BULETINUL ACADEMIEI DE ŞTIINŢE
A REPUBLICII MOLDOVA. MATEMATICA
Number 3(79), 2015, Pages 102–109
ISSN 1024–7696

A Parametric Scheme for Online Uniform-Machine Scheduling to Minimize the Makespan

Alexandre Dolgui, Vladimir Kotov, Alain Quilliot

Abstract. In this paper, we consider the Online Uniform Machine Scheduling problem in the case when speed $s_i = 1$ for $i = n-k+1, \dots, n$ and $S_i = s, 1 \le s \le 2$ for $i = 1, \dots, k$, where k is a constant, and we propose a parametric scheme with an asymptotic worst-case behavior (when m tends to infinity).

Mathematics subject classification: 34C05.
Keywords and phrases: Online Scheduling, Uniform Parallel Machine, worst-case behavior, parametric scheme.

1 Introduction

In this paper, we study the classic problem of scheduling jobs *online* on m uniform machines $(M_1, M_2, \dots M_m)$ with speeds $(s_1, s_2, \dots, s_m)$ without preemption: jobs arrive one at a time, according to a linear ordering (a list) σ, with known processing times and must immediately be scheduled on one of the machines, without knowledge of what jobs will come afterwards, or how many jobs are still to come; all machines can perform the same tasks, according to distinct speeds. However, the way jobs are ordered inside the list σ has no correlation with the starting times which are assigned to them in the schedule: some future (in the list σ) job may come to start earlier than the current one, because what we do here is only distributing the jobs among the machines.

We denote by J_j the jth job in the list s, and say that job J_j arrives at *step* j according to s. We denote by p_j the processing time of job J_j. If job p_j is assigned to machine M_i, then p_j/s_i time units are required to process this job.

The quality of an online algorithm A is measured by its competitive ratio, defined as the smallest number c such that, for every list of jobs σ which describes jobs together with their arrival order, we have $F(A, \sigma) \le c \cdot Opt(\sigma)$, where $F(A, \sigma)$ denotes the makespan of the schedule which derives from application of algorithm A to the list σ, and $Opt(\sigma)$ denotes the makespan of some optimal schedule of the jobs of σ, computed while considering σ as a set of jobs, and not as an ordering. We may also say that $Opt(\sigma)$ is the optimal value of the *offline* scheduling problem induced by the jobs contained in the list σ. The algorithm A is said to be c-competitive.

The online *Multi-machine Scheduling* problem for identical machines (they are all provided with the same speed) was first investigated by Graham, who showed that the *List* algorithm (LS) which always puts the next job on the least loaded machine is exactly $(2 - 1/m)$-competitive [2].

In the case of uniform machines Cho and Sahni [1] proved that the LS algorithm has a worst-case bound of $(3m-1)/(m+1)$ for $m \geq 3$. When $s_i = 1, i = 1, \ldots, m-1$ and $s_m > 1$, Cho and Sahni also showed that the LS algorithm has a worst-case bound c of $1 + (m-1) \cdot (\min(2, s)/(m + s - 1)) \leq 3 - 4/(m+1)$, and the bound $3 - 4/(m+1)$ is achieved when $s = 2$. Li and Shi [3] proved that the LS algorithm is the best possible one for $m \leq 3$, and proposed an algorithm that is significantly better than the LS algorithm when $s_i = 1, i = 1, \ldots, m-1$ and $s_m = 2, m \geq 4$. The algorithm has a worst-case bound of 2.8795 for a big m. For the same problem Cheng, Ng and Kotov [4] proposed a 2.45-competitive algorithm for any $m \geq 4$ and any $s_m, 1 \leq s_m = s \leq 2$. Also, some results in the case of fixed number of machines can be found in [5–7]. It should be mentioned that the worst-case behavior of all previous algorithms occurs when m tends to infinity.

In this paper we use ideas of reserved classes and a dynamic lower bound of the optimal solution from [8, 9].

2 A Parametric Scheme for the OnLine Uniform Machine Scheduling Problem

Before presenting the main results, we introduce some notations.

1. m denotes the total number of machines;

2. k denotes the number of machines with a speed $1 < s \leq 2$, k is a constant.

