AN ONLINE ALGORITHM FOR MINIMZATION OF MAKESPAN IN SCHEDULING PROBLEMS WITH UNIFORM PROCESSORS

R. CHANDRASEKARAN AND P.R. NARAYANAN
THE UNIVERSITY OF TEXAS AT DALLAS

Abstract. In this paper, we consider on-line algorithms for the problem of minimization of makespan on parallel uniform processors. An on-line algorithm for the problem when the number of machines m equals 2 is presented and its performance is analyzed. As we would expect, the worst case performance ratio depends on the ratio of the speeds of the two processors. We prove that this algorithm is optimal.

Introduction:

In this paper, we consider on-line algorithms for the problem of minimization of makespan on parallel uniform processors. An on-line algorithm for the problem when the number, m, of machines equals 2 is presented and its performance is analyzed. As we would expect, the worst case performance ratio depends on the ratio of the speeds of the two processors. We prove that this algorithm is optimal. This extends a similar result for the identical machine case.

In an uniform processor system, the processors operate at different speeds. Processor i whose speed is s_i takes $\frac{t_j}{s_i}$ units to complete processing a task j with a processing time requirement of t_j. Without loss of generality we can scale the speeds of the processors up or down by a constant, so that the speed of the fastest processor is 1. Our problem deals with the uniform parallel processor system with $m = 2$; so we assume the speed of the slower processor to be $p < 1$.

The problem of minimization of makespan on m parallel uniform processors is the following problem: Given a set of tasks $1 \leq i \leq n$ we wish to minimize $\max_i C_i$, where C_i denotes the completion time of task i. The off-line version of the problem is NP — hard. Polynomial approximation schemes are available using the similar scheme for the problem of dual bin packing [3].

As far as we know the on-line version of the problem is open. In the unpublished extended abstract [5] it is shown that if there is an on-line algorithm for the problem when all tasks are available at time zero, with a worst case performance ratio less than or equal to γ, then it can be extended to the case where tasks arrive over a period of time to give an on-line algorithm with a worst case performance ratio less than or equal to 2γ.

The Model:

The model for the problem is similar to the one described in [1], [2],. There are two competing players, the *scheduler* and the *taskmaster*. The *taskmaster* supplies the task and the *scheduler* assigns the task to a processor using an algorithm. The number of tasks and the processing time of the next task can be arbitrarily fixed by the *taskmaster*.

In [2], the *taskmaster* did not have to exploit the power of deciding on the processing time of the next task, with a knowledge of the previous assignment of tasks by the *scheduler*. The example which forced a lower bound on the performance of any on-line algorithm for $m \geq 4$, and the example that proved the optimality of Algorithm I for

$m = 2, 3$ (theorem 7) (see [4]), were sequences of tasks, where the processing time of the next task did not depend on the previous assignment of tasks to processors. In spite of this, the *scheduler* could do no better. In this paper, however, we will prove lower bounds on the performance ratio of some algorithms using examples where the processing time of the next task in the sequence depends on the assignment of the previous tasks to processors.

Algorithms:

The following notation is used in the description and analysis of the algorithm for the problem. Let L_i, $i = 1, 2$ denote the completion time of the last task on processor I and processor II respectively. Here processor I is the slower processor. Let C_{opt} denote the optimal makespan using an off-line algorithm. We assume without loss of generality, as stated before, that the speed of the processor I is $p < 1$ and the speed of the processor II is 1. In the next few sections, we will discuss the performance ratio of an algorithm for the problem. We prove that the worst case performance ratio R of this algorithm is given by :

$$R = \begin{cases} 1 + \frac{1}{p} & p \geq 1 + \frac{1}{p} \\ 1 + \frac{p}{1+p} & p < 1 + \frac{1}{p} \end{cases}$$

Please note that $p > 1 + \frac{1}{p} \iff p > \frac{\sqrt{5}+1}{2}$. We prove that this algorithm is optimal for the problem in either case.

Algorithm I':

When a new task is supplied, assign the task to a processor which will complete it the earliest.

procedure Algorithm I':
begin
accept new task x
begin
if $L_2 + x < L_1 + \frac{x}{p}$ then
set $L_2 \leftarrow L_2 + x$; /* assign x to processor II */
else
set $L_1 \leftarrow L_1 + \frac{x}{p}$; /* assign x to processor I */
end (Algorithm I').

THEOREM 0.1. *The algorithm is optimal.*

Proof. : Assume, by induction, that the performance ratio of Algorithm I' is less than or equal to R given above for a task system with n tasks. Clearly, the induction statement is true for the first task. Let us analyze the performance ratio of the algorithm at the instance a new task is assigned to a processor. Let x be the processing time of the $(n+1)^{st}$ task. Let the loads on the two machines be denoted by L_1 and pL_2.

$$R \leq \frac{\min[\max(L_1 + \frac{x}{p}, L_2), \max(L_1, L_2 + x)]}{\max(\frac{pL_1 + L_2 + x}{p+1}, x)}$$

The numerator is the on-line schedule length and the denominator is a lower bound on the off-line schedule length. If the numerator is less than or equal to $\max(L_1, L_2)$ then we are done by induction. This occurs if $[0 \leq \frac{x}{p} \leq L_2 - L_1]$ or $[0 \leq x \leq L_1 - L_2]$. Hence we only consider the other cases.

