浙江理工大学学报，2019，41(4)：527-531
Journal of Zhejiang Sci-Tech University
DOI：10.3969/j.issn.1673－3851(n).2019.04.017

三台同类机 MapReduce 排序问题的最优算法

韩曙光，郑　聪
（浙江理工大学理学院，杭州 310018）

摘　要：研究 MapReduce 环境下的可中断同类平行机排序问题。在 MapReduce 环境中，每个工件含有两种类型的任务集，即 Map 任务集和 Reduce 任务集。在加工完工件的 Map 任务集后才能开始加工 Reduce 任务集中的任务。考虑 Map 任务为可分的情况，即 Map 任务可以任意分割为不同的小任务并能在不同机器上同时进行并行加工，而对于 Reduce 任务则考虑允许中断的情形，目标设为极小化最大完工时间。针对三台同类机的离线排序问题，通过分解所有实例的类型，给出了最优解算法。

关键词：MapReduce；同类机排序；完工时间；最优算法
中图分类号：O233　　**文献标志码**：A　　**文章编号**：1673-3851 (2019) 07-0527-05

An optimal preemptive algorithm for MapReduce scheduling on three uniform machines

HAN Shuguang, *ZHENG Cong*
(School of Sciences, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: Interruptible uniform parallel machine scheduling under MapReduce environment is studied in this paper. Under MapReduce environment, each workpiece contains two types of task sets: namely Map task set and Reduce task set. The tasks in Reduce task set can only be processed after finishing all tasks in Map tasks. The Map tasks are separable, i.e., they can be arbitrarily split into different small tasks and processed on different machines in parallel. Interruption is considered for Reduce tasks. Our objective is to minimize the maximum makespan. For the offline scheduling problem of three uniform machines, an optimum solution algorithm is provided by the analysis of the types of all instances.

Key words: MapReduce; uniform parallel machine scheduling; makespan; optimal scheduling

0　引　言

MapReduce[1]是由 Google 提出的一种编程模型，主要是用来处理大规模数据信息的一种计算模型和方法，它能在云端组织并分析数据，进而输出相应的结果。它的主要过程分两个阶段进行，即 Map 阶段和 Reduce 阶段。Map 阶段通过分析初始数据和提取特征得到键值对（Key-value pairs），而 Reduce 阶段则通过前者输出的键值对计算出最后的结果。所有的任务都在一组平行机上进行处理，因而可归结为平行机排序问题。

MapReduce 环境下的平行机排序问题比经典的平行机排序问题更为复杂，近十年来，大部分的研究都集中在启发式算法以及数据实验等方面[2-4]。对于 MapReduce 平行机排序问题的理论研究近几年也得到了国内外学者的广泛关注。Zhu 等[5]研究了 Map 任务可分的 m 台同型机排序问题的离线模型。他们分别针对 Reduce 任务可中断和不可中断

收稿日期：2018－09－18　　网络出版日期：2018－11－01
基金项目：国家自然科学基金项目(11571013,11471286,11701518)
作者简介：韩曙光(1977－　)，男，江苏建湖人，副教授，博士，主要从事组合优化方面的研究.

情形,给出了一个最优解算法和一个最坏情况界为$\frac{3}{2}-\frac{1}{2m}$的近似算法。此外,他们还考虑了极小化总完工时间的可中断和不可中断近似算法。对于Map任务可分的m台同类机排序问题,Jiang等[6]在Reduce可中断情形下给出了一个最坏情况界为2的近似算法,而Reduce不可中断时,设计了一个最坏情况界至多为$\max\left\{1+\frac{a}{2}-\frac{1}{n},a\right\}$的近似算法,其中$a$为最快机器速度与最慢机器速度之比。周维[7]对上述结果进行了改进,分别对Reduce可中断和不可中断情形给出了最坏情况界为$2-\frac{s_1}{\sum_{j=1}^{m}s_j}$和$2+\frac{\sqrt{3}}{3}$的近似算法。

