larger size inputs one million jobs were generated and again we used 10 machines in the test. For every class we performed 100 randomly generated tests, the average results are summarized in table 2.

Table 1: The cost for real data

	A(Small)	A(Large)	B(Small)	B(Large)
Greedy	883549	4326745	2813479	13976483
RTP	843756	4117672	2889437	14245263
PAROLE	843679	4118935	2876559	14169854

We also investigated the number of rejected jobs. There was a big difference between test A and test B, in test A about 25 percent of the jobs was rejected and in test B around 42. This was expected since in test B the ratio of the expected values of the penalties and the jobs are smaller (in test A this ratio is 0,105 and in

Table 2: The average costs for randomly generated inputs

	C(Small)	C(Large)	D(Small)	D(Large)
Greedy	7487543	713456754	7964338	701341876
RTP	6822435	674523268	7657424	731735218
PAROLE	7014345	698764525	7776885	707823932
	E(Small)	E(Large)	F(Small)	F(Large)
Greedy	5698614	498238711	6718917	614492728
RTP	5254355	476521848	6396234	592718325
PAROLE	5317817	475632194	6512922	595598238

test B it is 0.037). In the randomly generated tests the ratio of the rejected jobs moved between 22 and 30 percent.

Considering the behavior of the algorithms we can observe that in each cases Greedy rejected the less jobs, and RTP rejected the most jobs. It is what we expected RTP and PAROLE reject each job which is rejected by Greedy.

If we consider the two type of costs then Greedy pays the less penalty and RTP the most. On the other hand only a small difference appeared in the amount of the paid penalty, it was less than 2 percent in all cases. In most tests the penalty is between 35 and 45 percent of the total cost, in the case of test B this ratio is larger, 60 percent.

4.2 Analysis of the results

Evaluating the results of the tests we can make the following observations:

- In most cases there is only small difference in the efficiency of the algorithms. The largest ratio of the costs occurred in test C(Small), where the ratio Greedy/RTP is 1.098. In most cases the ratio of the best and worst solution is below 1.05.

- *RTP* gave the best results in 6 test cases, PAROLE in 4 test cases and Greedy in 2 cases. On the other hand we note that PAROLE never resulted the worst result among the three algorithms. Thus we can conclude that PAROLE either gives the best result or its performance is between the performance of the other algorithms.

- The experimental results show that Greedy has better performance for the larger inputs. We think so that the reason of this property is that for large inputs Greedy can correct more easily the consequences of its bad decisions (when it accepts a large job, which actually should be rejected).

- We excepted that Greedy rejects the less jobs, and it has the less penalty cost and RTP pays the most penalty cost. Our tests confirmed this assumption.

It seems that PAROLE is more careful in rejecting jobs than RTP. On the other hand we can conclude that there are not big differences in the number of rejected jobs, in the test cases the three algorithms have similar behavior.

Summarizing the results of our experimental analysis we can conclude that the performance of PAROLE is usually between the performances of the other algorithms, therefore it can be useful in online computation where we have no information about the input.

5 Conclusions and further questions

In this paper we presented a new algorithm for the solution of the online scheduling with rejection problem. The efficiency of the algorithm is studied by an experimental analysis. The analysis shows that the algorithm gives good results on randomly generated inputs and on real data. Considering the algorithm defined in this paper some further open questions arise, we list them below.

The most important question is to characterize the complexity of finding the optimal value of the parameter of RTP. We conjecture so that this problem is NP-hard. It would be interesting to find better algorithms than CHOOSE to generate the next value of the parameter, we think so that such algorithm might improve significantly the efficiency of the algorithm. Finally we note that there exist further online problems where the best algorithm belongs to a class of algorithms and it is defined by fixing a parameter value which achieves the best competitive ratio. It is also an interesting question whether the idea of parameter learning is useful for such problems.

References

[1] Albers, S. and Schröder, B. An Experimental Study of Online Scheduling Algorithms. *ACM Journal of Experimental Algorithms*, article 3, 2002.

[2] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J. and Stougie, L. Multiprocessor scheduling with rejection, *SIAM Journal on Discrete Mathematics*, 13:64–78, 2000.

[3] Bilo, V., Flammini, M. and Giovannelli, R. Experimental analysis of online algorithms for the bicriteria scheduling problem, *J. Parallel Distrib. Comput.*, 64:1086–1100, 2004.

[4] Borodin, A. and El-Yaniv, R. *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.

[5] Correa, J.R., and Wagner, M. LP-Based Online Scheduling: From Single to Parallel Machines *Mathematical Programming*, to appear (prelinimary version in Proceedings of IPCO 2005 LNCS 3509, 186–209).

[6] Dósa, Gy. and He, Y. Scheduling with machine cost and rejection, *Journal of Combinatorial Optimization*, 12(4):337–350, 2006.

[7] Engels, D.V, Karger, D.R., Kolliopoulos, S.G., Sengupta, S., Uma, R.N. and Wein, J. Techniques for scheduling with rejection, *Journal of Algorithms*, 49:175–191, 2003.

[8] Epstein, L. and Sgall, L. Approximation schemes for scheduling on uniformly related and identical parallel machines, *Algorithmica*, 39(1):43–57, 2004.

[9] Fiat, A. and Woeginger, G.J., editors, *Online algorithms: The State of the Art, LNCS 1442*, Springer-Verlag Berlin, 1998

[10] Graham, R.L. Bounds for certain multiprocessor anomalies, *Bell System Technical Journal*, 45:1563–1581, 1966.

[11] Imreh, Cs. Scheduling problems on two sets of identical machines, *Computing*, 70:277–294, 2003.

[12] Imreh:, Cs. Competitive analysis, In Iványi, J., editor, *Algorithms of Informatics Volume 1*, pages 395–428, Budapest 2007, mondAt.

[13] Nagy-György, J. and Imreh, Cs. On-line scheduling with machine cost and rejection, *Discrete Applied Mathematics*, 155: 2546–2554, 2007.

[14] Seiden, S.S. Preemptive Multiprocessor Scheduling with Rejection, *Theoretical Computer Science*, 262:437–458, 2001.

[15] Sgall, J. On-line scheduling, In Fiat, A. and Woeginger, G.J., editors *Online algorithms: The State of the Art, LNCS 1442*, pages 196-231, Berlin, 1998, Springer Verlag.