Case (i): $L_1 \geq L_2; x > L_1 - L_2$:

In this case, the online schedule length equals $\min[L_1 + \frac{x}{p}, L_2 + x]$. The two quantities are equal when $x = \frac{p}{1-p}(L_2 - L_1)$. The lower bound on offline schedule length is $\max[\frac{pL_1 + L_2 + x}{p+1}, x]$. The two quantities here are equal when $x = \frac{pL_1 + L_2}{p}$. A comaprison of these two values for x is given below:

$$\frac{p}{1-p}(L_2 - L_1) \leq \frac{pL_1 + L_2}{p} \iff 0 \leq pL_1 - (p^2 + p - 1)L_2$$

Case (i.1)$0 \leq pL_1 - (p^2 + p - 1)L_2$: Incidentally, this is the only possible case if $(p^2 + p - 1) < 0$. This condition is equivalent to the condition $\frac{1}{p} > p + 1$. There are three subcases possible depending on the value of x relative to its changeover points:

Case (i.1.1):$L_1 - L_2 \leq x \leq \frac{p}{1-p}(L_2 - L_1)$: In this case

$$R \leq \frac{L_1 + \frac{x}{p}}{\frac{pL_1 + L_2 + x}{p+1}} = (p+1)\frac{pL_1 + x}{p^2 L_1 + pL_2 + x}$$

This quantity on the right is a ratio of linear functions in x and hence achieves its maximum at an extreme value of x as shown below:

$$R \leq \begin{cases} 1 + \frac{2p(L_1 - L_2)}{(p^2+1)L_1 + (p-1)L_2} & x = L_1 - L_2 \\ 1 + \frac{1}{p + \frac{(p^2-1)L_1}{p(L_2 - L_1)}} & x = \frac{p}{1-p}(L_2 - L_1) \end{cases}$$

The second term is the larger of the two and hence it is the maximum.

Case (i.1.2):$\frac{p}{p-1}(L_2 - L_1) \leq x \leq p(L_1 + pL_2)$: In this case,

$$R \leq \frac{L_2 + \frac{x}{p}}{\frac{L_1 + pL_2 + x}{p+1}} = (p+1)\frac{L_2 + \frac{x}{p}}{L_1 + pL_2 + x}$$

Again by the same arguement,

$$R \leq \begin{cases} 1 + \frac{1}{p + \frac{(p^2-1)L_1}{p(L_2 - L_1)}} & x = \frac{p}{p-1}(L_2 - L_1) \\ 1 + \frac{1}{p + \frac{L_1}{L_2}} & x = p(L_1 + pL_2) \end{cases}$$

Case (i.1.3): $x > p(L_1 + pL_2)$: In this case, $R \leq \frac{L_2 + \frac{x}{p}}{\frac{x}{p}} = 1 + \frac{pL_2}{x}$. This quantity achieves maximum in this interval when $x = p(L_1 + pL_2)$ at this point $R \leq 1 + \frac{1}{p + \frac{L_1}{L_2}}$.

In case (i.1) $\max[1+\frac{1}{p+\frac{L_1}{L_2}}, 1+\frac{1}{p+\frac{(p^2-1)L_1}{p(L_2-L_1)}}, 1] = 1+\frac{1}{p+\frac{L_1}{L_2}}$. When $p > 1+\frac{1}{p}$, (i.1)$\Longrightarrow$ $0 \le L_1 \le L_2$ is the range for L_1. In this range maximum of $1+\frac{1}{p+\frac{L_1}{L_2}} = 1+\frac{1}{p}$ and this value occurs at $L_1 = 0$, and $x = p(L_1 + pL_2)$. Hence when $p > 1+\frac{1}{p}$, in all these cases we have $R \le 1+\frac{1}{p}$.

Now suppose $1 \le p \le 1+\frac{1}{p} \Longrightarrow p^2 - p - 1 < 0$: Case (i.1) requires that $L_1 \ge \frac{1+p-p^2}{p} L_2$. The maximum value of R is achieved when this inequality is an equation and this value of $R = 1+\frac{p}{1+p}$.