除了上述研究MapReduce平行机排序问题的离线模型,一些学者研究了该问题的在线情形。Jiang等[8]给出了Map任务可分的两台机情形的最优可中断在线算法,竞争比为$\sqrt{2}$。Luo等[9]研究了只有在所有Map任务完成后才能获取Reduce任务信息的m台同型机在线问题,分别对Reduce可中断和不可中断情形给出了最优算法,其竞争比都为$2-\frac{1}{m}$。对于工件有释放时间的在线问题,Chen等[10]给出了Reduce任务不可中断时的一个竞争比为$2-\frac{1}{m}$的MF-LPT算法和Reduce任务可中断时的两台机最优算法。Le等[11]研究了倾斜数据输入下的在线MapReduce排序问题,给出了一个竞争别为2的在线算法。

注意到文献[6-7]中对同类机的离线情形都只给出了可中断的近似算法,本文继续考虑此类问题,给出了三台同类机情形的最优算法,从而改进了已有的结果。本文算法的主要思想是通过分析工件类型以及不同的分配方式,给出了各种类型工件的最优解结构,为后续设计一般的m台同类机问题的最优算法提供了思路。

1 问题描述及符号定义

本文所考虑的问题可具体描述为:给定一个n个工件的工件集合$\{J_1,J_2,\cdots,J_n\}$和包含三台机器的机器集$\{\sigma_1,\sigma_2,\sigma_3\}$。所有的工件必须在这三台机器上进行加工。这三台机器的速度分别为s_1,s_2和s_3,不妨设$s_1\geqslant s_2\geqslant s_3$。工件$J_j$包含一个Map任务集$M_j$和一个Reduce任务集$R_j$。工件$J_j$的Reduce任务必须在$J_j$的Map任务完工后才可以开始加工。在本文中,Map任务是可分的(Fractional),即Map任务可以任意分割且同一Map任务的不同部分可以同时在不同的机器上并行加工。对于Reduce任务,考虑可中断(Preemptive)情形,即Reduce任务可以在加工过程中被中断,剩余部分可在后续的时间在任意机器上加工。目标是极小化最大完工时间。

为方便叙述,引入以下符号:

M_j:表示Map任务集或者M_j中所有任务的总长度;

R_j:表示Reduce任务集或者R_j中所有任务的总长度;

u_j:表示R_j中Reduce任务的个数;

r_j^i:表示R_j中的一个Reduce任务或者该任务的长度,且假设$r_j^1\geqslant r_j^2\geqslant\cdots\geqslant r_j^{u_j}$;

$r_i^{t_i}$:表示所有Reduce任务集$\bigcup_{1\leqslant j\leqslant n}R_j$中第$i$大的任务,这里$t_i$表示$r_i^{t_i}$属于$R_{t_i}$;

q:表示所有$\bigcup_{1\leqslant j\leqslant n}R_j$中的Reduce任务的总个数,即$q=\sum_{j=1}^{n}u_j$;

$\widetilde{M}$:表示除M_{t_1}和M_{t_2}以外的所有Map任务的集合。

2 最优解算法

经典同类机排序问题是本文所研究问题的一种特殊情形,因此已有的经典问题的最优算法对研究本文的问题具有重要的借鉴意义。对于经典的同类机排序问题,Gonzalez和Sahni早在1978年给出了线性时间的最优算法,记作GS算法[12]。在经典同类机排序问题中,给定n个工件的工件集$\{p_1,p_2,\cdots,p_n\}$和三台机器$\{\sigma_1,\sigma_2,\sigma_3\}$,其中$p_1\geqslant p_2\geqslant\cdots\geqslant p_n$,三台同类机的速度分别为$\{s_1,s_2,s_3\}$且$s_1\geqslant s_2\geqslant s_3$,由GS算法可得最优目标值为

$$C_{\max}=\max\left\{\frac{p_1}{s_1},\frac{p_1+p_2}{s_1+s_2},\frac{\sum_{j=1}^{n}p_j}{\sum_{j=1}^{3}s_j}\right\}。$$

为方便后续算法的描述,记

$$\omega_1=\frac{r_1^{t1}}{s_1},\omega_2=\frac{r_2^{t2}}{s_2},\omega_3=\frac{\sum_{j=3}^{q}r_j^{tj}}{s_3},$$

$$\omega_4=\frac{r_1^{t1}+r_2^{t2}}{s_1+s_2},\omega_5=\frac{\sum_{j=2}^{q}r_j^{tj}}{s_2+s_3},\omega_6=\frac{\sum_{j=1}^{q}r_j^{tj}}{s_1+s_2+s_3}。$$