We are going to describe here a strategy (an algorithm) which will allow us to assign for any index j the job J_j with processing time p_j which arrives at step j ($j = 1, \ldots, Length(s)$) according to the list ordering σ to some machine $M_i, i = 1, \ldots, m$. We shall do in such a way that J_j will then be scheduled immediately after the end of the latest job which was assigned to M_i. As a matter of fact, since no precedence relation is imposed to the jobs, jobs assigned to a same machine will be consecutively run, without any idle time. So, any time we have to deal with a current job J_j of the input list σ, we denote by:

1. $L_{i,j}$ the current load of machine i before assigning job J_j;

2. $L^*_{i,j}$ the current load of machine i after assigning job J_j;

3. V_j the theoretical optimal makespan for the *offline* scheduling problem induced by the job set $J(j) = \{J_1, \ldots, J_j\}$ made of the jobs which arrived no later than step j.

It is easy to check that, if we denote by $q_1, .., , q_j$ the processing time of the jobs of $J(j)$, sorted by decreasing order, which means that we have: $q1 \geq q2 \geq \cdots \geq q_j$, then we may state:

Lemma 1. *The following inequalities hold:*

1. $V_j \geq (q_1 + q_2 + \cdots + q_j)/(m - k + s \cdot k)$;
2. $V_j \geq q_1/s$;
3. $V_j \geq \min\{(q_k + q_{k+1})/s, q_{k+1}\}$.

Proof. Left to the reader. It is important to notice that the last inequality $V_j \geq \min\{(q_k + q_{k+1})/s, q_{k+1}\}$ derives from the hypothesis $1 \leq s \leq 2$. As a matter of fact, it will be the only place, inside our reasoning process, where the hypothesis plays a role.

So, for any step value j, we set:

$$LB_j = \max\{(q_1 + q_2 + \cdots + q_j)/(m - k + s \cdot k), q_1/s, \min\{(q_k + q_{k+1})/s, q_{k+1}\}\}. \quad (1)$$

Clearly, LB_j is a lower bound for the optimal *offline* makespan related to step j and we have: $LB_{j-1} \leq LB_j$ (LB_j is monotonic). □

2.1 The Assignment Process Assign

We suppose now that some positive number α is given together with three integral numbers R, m_1 and m_2 in such a way that:

$$(1 + \alpha) \cdot s \cdot k + (1 + \alpha/2) \cdot m_1 \geq s \cdot k + m_1 + m_2, \quad (2)$$

$$k + m_1 + m_2 = m, \quad (3)$$

$$m_2 = R \cdot k, \quad (4)$$

$$R \geq \log_{1+\alpha/2}((1 + \alpha/2)/(2 + \alpha - s)). \quad (5)$$

It is easy to see that, if we fix k, s and α, and if we require R and m_1 to take the smallest possible values, then R, m, m_1 and m_2 are completely determined by k, s and α.

This assumption about the way the machine number m may be decomposed, allows us to split the machine set machines into three classes:

1. machines with speed s are called *Fast*;
2. we pick up m_1 machines among the $m - k$ machines with speed 1 and call them *Normal*;
3. the m_2 remaining machines with speed 1 are called *Reserved* and the $m_2 = R \cdot k$ *Reserved* machines are split into R groups $G_0, \ldots, G_{R-1}$, each group containing exactly k machines.

By the same way, we say that job J_j, which arrives at step j is:

1. *Small* if its processing time p_j is at most equal to $(1+\alpha/2)\cdot LB_j$;
2. *Large* else.

Finally, we say that this job J_j fits machine M_i, $i=1\dots m$, if $L_{i,j}+p_j/s_i \leq (2+\alpha)\cdot LB_j$.

We easily see that:

Lemma 2. *If Large job J_j does not fit machine i from class Fast then $L_{i,j} > (1+\alpha)\cdot LB_j$.*

Proof. It comes in a straightforward way from the fact that $p_j/s_i = p_j/s \leq LB_j$. □

Doing this allows us to describe our online algorithm *Assign*, which will work on any instance of the *Online Uniform Machine Scheduling* Problem such that m, k, s may be written according to the relations (2)-(5). The main idea here is that at any step j, we are going to be able to assign job J_j to some machine $i(j)$ in such a way that we keep the following inequality: $\max_i L^*_{i,j} \leq (2+\alpha)\cdot LB_j$. While doing this will happen to be easy in the case when j is a *Small* job, the trick will be to show that, if j is a *Large* job, we may, by conveniently switching machines inside the *Normal* and *Reserved* classes, do in such a way that if j does not fit any of machine of classes *Fast* and *Normal*, then it fits at least some machine in current group G_0, whose machines are, at any time during the process, provided with current labels in $\{1,...,k\}$. It is important to understand here that the status *Normal* or *Reserved* of a given machine with speed 1 is not going to be fixed, and will be evolving all throughout the process.