Case (i.2)$pL_1 + (p^2 - p - 1)L_2 \le 0$

In this case, we must have $p < 1+\frac{1}{p}$, which implies that $p^2 - p - 1 \le 0$ and hence $L_1 \le \frac{1+p-p^2}{p} L_2$. Moreover, $p(L_1 + pL_2) \le \frac{p}{p-1}(L_2 - L_1)$.

$$R \le \begin{cases} 1 & x = L_2 - L_1 \\ p + \frac{1}{1+p\frac{L_2}{L_1}} & x = p(L_1 + pL_2) \end{cases}$$

$$R \le \begin{cases} p + \frac{1}{1+p\frac{L_2}{L_1}} & x = p(pL_2 + L_1) \\ p + (p-1)\frac{L_1}{L_2 - L_1} & x = \frac{p}{p-1}(L_2 - L_1) \end{cases}$$

In this range, maximum value of the upper bound for R equals $p + \frac{1}{1+p\frac{L_2}{L_1}}$. Since the lower bound on the ratio $\frac{L_2}{L_1}$ is $\frac{p}{1+p-p^2}$, we get a maximum value for the upper bound for R equals $1+\frac{p}{1+p}$.

Thus, the maximum value of R for this algorithm when $L_1 \le L_2$ is given by:

$$R \le \begin{cases} 1+\frac{1}{p} & p \ge 1+\frac{1}{p} \\ 1+\frac{p}{p+1} & p < 1+\frac{1}{p} \end{cases}$$

Case (ii): $L_1 \ge L_2; x > p(L_1 - L_2)$

online schedule length: $L_2 + \frac{x}{p}$

offline schedule length: $\max[\frac{L_1+pL_2+x}{p+1}, \frac{x}{p}]$

$$R \le \begin{cases} 1 & x = p(L_1 - L_2) \\ 1+\frac{1}{p+\frac{L_1}{L_2}} & x = p(L_1 + pL_2) \end{cases}$$

The maximum of these is the second term in the above expression. In this range of values for L_1 and L_2 the maximum possible value of R is $1+\frac{1}{1+p}$. This is less than the corresponding expressions for R obtained in case (i).

Thus, the maximum value of R for this algorithm is given by:

$$R \le \begin{cases} 1+\frac{1}{p} & p \ge 1+\frac{1}{p} \\ 1+\frac{p}{p+1} & p < 1+\frac{1}{p} \end{cases}$$

We now show that these are lower bounds for any algorithm thereby proving that this algorithm is optimal.

EXAMPLE 1. *When $p > 1 + \frac{1}{p}$ consider the following example:*

First we give a job whose length is 1. *If this is put on machine I, we quit since the ratio of the length of the schedule to the minimum offline length is p. Hence this job must go on machine II. Now we give a job of length p. If this is put on machine I, again we get a ratio of p and we are done. Hence this job must also go on machine II. Now we give a job of length $p(p+1)$. If this job goes on machine I, the ratio is again p. On the other hand, if it goes on machine II, the ratio is $1 + \frac{1}{p}$ and we are again done. Hence $1 + \frac{1}{p}$ is a lower bound for this case for any algorithm.*

EXAMPLE 2. *When $p < 1 + \frac{1}{p}$, $p^2 - p - 1 < 0$. First we give a job of length* 1. *the next job depends on what is done for the first job.*

Case (i): First job is put on machine I. In this case we give a job of length $\frac{p^2}{1+p-p^2}$. If this also goes on machine I, the ratio is $1 + \frac{1}{p} > 1 + \frac{p}{p+1}$ and we are done. Hence this must go on machine II.

Case (ii) First job goes on machine II. In this case we give a job of length $\frac{1+p-p^2}{p^2}$. If this is also put on machine II, we get a ratio of $1 + p > 1 + \frac{p}{1+p}$. Hence, this must go on machine I.

In either case we get $\frac{L_1}{L_2} = \frac{1+p-p^2}{p}$. Now we give a job of length $x = p(L_1 + pL_2)$ and this forces a ratio of $1 + \frac{p}{p+1}$ whether this last job goes machine I or II.

This proves that this algorithm is optimal.

REFERENCES

[1] R. Chandrasekaran and P.R. Narayanan: *A Note on On-line Algorithms with a Performance Ratio less than $2 - \frac{1}{m}$*, Submitted for publication.

[2] Gabor Galambos and Gerhard Woeginger: *An On-line Scheduling Heuristic with Better Worst Case Ratio than Graham's List Scheduling*, SIAM J. on Computing, 22, #2, pp. 349-355.

[3] Dorit S. Hochbaum and David B. Shmoys: *A Polynomial Approximation Scheme for Scheduling on Uniform Processors Using Dual Approximation Approach*, SIAM J. on Computing, 17, #3, June 1988, pp. 539-551.

[4] P.R. Narayanan: *Performance Analysis of On-line Algorithms Under Various Scheduling Criteria*, Ph. D Dissertation, University of Texas at Dallas, August 1992.

[5] David B. Shmoys, Joel Wein, and David P. Williamson: *Scheduling on Parallel Machines On-line*, Extended Abstract, MIT, April 18, 1991.