在给出本文主要算法之前,先给出一个算法

A_0，该算法用来加工所有 Reduce 任务，其中步骤 c）中对于任务在机器上的加工方式与 GS 算法一致。

a）令 $W=\bigcup_{1\leqslant j\leqslant n}R_j$，$e_i$ 表示工件在机器 $\sigma_i(i=1,2,3)$ 上的完工时间。

b）整理 W 中的任务得到 $r_1^{t_1}\geqslant r_2^{t_2}\geqslant\cdots\geqslant r_q^{t_q}$。

c）根据 ω_i 的大小分以下情况考虑，$1\leqslant i\leqslant 6$。

c1）若 $\omega_1>\omega_4$，$\omega_1>\omega_6$ 且 $\omega_2>\omega_5$，则将 $r_1^{t_1}$ 和 $r_2^{t_2}$ 分别安排在第一台和第二台机器上加工，剩下所有的 Reduce 任务放在第三台机器上加工。此时有 $e_i=\omega_i$，$i=1,2,3$。

c2）若 $\omega_1>\omega_4$，$\omega_1>\omega_6$ 且 $\omega_2\leqslant\omega_5$，则将 $r_1^{t_1}$ 安排在第一台机器上加工，将 $r_2^{t_2}$ 及剩下所有的 Reduce 任务放在第二和第三台机器上加工。此时有 $e_1=\omega_1$ 和 $e_2=e_3=\omega_5$。

c3）若 $\omega_4>\omega_1$，$\omega_4>\omega_6$，则将 $r_1^{t_1}$ 和 $r_2^{t_2}$ 安排在在前两台机器上加工，剩下所有的任务放在第三台机器上加工。此时有 $e_1=e_2=\omega_4$ 和 $e_3=\omega_3$。

c4）若 $\omega_6>\omega_1$，$\omega_6>\omega_4$，则将所有 Reduce 任务放在三台机器上加工。此时机器的完工时间为 $e_1=e_2=e_3=\omega_6$。

算法 A_0 的执行情况见图 1(a)—(d)，它们分别给出了四种情况下关于 Reduce 任务的排序结果。

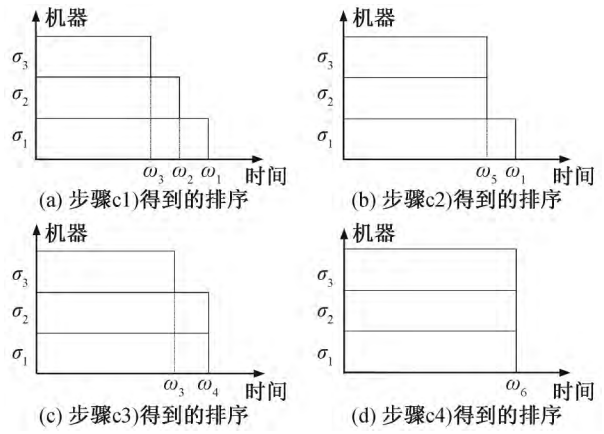


图 1　Reduce 任务排序结果

接下来给出本文所研究问题的最优解算法，记作 A。

算法 A：

a）用算法 A_0 得到所有 $\bigcup_{1\leqslant j\leqslant n}R_j$ 中的任务的排序；

b）若 $\omega_6>\omega_1$，$\omega_6>\omega_4$，则将所有 Map 任务平均安排到三台处理机上同时加工，转入 f）。否则，进入 c）；

c）将 M_{t_1} 尽可能早的安排到时间 e_1 之后；

d）将 M_{t_2} 尽可能安排在空闲时间$[e_2,e_1]$中的机器上，若没有空闲，则尽可能早的安排到时间 e_1 之后；

e）将剩余的全部 Map 任务 $\bar{M}$ 安排在 e_1 之前的空闲时间，如果 e_1 之前没有多余的空闲，则尽可能早的安排到时间 e_1 之后；

f）按照 Map 部分在前，Reduce 部分在后，将得到的排序顺序互换，停止。

定理 1 给出了问题的最优解的下界，后续将证明算法的目标函数值与下界相等，进而证明本文给出的算法是最优算法。

定理 1　问题的最优解下界 LB 至少为

$$LB=\max\left\{\frac{\sum_{j=1}^{n}(M_j+R_j)}{\sum_{j=1}^{3}s_j},\max_{1\leqslant i\leqslant 2}\{f_i\}\right\}\quad(1)$$