Algorithm Assign

Initialization: Set $n=1$; (*n denotes the index of the current target *Reserved* machine in group G_0; machines in every group G_R are indexed from 0 to $k-1$*); Set $j=0$; $LB_j=0$;

Read(σ);

While σ is non empty do

$j := j+1$;

Read the current job J_j and perform Step j as follows:

Update LB_j according to formula (1).

If job J_j fits some machine i in classes *Fast* and *Normal*

    then (I1)

        assign j to this machine i

    Else

        If $n<k$ then (I2)

            Assign job J_j on the machine (with label) n in G_0;

            Let i_0 be the machine from class *Normal* with minimal current load. Switch machines n and i_0 between groups *Normal* and G_0 in such a way that machine i_0 comes in G_0 with label n,

and machine n is put into class *Normal*. Set $n = n + 1$;
If $n = k$ then (I3)
Update the labeling of groups $G_0, \ldots, G_{R-1}$ in such a way that group $r, 1 \leq r \leq R - 1$, becomes group $r - 1$, and group 0 becomes group $R - 1$. Set $n = 1$.

2.2 Worst Case Performance of Assign

The *Assign* algorithm works on an instance $(M_1, M_2, \ldots, M_m; s_1, s_2, \ldots, s_m)$ of the *Online Uniform Machine Scheduling* Problem which is such that:

1. $s_i = s \in [1, 2]$ for $i = 1, ..., k$; $s_i = 1$ for $i = k + 1, ..., m$;

2. m may be decomposed as a sum $m = k + m_1 + m_2 = k + m_1 + k \cdot R$ with m_1, m_2, R as in (2)-(5).

We are now going to show that, if k, α is fixed and if m is large enough, then the competitive ratio of *Assign* is no more than $(2+\alpha)$.More specifically, we are going to prove that, if a job list s is some input for *Assign*, then the makespan $F(Assign, \sigma)$ of the schedule which is computed by *Assign* does not exceed $(2+a) \cdot LB(\sigma)$, where $LB(\sigma)$ denotes the lower bound for $Opt(\sigma)$ which may be derived from the list s according to Lemma 1.

Lemma 3. *At every step j during the execution of the Assign algorithm there exists either a machine i in class Fast such that $L_{i,j} \leq (1 + \alpha) \cdot LB_j$ or a machine i from class Normal such that $L_{i,j} \leq (1 + \alpha/2) \cdot LB_j$.*

Proof. Let us suppose the converse, which means that, at some step j, we have, for any *Fast* machine i: $L_{i,j} > (1 + \alpha) \cdot LB_j$, and for any *Normal* machine i : $L_{i,j} > (1 + \alpha/2) \cdot LB_j$. It means that $p_1 + p_2 + \cdots + p_j = s \cdot \sum_{i \in Fast} L_{i,j} + \sum_{i \in \text{Normal } \cup \text{Reserved}} L_{i,j} > k \cdot s \cdot (1 + \alpha) \cdot LB_j + m_1 \cdot (1 + \alpha/2) \cdot LB_j$. But Lemma 1 tells us that $p_1 + p_2 + \cdots + p_j s \leq s \cdot k + m_1 + m_2) \cdot LB_j$, while relation 2 tells us that $k \cdot s \cdot (1 + \alpha) + m_1 \cdot (1 + \alpha/2) \geq (s \cdot k + m_1 + m_2)$. We deduce a contradiction and conclude. □

We deduce:

Lemma 4. *If current job J_j is a Small job then there is a machine from class Fast or Normal such that job J_j fits with it.*

Proof. Let us apply above Lemma 2 and consider a machine i as in the statement of Lemma 2. If i is Fast, then $L_{i,j} \leq (1 + \alpha) \cdot LB_j$. We deduce from the fact that $p_j/s_i = p_j/s \leq LB_j$ that $L_{i,j} + p_j/s_i \leq (2 + \alpha) \cdot LB_j$ and the result. If i is Normal, then $L_{i,j} \leq (1 + \alpha/2) \cdot LB_j$, and $L_{i,j} + p_j/s_i = L_{i,j} + p_j \leq (2 + \alpha) \cdot LB_j$. We conclude. □

Given some input job list σ: let us denote by $j(1), ..., j(Q)$ the steps when process *Assign* performs instructions (I2) or (I3) while running σ. Clearly, those instructions are performed according to some kind of cyclic scheme, and every index $q = 1, ..., Q$ may be written as $q = h + t \cdot k + T \cdot k \cdot R$, where $h \in \{0, ..., k-1\}$ and $t \in \{0, ..., R-1\}$, $T \geq 0$, with the following meaning: when performing (I2) or (I3), *Assign* deals with the job group which was originally group G_t, and, inside this group, deals with machine with label h.

