有中断时间代价的一致并行机抢先调度问题*

孙广中[1,2], 陈国良[1,2], 许胤龙[1,2], 顾 钧[3]

[1](国家高性能计算中心(合肥),安徽 合肥 230027);
[2](中国科学技术大学 计算机科学与技术系,安徽 合肥 230027);
[3](香港科技大学 计算机科学系,香港)
E-mail: sungz@mail.ustc.edu.cn
http://www.ustc.edu.cn

摘要: 提出了一种具有中断时间代价的抢先调度问题($P|ptmn(\delta)|C_{max}$):在抢先调度中,一个任务发生一次中断,其总的执行时间会增加一个δ.该问题在工程任务分配、分布式计算和网络通信等实际问题中有着广泛的应用背景.证明了这是一个 NP-hard 问题,给出了一个时间复杂度为 $O(n\log n+m)$的脱线近似算法 LPT-Wrap,其近似比小于等于 1.40825,并分析了 $P|ptmn(\delta)|C_{max}$ 的在线特性,给出一个线性时间复杂度的在线近似算法,其竞争比为 2.

关 键 词: 调度问题;抢先允许;组合优化;时间代价;NP 难;近似算法;近似比
中图法分类号: TP301 文献标识码: A

调度问题是一类组合优化问题,在计算机及通信等许多领域有着广泛的应用,在理论上又与算法设计、复杂性理论关系密切,因而引起了许多学者的研究兴趣[1].根据实际应用中的不同要求,存在很多种调度问题,目前被研究过的就有几百种.一个调度问题可由 3 方面的特征来刻画[2]:机器环境、问题的限制条件和问题的目标函数.其中最早被研究的问题是一致并行机(identical parallel machine)上最小化停机时间(makespan)问题 $P||C_{max}$:给定 n 个任务和 m 台机器,机器环境为一致并行机(各个机器处理速度一致且相互独立),每个任务有一个执行时间,要在某台机器上执行完毕且执行过程中不能发生中断.每台机器在任一时刻只能处理一个任务,目标函数为最小化所有任务均执行完的时间[3].$P||C_{max}$ 问题是一个 NP 完全问题,关于它的近似算法的研究已有相当丰富的结果[1,2].

已有的调度问题虽然众多,但根据实际情况,近年来依然有更为实际的新模型被研究者们陆续提出,并加以研究[2,4].在计算机及通信领域中大量存在着抢先调度的情况:一个任务可以在执行一段时间后停止执行,未执行部分在以后的某个时刻重新执行.在实际的抢先调度中,当任务发生中断后,该任务的总执行时间往往会增加:被中断任务的未执行部分重新执行时需要一个额外的执行时间.这种情况在大型软件项目开发[5]、分布式计算[6]、光纤网络通信[7]中都实际存在.

在已有的各种抢先调度问题的模型中,均不考虑抢先调度中任务中断所具有的额外时间代价[8,9].比如 $P|ptmn|C_{max}$ 问题就是在原来的 $P||C_{max}$ 问题中加入了中断允许的 $ptmn$ 条件:一个任务可以被中断执行,但中断不需要花费任何额外的时间.

本文提出了一种任务中断具有时间代价的允许抢先调度的条件 $ptmn(\delta)$:在每次发生任务中断时,任务的未执行部分在重新执行时需要一个额外的时间代价δ.$ptmn(\delta)$条件比 $ptmn$ 条件更加符合实际各种抢先调度的情

* 收稿日期: 2001-04-29; 修改日期: 2001-07-12
基金项目: 国家重点基础研究发展规划 973 资助项目(G1998030403)

作者简介: 孙广中(1978－),男,安徽蚌埠人,硕士生,主要研究领域为组合优化,复杂性理论;陈国良(1938－),男,安徽颍上人,教授,博士生导师,主要研究领域为并行与分布式计算,复杂性理论;许胤龙(1963－),男,安徽庐江人,副教授,主要研究领域为复杂性理论,并行算法;顾钧(1956－),男,江苏盐城人,博士,教授,博士生导师,主要研究领域为复杂性理论,算法设计.

况.虽然在实际中任务中断的时间代价往往也是任务相关的,即不同的任务中断可能具有不同的时间代价,但考虑到中断的时间代价一般主要依赖于机器本身,所以可以近似地认为在一致并行机上,所有任务都具有相同的中断时间代价.

针对这一新的限制条件,我们对一个新的调度问题 $P|ptmn(\delta)|C_{max}$ 进行了研究.首先证明了该问题是一个 NP-hard 问题.如果 $P\neq NP$,NP-hard 问题则不存在有效时间内求得最优解的算法[3],因此本文重点考虑 $P|ptmn(\delta)|C_{max}$ 问题的近似算法,并对这些算法的性能进行分析.