其中 $f_i=\dfrac{\sum_{j=1}^{i}M_{tj}+\sum_{j=1}^{i}r_j^{tj}+e_i\left(\sum_{j=i+1}^{3}s_j\right)}{\sum_{j=1}^{3}s_j}$。

证明：将 Map 任务和 Reduce 任务平均分配到三台机器上，很容易得到算法 A 的目标值至少是 $\dfrac{\sum_{j=1}^{n}\left(M_j+R_j\right)}{\sum_{j=1}^{3}s_j}$。下面证明算法 A 的目标值至少是 f_i，$i=1,2$。

对任意 $1\leqslant i\leqslant 2$，考虑 $J_1=\{r_1^{t_1}\}\cup M_{t_1}$，$J_2=\{r_1^{t_1},r_2^{t_2}\}\cup(M_{t_1}\cup M_{t_2})$ 的最优排序。令 z_i^* 为 J_i 的最优解，先证明对 J_1 存在一个最优排序，使得 $\{r_1^{t_1}\}$ 安排在第一台机器上，则有 $M_{t_1}+r_1^{t_1}\leqslant z_1^*\sum_{j=1}^{3}s_j-(s_2+s_3)e_1$，即

$$z_1^*\geqslant\frac{M_{t_1}+r_1^{t_1}+e_1(s_2+s_3)}{\sum_{j=1}^{3}s_j}=\frac{M_{t_1}}{\sum_{j=1}^{3}s_j}+r_1^{t_1}=f_1。$$

对于 J_2，则存在一个最优排序使得$\{r_1^{t_1},r_2^{t_2}\}$安排速度最快的第一和第二台机器上，则有 $M_{t_1}+M_{t_2}+r_1^{t_1}+r_2^{t_2}\leqslant z_2^*\sum_{j=1}^{3}s_j-s_3e_2$，即 $z_2^*\geqslant\dfrac{M_{t_1}+M_{t_2}+r_1^{t_1}+r_2^{t_2}+e_2s_3}{\sum_{j=1}^{3}s_j}=f_2$。

综上可得，算法 A 的目标值至少为 f_i。

定理 2　算法 A 的目标函数值满足 $C_A=LB$，因此该算法是最优的。

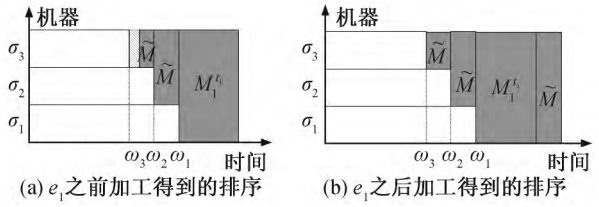
证明:记 c_1 为算法 A 安排完 Reduce 任务$\{r_1^{t_1}\}$和 Map 任务 M_{t_1} 后的完工时间,c_2 表示算法 A 安排完 Reduce 任务$\{r_1^{t_1}, r_2^{t_2}\}$和 Map 任务 $M_{t_1} \cup M_{t_2}$ 后的完工时间,分情况讨论:

a) $\omega_1 > \omega_4$ 且 $\omega_1 > \omega_6$,此时将 M_{t_1} 放在 e_1 之后尽可能早的位置,可得

$$c_1 = e_1 + \frac{M_{t_1}}{\sum_{j=1}^{3} s_j} = \frac{M_{t_1} + e_1 \sum_{j=1}^{3} s_j}{\sum_{j=1}^{3} s_j} = \frac{M_{t_1} + e_1 s_1 + e_1(s_2 + s_3)}{\sum_{j=1}^{3} s_j} = \frac{M_{t_1} + r_1^{t_1} + e_1(s_2 + s_3)}{\sum_{j=1}^{3} s_j} = f_1。$$

接下来分两个情形讨论:

a1) $\omega_2 > \omega_5$。若 $t_2 = t_1$,则 $c_2 = c_1 = f_1$。最后考虑 $\tilde{M}$,若所有任务在 e_1 之前加工,则有 $C_A = c_1 = f_1$,反之,$C_A = \frac{\sum_{j=1}^{n}(M_j + R_j)}{\sum_{j=1}^{3} s_j}$,见图 2(a)—(b)。



