ОБЩАЯ ПАРАМЕТРИЧЕСКАЯ СХЕМА УНИВЕРСАЛЬНОЙ ПРОБЛЕМЫ ПЛАНИРОВАНИЯ С ДВУМЯ РАЗЛИЧНЫМИ СКОРОСТЯМИ

А. Долгий[1], В. Котов[2], А. Кьюиллиот[3], А. Некрашевич[2]

[1]*Ecole des Mines de Nantes IRCCYN*
Нант, Франция
[2]*Белорусский государственный университет*
Минск, Беларусь
[3]*LIMOS, UMR CNRS 6158, ISIMA*
Обьера, Франция
e-mail: alexandre.dolgui@mines-nantes.fr, quilliot@isima.fr , kotovvm@bsu.by, nekrald@gmail.com

Мы рассматриваем проблему онлайнового универсального машинного планирования на m машинах в случае, когда скорость $si = 1$ для $i = m - k+ 1, \ldots, m$ и $si = s$, $s > 1$, для $i = 1, 2, \ldots, k$. Мы предлагаем параметрическую схему с поведением худшего случая 2,618, когда $1 < s \le 2$, и асимптотическим поведением худшего случая $(1 + s + \sqrt{5 - 2s + s * s})/2$ для всех s, когда отношение m/k стремится к бесконечности. Кроме того, изучены некоторые вычислительные подходы.

Ключевые слова: онлайновый алгоритм; универсального машинного планирования; поведение худшего случая.

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

A. Dolgui[1], V. Kotov[2], A. Quilliot[3], A. Nekrashevich[2]

[1]*Ecole des Mines de Nantes IRCCYN*
Nantes, France
[2]*Belarusian State University*
Minsk, Belarus
[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 $1 < s \le 2$ and with asymptotic worst case behavior $(1 + s + \sqrt{5 - 2s + s * s})/2$ 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, ..., M_m) with speeds (s_1, s_2, …, 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.

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 $s_i = 1$ ($i = 1, \dots, m-1$) and $s_m = s > 1$, Cho and Sahni also showed that the LS algorithm has a worst-case bound c of $1+ (m-1)\cdot(\mathrm{Min}(2; s)/(m+s-1)) \leq 3-4/(m+1)$, 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 $s_i = 1$ ($i = 1, \dots, m-1$) 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 jth job in the list σ, and say that job J_j arrives at *step j* according to σ. 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_i 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;

Any time we have to deal with a current job J_j of the input list σ, and

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

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

V_j the theoretical optimal makespan for jobs J_1.., J_j (set $J(j)$).

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

Property 1. The following inequalities hold:

$V_j \geq (q_1 + q_2 + \dots + q_j)/(m-k+s*k)$;

$V_j \geq q_1/s$;

$V_j \geq \min \{q_{(z-1)k+1}, (q_{(z-1)(k-1)+} \dots + q_{(z-1)k+1})/s\}$.

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 α is given together with three integral numbers R, m_1 and m_2 in such a way that:

$$(B-1)*s*k + (1-\varphi)*m_1 \geq s*k + m_1+m_2, \tag{1}$$

$$k + m_1+m_2 = m, \tag{2}$$

$$m_2 = R*(z-1)*k, \tag{3}$$

$$(1-\varphi)*B \leq (\varphi*B)^R*(B-s) \tag{4}$$

$$z-1 < s \le z, \quad (5)$$

$$\varphi \in [0,1],\ k, m_1,\ m_2,\ R,\ z \in \mathrm{N}. \quad (6)$$

We split the machine set into three classes:

machines with speed s are called *Fast*;

we pick up m_1 machines among the $m - k$ machines with speed 1 and call them *Normal*;

the m_2 remaining machines with speed 1 are called *Reserved.*

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

Small if its processing time p_j is no greater than $\phi * B * LB_j$;

Large otherwise.

We say that job J_j fits with machine M_i, $i = 1, ..., m$, if $L_{i,j} + p_j/s_i \le B * LB_j$.
For the purpose of algorithm description we would separate Reserved processors onto the groups $G_1, \ldots, G_R$. 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 Big and Normal, it would fit to one of the machines from class 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 Fast or Normal, then assign it there and go to the next job.
3. This instructions are executed only if $u \le (z-1)k$. 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 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 Reserved. Correspondingly, g_u moves to the class 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 $G_1, \ldots, G_R$ cyclically left onto 1. I. e. the group G_r becomes G_{r-1} if $r > 1$, and group G_1 becomes G_R. Afterwards set $u := 1$.
5. Move to the next iteration and the next job, if there is any.

Property 2. If big job does not fit onto the processor i from class Fast, then $Li;j$ is strictly greater than $(B - 1)\ LB_j$.

Lemma 1. For each step j of the algorithm either exists Fast processor i with load $L_{i,j} \le$ $\le (B - 1) * LB_j$, or Normal processor i with load at most $(1 - \varphi) * B * LB_j$.

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

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.

LITERATURE

1. Cho Y., Sahni S. Bounds for list schedules on uniform processors // SIAM J. Comput. – 1980. Vol. 9. P. 91–103.

2. Graham R. L. Bounds on multiprocessing timing anomalies // SIAM J. Appl. Math. 1969. Vol. 17. P 263–269.

3. Li R., Shi L. An on-line algorithm for some uniform processor scheduling // SIAM J. Comput. 1998. Vol. 27. P. 414–422.

4. Cheng T. C. E., Ng C. T., Kotov V. A new algorithm for online uniform-machine scheduling to minimize the makespan // Information Processing Letters. 2006. Vol. 99. P. 102–105.

5. Angelelli E., Speranza M. G., Tuza Z. Semi-online scheduling on two uniform processors // Theoretical Computer Science. 2008. Vol. 393. P. 211–219.

6. Wen J., Du D. Preemptive on-line scheduling for two uniform processors // Operations Research Letters. 1998. Vol. 23. P. 113–116.