我们首先给出一个脱线近似算法 LPT-Wrap.它同时利用 LPT(longest processing time)策略和 wrap-around 算法[2]思想得到两个任务安排表,返回其中较好的一个.通过相当细致的分析,了解到该算法的近似比小于等于 1.408 25.然后对该问题的在线算法进行讨论,指出 LS(list scheduling)算法[3]可以用来求解该问题,其竞争比为 2.

本文第 1 节对 $P|ptmn(\delta)|C_{max}$ 问题进行分析,并证明该问题是一个 NP-hard 问题.第 2 节给出一个时间复杂度为 $O(n\log n+m)$的脱线近似算法 LPT-Wrap 算法,并对该算法的近似性能进行分析.第 3 节说明 LS 算法是该问题的一个竞争比为 2 的在线算法,时间复杂度为 $O(n)$.第 4 节总结全文,并指出下一步的工作方向.

1 具有中断时间代价的抢先调度问题

本节先给出 $P|ptmn(\delta)|C_{max}$ 问题的一些相关定义,并证明该问题是一个 NP-hard 问题.

定义 **1**. $P|ptmn(\delta)|C_{max}$ 问题是指,给定 n 个任务集合 $J=\{J_1,\ldots,J_n\}$和 m 台一致并行机,每个任务有一个执行时间 $p_j>0(1\leq j\leq n)$.每台机器在任意时刻只能处理一个任务.记任务 J_j 的执行完毕时间为 C_j,目标函数为最小化所有任务均执行完的时间,即 $\min(\max_{1\leq j\leq n}C_j)$,其中 $\max_{1\leq j\leq n}C_j$ 通常称作停机时间(makespan),记为 C_{max}.一个任务 J_j 可以在执行一段时间 t 后中断执行,同时未完成部分的执行时间会增加一个 δ,从 p_j 变为 $p_j-t+\delta$.

定义 **2**. 任务安排表 Σ 是指任务集合在机器上执行的方案.最优任务安排表是指使 $P|ptmn(\delta)|C_{max}$ 问题的停机时间最小化的任务安排表.最优任务安排表中的任务中断次数可能不同,记中断次数最少的最优任务安排表为 Σ^*.

若一个 $p_j\leq\delta$ 的任务发生中断产生两个执行时间,分别为 p_j' 和 p_j'' 的任务执行部分,则后执行部分的执行时间 p_j'' 大于 δ.此时可调整任务安排表,取消这次中断,使该任务在 p_j'' 时间内完成,其余任务的执行安排不变.由此得到命题 1.

命题 **1**. Σ^* 中发生中断的任务 J_j 的执行时间 $p_j>\delta$.

$P||C_{max}$ 问题是一个经典的 NP 完全问题[10],而 $P|ptmn|C_{max}$ 问题则是一个多项式时间内可解的 P 类问题.下面的定理将证明 $P|ptmn(\delta)|C_{max}$ 问题是一个 NP-hard 问题.

定理 **1**. $P|ptmn(\delta)|C_{max}$ 问题是 NP-hard 问题.

证明:假设 $P|ptmn(\delta)|C_{max}$ 问题具有多项式时间的完全算法 A,下面考察 $P||C_{max}$ 问题的任意输入实例 I_0,对于 I_0 中的任务集合 J 和机器数目 m 保持不变,取 $\delta=\max_{1\leq j\leq n}p_j$.这样得到 $P|ptmn(\delta)|C_{max}$ 问题的一个实例 I_0',根据假设,算法 A 可以在多项式时间内得到实例 I_0' 的一个任务安排表 Σ^*,使得 C_{max} 最小.又 $\delta=\max_{1\leq j\leq n}p_j$,由命题 1 可知 Σ^* 中没有任务发生中断,此时 Σ^* 就是 $P||C_{max}$ 问题的实例 I_0 的最优任务安排表.因此,算法 A 也是 $P||C_{max}$ 问题的多项式完全算法.又因为 $P||C_{max}$ 问题是 NP 完全的,所以 $P|ptmn(\delta)|C_{max}$ 问题是一个 NP-hard 问题. □

NP-hard 问题在 $P\neq NP$ 的假设下不存在多项式时间的完全算法,因此,下面考虑该问题的近似算法.这里先给出有关近似算法的一些概念的定义.

定义 **3**. 给定最小化问题 Π,记 $OPT(I)$ 为实例 I 的最优解,$A(I)$ 为算法 A 在实例 I 上的解,若对问题 Π 的任意实例 I,都有 $A(I)\leq\alpha OPT(I)$,则称这个算法是 α-近似的,若 α 是所能取到的最小值,则称这个算法的近似比为 α.