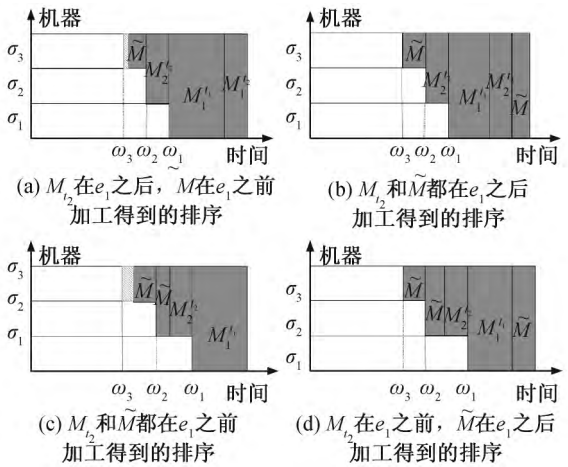
(a) e_1之前加工得到的排序 (b) e_1之后加工得到的排序

图 2 $t_2 = t_1$ 时 Map 任务排序

若 $t_2 \neq t_1$,将 M_{t_2} 分配到空闲时间$[e_2, e_1]$内,若 $\frac{M_{t_2}}{s_2 + s_3} > e_1 - e_2$,则由算法 A 第 d)步可知,先将 M_{t_2} 安排在$[e_2, e_1]$之间的空闲时间,剩余的工件放在 e_1 之后尽可能早的位置,此时

$$c_2 = \frac{M_{t_1} + M_{t_2} + r_1^{t_1} + r_2^{t_2} + e_2 s_3}{\sum_{j=1}^{3} s_j} = f_2。$$

对于 $\tilde{M}$,若所有任务在 e_1 之前加工,$C_A = c_2 = f_2$,反之,$C_A = \frac{\sum_{j=1}^{n}(M_j + R_j)}{\sum_{j=1}^{3} s_j}$,见图 3(a)—(b)。另一方面,若 $\frac{M_{t_2}}{s_2 + s_3} \leqslant e_1 - e_2$,此时 $c_2 = c_1 = f_1$,最后考虑 $\tilde{M}$,若所有任务在 e_1 之前加工,$C_A = c_1 = f_1$,反之,$C_A = \frac{\sum_{j=1}^{n}(M_j + R_j)}{\sum_{j=1}^{3} s_j}$,见图 3(c)—(d)。

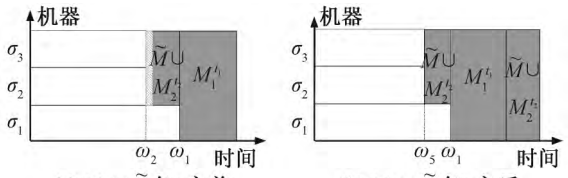


(a) M_{t_2}在e_1之后,$\tilde{M}$在e_1之前加工得到的排序 (b) M_{t_2}和$\tilde{M}$都在e_1之后加工得到的排序

(c) M_{t_2}和$\tilde{M}$都在e_1之前加工得到的排序 (d) M_{t_2}在e_1之前,$\tilde{M}$在e_1之后加工得到的排序

图 3 $t_2 \neq t_1$ 时相应的 Map 任务排序

a2) $\omega_2 \leqslant \omega_5$,此时若 $M_{t_2} \cup \tilde{M}$ 中所有的 Map 任务能在 e_1 之前加工,则有 $C_A = f_1$,否则不难得到 $C_A = \frac{\sum_{j=1}^{n}(M_j + R_j)}{\sum_{j=1}^{3} s_j}$,见图 4。



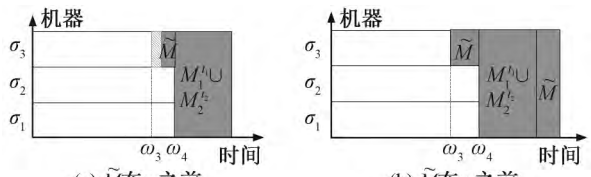
(a) $M_{t_2} \cup \tilde{M}$在e_1之前加工得到的排序 (b) $M_{t_2} \cup \tilde{M}$在e_1之后加工得到的排序