For every $q = 1, ..., Q$, we denote by $i(q)$ the related target machine, which is, at this time, a *Reserved* machine located in current group G_0, with index h.

We may notice that:

– instruction (I3) occurs every time t is incremented: $t \to t+1$;

– original group G_0 takes again label 0 every time T is incremented: $T \to T+1$.

We claim:

Lemma 5. *For $q = 1, ..., Q$, we have $L_{i(q),j(q)} \leq (2 + \alpha - s) \cdot LB_{j(q)}$. (*)*

Proof. Let us consider $q = h + t \cdot k + T \cdot k \cdot R$, and try to prove above inequality (*). Obviously, (*) is true in case $T = 0$, since all machines from class *Reserved* are empty. So we may suppose $T \geq 1$. After assigning a *Large* current job $J_{j(q-k \cdot R)}$ to the machine $j(q - k \cdot R) = h$ in current group G_0, we switch machine $j(q - k \cdot R)$ with some *Normal* machine i_0 according to instruction (I2). Since we could not assign job $J_{j(q-k \cdot R)}$ neither to a *Fast* nor to a *Normal* machine, Lemma 3 tells us that there is a machine i in class *Normal* such that: $L_{i,j(q-k \cdot R)} \leq (1 + \alpha/2) \cdot LB_{j(q-k \cdot R)}$. So, this inequality also holds for the machine i_0 which becomes machine h in group G_0. We deduce that the load, after instruction (I2) has been performed, of machine h in group G_0 is bounded by $(1 + \alpha/2) \cdot LB_{j(q-k \cdot R)}$. This machine is going to keep with the same load until we arrive to step $q = h + t \cdot k + T \cdot k \cdot R$ and at this time this machine corresponds to machine $i(q)$. So we may state:

$$L_{i(q),j(q)} \leq (1 + \alpha/2) \cdot LB_{j(q-k \cdot R)}. \tag{6}$$

On the one hand, we see that, for any value $q \geq k+1$, we have been provided with *Large* (at the time when they arrived) $k+1$ jobs $J_{j(q)}, ..., J_{j(q-k)}$, all with processing times respectively larger than $(1 + \alpha/2) \cdot LB_{j(q)}, ..., (1 + \alpha/2) \cdot LB_{j(q-k)}$, which means, because of the monotonicity of *LBj*, all with processing times larger than $(1+\alpha/2) \cdot LB_{j(q-k)}$. It comes from the relation $LB_j \geq \min\{(q_k + q_{k+1})/s, q_{k+1}\}$ of Lemma 1, that $(1 + \alpha/2) \cdot LB_{j(q-k)} < LB_{j(q)}$. We may propagate this relation and get:

$$(1 + \alpha/2)^R \cdot LB_{j(q-R \cdot k)} \leq LB_{j(q)}. \tag{7}$$

Combining (6) and (7) yields: $L_{i(q),j(q)} \leq (1 + \alpha/2)^{1-R} \cdot LB_{j(q-R \cdot k)}$. We deduce (*) if $(1 + \alpha/2)^{1-R} \leq 2 + \alpha - s$, that means if $R \geq \log_{1+\alpha/2}((1 + \alpha/2)/(2 + \alpha - s))$. We conclude since this last inequality is part of our hypothesis (equation (5)). $\square$

Theorem 1. *Let us suppose that σ is given and that our Online Uniform Machine Scheduling instance is such that m, k, s may be written according to the relations*

(2)-(5). Then, for any input job list s, the Assign algorithm works in such a way that: $F(Assign, \sigma) \leq (2+\alpha) \cdot Opt(\sigma)$. *That means that its competitive ratio does not exceed* $(2+\sigma)$ *in the case of such instance.*

Proof. Lemma 4 tells us that, if, at any step j, current job J_j is *Small*, then it is possible to assign it to some machine in *Normal* $\cup$ *Fast* in such a way that the resulting makespan does not exceed $(2+\alpha) \cdot LB_j$. By the same way, if J_j is *Large* and fits with some *Fast* machine, then it is possible, according to the mere definition of fitness, to assign it to this machine in such a way that the resulting makespan does not exceed $(2+\alpha) \cdot LB_j$. Finally, Lemma 2 and 5 tell us that if if J_j is *Large* and cannot be assigned to some *Fast* machine, then *Reserved* machine i with label n in group G_0 is such that $L_{i,j} \leq (2+\alpha-s) \cdot LB_j$. Since Algorithm *Assign* assigns job J_j to machine i, we see that the resulting load $L^*_{i,j}$ does not exceed $(2+\alpha-s) \cdot LB_j + p_j$. Since Lemma 1 tells us that $p_j \leq s \cdot LB_j$, we deduce that the makespan which results from assigning job J_j to machine i does not exceed $(2+\alpha) \cdot LB_j$. In any case, we see that we are able to bound, at the end of every iteration of *Assign*, the current makespan by $(2+\alpha) \cdot LB_j$. Since LB_j is a lower bound of the optimal makespan related to the *offline Uniform Machine Scheduling* problem induced by the job set $J(j) = \{J_1, ..., J_j\}$, we conclude. □

