GENERAL PARAMETRIC SCHEME FOR THE UNIFORM SCHEDULING PROBLEM WITH TWO DIFFERENT SPEEDS

A. Dolgui\textsuperscript{1}, V. Kotov\textsuperscript{2}, A. Quilliot\textsuperscript{3}, A. Nekrashevich\textsuperscript{2}

\textsuperscript{1}Ecole des Mines de Nantes IRCCYN  
Nantes, France

\textsuperscript{2}Belarusian State University  
Minsk, Belarus

\textsuperscript{3}LIMOS, UMR CNRS 6158, ISIMA  
Aubiere, France

e-mail: alexandre.dolgui@mines-nantes.fr, quilliot@isima.fr, kotovvm@bsu.by, nekrald@gmail.com

We consider the Online Uniform Machine Scheduling problem on $m$ machines in the case when speed $s_i = 1$ for $i = m - k + 1, \ldots, m$ and $s_i = s$, $s > 1$, for $i = 1, 2, \ldots, k$. We propose a parametric scheme with worst-case behavior 2,618 when $1 < s \leq 2$ and with asymptotic worst case behavior $(1 + s + \sqrt{5 - 2s + s \ast s})/2$ for all $s$ when ratio $m/k$ tends to infinity. Moreover, some computation approaches are studied.

Keywords: online algorithm; uniform machine scheduling; worst-case behavior.
INTRODUCTION

In this paper, we study the classic problem of scheduling jobs online on \( m \) uniform machines \( (M_1, M_2, \ldots, M_m) \) with speeds \( (s_1, s_2, \ldots, s_m) \) without preemption: jobs arrive one at a time, according to a linear ordering (a list) \( \sigma \), with known processing times and must immediately be scheduled on one of the machines.

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 \( s_m = s > 1 \), Cho and Sahni also showed that the LS algorithm has a worst-case bound \( c \) of \( 1 + \left( \frac{m-1}{\min(2, s)(m + s - 1)} \right) \leq 3 - 4/(m+1) \), 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 \( m \). Some results in case of fixed number of machines can be found in [4–6]. The worst-case behavior of all previous algorithms occurs when \( m \) tends to infinity.

A PARAMETRIC SCHEME

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

We introduce the notation:
- \( m \) denotes the total number of machines;
- \( k \) denotes the number of machines with a speed \( 1 < s \leq 2 \), \( k \) is a constant;
- \( L_{ij} \) the current load of machine \( i \) before assigning job \( J_j \);
- \( L^*_{ij} \) the current load of machine \( i \) after assigning job \( J_j \);
- \( V_j \) the theoretical optimal makespan for jobs \( J_1, \ldots, J_j \) (set \( J(j) \)).

It is easy to check that, if we denote by \( q_1, \ldots, q_j \) the processing time of the jobs of \( J(j) \), sorted in decreasing order \( q_1 \geq q_2 \geq \ldots \geq q_j \), then we may state:

Property 1. The following inequalities hold:
- \( V_j \geq (q_1 + q_2 + \ldots + q_j)/(m-k+s\cdot k) \);
- \( V_j \geq q_1/s \);
- \( V_j \geq \min \{ q_{(z-1)k+1}, q_{(z-2)k+1} \ldots, q_{(z-1)k+s} \}/s \).

\( LB_j \) is a lower bound for the optimal makespan and \( LB_{j-1} \leq LB_j \)

THE ONLINE ALGORITHM ASSIGN

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

\[
\begin{align*}
(B-1)s\cdot k + (1-\varphi) s \cdot m_1 & \geq s \cdot k + m_1 + m_2, \\
k + m_1 + m_2 & = m, \\
m_2 & = R \cdot (z-1) \cdot k, \\
(1-\varphi) \cdot B & \leq (\varphi \cdot B)^R \cdot (B-s)
\end{align*}
\]
\[ z - 1 < s \leq z, \]  
(5)\[
\phi \in [0,1], k, m_1, m_2, R, z \in \mathbb{N}. \]  
(6)

We split the machine set into three classes:
machines with speed \( s \) are called \textit{Fast};
we pick up \( m_2 \) machines among the \( m - k \) machines with speed 1 and call them \textit{Normal};
the \( m_2 \) remaining machines with speed 1 are called \textit{Reserved}.

By the same way, we say that job \( J_j \), which arrives at step \( j \) is:
\textit{Small} if its processing time \( p_j \) is no greater than \( \phi \cdot B \cdot LB_j \);  
\textit{Large} otherwise.

We say that job \( J_j \) fits with machine \( M_i \), \( i = 1, \ldots, m \), if
\[ L_{i,j} + p_j/s_i \leq B \cdot LB_j. \]

For the purpose of algorithm description we would separate \( m_2 \) processors onto the groups \( G_1, \ldots, G_{m_2} \). Every group consists of \( (z - 1)k \) processors. The processors are numbered from 1 to \( (z - 1)k \) within the group.

The main idea of the algorithm is to guarantee the following property: in case when big job does not fit any of the machines from classes \textit{Big} and \textit{Normal}, it would fit to one of the machines from class \textit{Reserved}.

Algorithm is build on the basis of any solution for the system (1–6). Initialize \( u = 1 \). Then do the following steps for each new job \( j \):

1. Recalculate lower bound \( LB_j \) for the job \( j \) according to 1.
2. If the job \( j \) fits into one of the processors in classes \textit{Fast} or \textit{Normal}, then assign it there and go to the next job.
3. This instructions are executed only if \( u \leq (z - 1)k \). Assign the current job onto machine with number \( u \) in \( G_1 \); denote this processor by \( g_u \). Denote by \( i \) the machine from class \textit{Normal} with minimal current load. Switch classes for processors \( g_u \) and \( i \), i. e. machine \( i \) substitutes in the group \( G_1 \) machine \( g_u \) with the number \( u \) within the group, and moved to the class \textit{Reserved}. Correspondingly, \( g_u \) moves to the class \textit{Normal}. After that increment \( u \) with \( u := u + 1 \).
4. In case we have \( u > (z - 1)k \) after the previous step, move group numeration \( G_1, \ldots, G_{m_2} \) cyclically left onto 1. I. e. the group \( G_r \) becomes \( G_{r-1} \) if \( r > 1 \), and group \( G_1 \) becomes \( G_{m_2} \). Afterwards set \( u := 1 \).
5. Move to the next iteration and the next job, if there is any.

\textbf{Property 2.} If big job does not fit onto the processor \( i \) from class \textit{Fast}, then \( L_{i,j} \) is strictly greater than \( (B - 1) \cdot LB_j \).

\textbf{Lemma 1.} For each step \( j \) of the algorithm either exists Fast processor \( i \) with load \( L_{i,j} \leq (B - 1) \cdot LB_j \), or Normal processor \( i \) with load at most \( (1 - \phi) \cdot B \cdot LB_j \).

\textbf{Property 3.} The system (1–6) have always a feasible solution with fixed \( k, m, s \).

\textbf{Theorem 1.} For any fixed \( m, k, s \) the system (1–6) has at least one solution \( (B, m_1, m_2, z, R) \). For any solution of the system the algorithm built upon it is \( B \)-competitive.

\textbf{LITERATURE}