图 4 $\omega_2 \leqslant \omega_5$ 时的排序

b) $\omega_4 > \omega_1$ 且 $\omega_4 > \omega_6$。此时

$$c_1 = c_2 = e_1 + \frac{M_{t_1} + M_{t_2}}{\sum_{j=1}^{3} s_j} = \frac{M_{t_1} + M_{t_2} + e_1(s_1 + s_2) + e_1 s_3}{\sum_{j=1}^{3} s_j} = \frac{M_{t_1} + M_{t_2} + r_1^{t_1} + r_2^{t_2} + e_2 s_3}{\sum_{j=1}^{3} s_j} = f_2。$$

对于 $\tilde{M}$,若所有任务在 e_1 之前加工,则有 $C_A = f_1 = f_2$,否则不难得到 $C_A = \frac{\sum_{j=1}^{n}(M_j + R_j)}{\sum_{j=1}^{3} s_j}$,见图 5。



(a) $\tilde{M}$在e_1之前加工得到的排序 (b) $\tilde{M}$在e_1之前加工得到的排序

图 5 $\omega_4 > \omega_1$ 且 $\omega_4 > \omega_6$ 时的排序

c) $\omega_6 > \omega_1$ 且 $\omega_6 > \omega_4$。此时由算法 A_0 可得，安排完所有 Reduce 任务 $\bigcup_{1\leqslant j\leqslant n} R_j$ 后三台机器的完工时间相等，即 $e_1 = e_2 = e_3 = \dfrac{\sum_{j=1}^{n} R_j}{\sum_{j=1}^{3} s_j}$。同时，由算法可知，所有 Map 任务也将平均分配到三台机器上，则有 $C_A = \dfrac{\sum_{j=1}^{n} (M_j + R_j)}{\sum_{j=1}^{3} s_j}$。

综上所述，由定理 2 得出结论 $C_A = LB$ 且算法 A 是最优的。

3　结　论

本文主要研究三台同类机 MapReduce 排序问题，目标是极小化最大完工时间，针对 Map 任务可分和 Reduce 任务可中断的情况，设计了最优解算法。后续研究可以考虑一般的 m 台同类机情形的最优算法以及 Reduce 任务不可中断的情形。

参考文献：

[1] Dean J, Ghemawant S. MapReduce: Simplified data processing on large clusters[J]. Communications of the ACM, 2008,51(1):107-113.

[2] Isard M, Prabhakaran V, Currey J, et al. Quincy: Fair scheduling for distributed computing clusters [C]// Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, New York:IEEE, 2009: 261-276.

[3] Gufler B, Augsten N, Reiser A, et al. Load balancing in mapreduce based on scalable cardinality estimates[C]// In Data Engineering (ICDE) 2012 IEEE 28th International Conference on, Italy: IEEE, 2012: 522-533.

[4] Bechini A, Marcelloni F, Segatori A. A MapReduce solution for associative classification of big data[J]. Information Sciences, 2015, 332:33-55.

[5] Zhu Y, Jiang Y, Wu W, et al. Minimizing makespan and total completion time in MapReduce-like systems [C]// In INFOCOM'14. IEEE, 2014:2166-2174.

[6] Jiang Y, Zhu Y, Wu W, et al. Makespan minimization for MapReduce systems with different servers [J]. Future Generation Computer Systems, 2017, 67:13-21.

[7] 周维. 若干 MapReduce 排序问题的算法研究[D].杭州：浙江理工大学,2017:5-11.

[8] Jiang Y, Zhou W, Zhou P. An optimal preemptive algorithm for online MapReduce scheduling on two parallelmachines[J].Asia-Pacific Journal of Operational Research,2018,35(3):1850013.

[9] Luo T, Zhu Y, Wu W, et al. Online makespan minimization in MapReduce like systems with complex reduce tasks[J]. Optimization Letters, 2017, 11(2): 271-277.

[10] Chen C, Xu Y, Zhu Y, et al. Online MapReduce scheduling problem of minimizing the makespan[J]. Journal of Combinatorial Optimization, 2017, 33(2): 590-608.

[11] Le Y, Liu J, Ergun F, et al. Online load balancing for MapReduce with skewed data input [C]//IEEE INFOCOM 2014 - IEEE Conference on Computer Communications. IEEE, 2014:2004-2012.

[12] Gonzalez T, Sahni S. Preemptive scheduling of uniform processor systems[J]. Journal of the Acm, 1978, 25: 92-101.

（责任编辑：康　锋）