Theorem 2. *Given the speed s value,* $1 < s \leq 2$, *and the number k of machines with speed s. Then, for any value* $\alpha > 0$, *there exists* m_0 *such that if an Online Uniform Machine Scheduling instance, defined with k machines with speed s and* $m-k$ *machines with speed* 1, *is such that* $m \geq m_0$, *then the Assign algorithm may be applied to this instance in such a way that, for any input job list* σ: $F(Assign, \sigma) \leq (2+\alpha) \cdot Opt(\sigma)$.

Proof. It comes in a straightforward way from the fact that, if m is large enough, then it is possible to compute R, m_1, m_2 in such a way that relations (2)-(5) hold. □

Remark. It should be mentioned that it is possible to reverse the way we have been using inequalities (2)-(5) in order to get a lower bound for the worst-case performance of the *Assign* algorithm. First, we may notice that we may generate input job lists σ, such that (*) inequality is going to hold as an equality, which will means that the worst case performance of *Assign* is going to converge to $(2+\alpha) \cdot LB(\sigma)$ when the size of s is going to increase. On the other side, we may, while starting from m_2, k and s, derive α, R and m_1 according to (2.5), and with minimal values. Indeed, when m_2 (and R) is fixed, the smallest value of α which ensures (*), is the value α_1 such that $(1+\alpha_1/2)^{1-R} \leq (2+\alpha_1-s)$. We may consider an example, related to $s=2, k=1, m_2=7$. In such a case, we derive from (2)-(5): $R = m_2 = 7, m_1 = 31$ and $m = 39, \alpha \approx 0.41$. Therefore for any $m \geq 39$ the proposed algorithm provides W.C.P. of at least $2.41 \cdot LB(\sigma)$.

Acknowledgements
This research was supported in part by Labex IMOBS3 and FEDER Funding.
Vladimir Kotov was also supported in part by BRFFI F15MLD-022.

References

[1] Cho Y., Sahni S. *Bounds for list schedules on uniform processors*. SIAM Journal on Computing, 1980, **9**, 91–103.

[2] Graham R. L. *Bounds on multiprocessing timing anomalies*. SIAM Journal on Applied Mathematics, 1969, **17**, 263–269.

[3] Li R., Shi L. *An on-line algorithm for some uniform processor scheduling*. SIAM Journal on Computing, 1998, **27**, 414–422.

[4] Cheng T. C. E., Ng C. T., Kotov Vladimir. *A new algorithm for online uniform-machine scheduling to minimize the makespan*. Information Processing Letters, 2006, **99**, 102–105.

[5] Liu Ming, Xu Yinfeng, Chu Chengbin, Zheng Feifeng. *Online scheduling on two uniform machines to minimize the makespan*. Theoretical Computer Science, 2009, **410 (21–23)**, 2099–2109.

[6] Angelelli E., Speranza Maria Grazia, Tuza Z. *Semi-online scheduling on two uniform processors*. Theoretical Computer Science, 2008, **393**, 211–219.

[7] Wen J., Du D. *Preemptive on-line scheduling for two uniform processors*. Operations Research Letters, 1998, **23**, 113–116.

[8] Kellerer H., Kotov V. *An efficient algorithm for bin stretching*. Operations Research Letters, 2013, **41(4)**, 343–346.

[9] Kellerer H., Kotov V., Gabay M. *An efficient algorithm for semi-online multiprocessor scheduling with given total processing time*. Journal of Scheduling 2015, **18**, 623–630.

Received November 16, 2015

Alexandre Dolgui
LIMOS, UMR CNRS 6158, Ecole Nationale Superieure
des Mines, 158 cours Fauriel, 42023
SAINT-ETIENNE cedex - FRANCE
E-mail: *dolgui@emse.fr*

Vladimir Kotov
Belarusian State University, 4, Nezalezhnasti Av., 220030,
MINSK - BELARUS
E-mail: *kotovvm@bsu.by*

Alain Quilliot
LIMOS, UMR CNRS 6158, ISIMA, Complexe scientifique
des Cezeaux, 63173 AUBIERE cedex - FRANCE
E-mail: *quilliot@isima.fr*