在最小化停机时间的调度问题中,一般使用 C^*_{max} 表示最优停机时间,用 C_A 表示某个算法 A 得到的停机时间.

定义 **4**. 如果近似调度算法 A 依序处理各输入任务,在处理当前任务的时候,不知道任何后续任务信息,并立即给出当前任务的调度安排,则称这样的近似调度算法 A 为在线(on-line)调度算法;首先得到所有任务信息之

后,统一处理所有任务安排方案的近似算法则为脱线(off-line)调度算法.在在线算法的研究中,算法的近似比常称作为算法的竞争比(competitive ratio).在实际应用中,很多时候都要求算法具有在线特性[11].

2 脱线算法

本节给出 $P|ptmn(\delta)|C_{\max}$ 问题的一个脱线近似算法 LPT-Swap,并分析该算法的近似性能.下面先给出这个算法的描述.

LPT-Swap 算法描述:

Step 1. 将任务集合 $J=\{J_1,\ldots,J_n\}$中的任务按照任务执行时间 p_j 非增排序,然后将排好序的任务序列依次置于 m 台机器上执行,若某台机器空闲,则将当前任务序列中的第 1 个任务安排在上面执行,直到所有任务均执行完毕为止.得到任务安排表$\sum_{\text{LPT}}$和相应的停机时间 C_{LPT}.

Step 2. 将任务集合中的任务任意排成一列,然后按照顺序将任务分配给一台机器,直到该机器上执行任务的总执行时间达到 $D=\max\{[(\sum_{j=1}^{n}p_j)+(m-1)\delta]/m,\max_{1\le j\le n}p_j+\delta\}$为止.此时若某任务只部分完成,则将该任务的未完成部分(包含中断时间代价δ)分配给下一台机器,再继续安排其他任务,直到将所有的任务安排完毕为止.然后,将每台机器第 1 个执行的任务(或任务中断部分)与它的最后一个执行的任务(或任务中断部分)对换,得到任务安排表$\sum_{\text{wrap}}$和相应的停机时间 C_{wrap}.

Step 3. 返回停机时间小的任务安排表$\sum$,停机时间 $C_{\text{LPT-wrap}}=\min\{C_{\text{wrap}},C_{\text{LPT}}\}$.

在 LPT-Swap 算法中,Step 1 的时间为 $O(n\log n)$,Step 2 的时间为 $O(n+m)$,所以整个算法的时间复杂度为 $O(n\log n+m)$.由于 Step 1 中要求对全部任务按照执行时间进行排序,所以该算法是一个脱线近似算法.

为了下面分析的方便,设对于 $P|ptmn(\delta)|C_{\max}$ 问题的任一实例 I,其最优停机时间为 $C^*_{\max}$,相应的任务中断次数最少的最优任务安排表为$\sum^*$;将实例 I 中的任务集合作为 $P||C_{\max}$ 问题的一个实例 I_0,记它的最优停机时间为 $C^*_{\max 0}$,相应的最优任务安排表为$\sum^*_0$.

因为$\sum^*$中任何一个任务在 $C^*_{\max}$ 时刻均已执行完毕,而同一个任务不能在不同的机器上同时执行,且每次中断都会增加一个额外的执行时间δ.因此,有下面的命题 2 成立.

命题 **2**. 若任务 J_j 在$\sum^*$中发生中断 k_j 次,有 $C^*_{\max}\ge p_j+k_j\delta$,其中 p_j 为任务 J_j 的执行时间.特别地,若任务 J_j 在$\sum^*$中发生中断,有 $C^*_{\max}\ge p_j+\delta$.

引理 **1**. 若$\delta>1/2\ C^*_{\max}$,$C_{\text{LPT}}\le 4/3\ C^*_{\max}$.

证明:当$\delta>1/2\ C^*_{\max}$时,若在$\sum^*$中任务 J_j 发生中断,由命题 1 知 J_j 的执行时间 $p_j>\delta$,于是有 $p_j+\delta>1/2\ C^*_{\max}+\delta>C^*_{\max}$,这与命题 2 的结果矛盾.所以由在$\sum^*$中没有任务发生中断可知,$\sum_{\text{LPT}}$对输入实例为 I_0 的 $P||C_{\max}$ 问题也是合法的,且有 $C^*_{\max}=C^*_{\max 0}$.又 LPT 算法对于 $P||C_{\max}$ 问题的近似度为 4/3[2],所以有 $C_{\text{LPT}}\le 4/3\ C^*_{\max 0}=4/3\ C^*_{\max}$. □

引理 **2**. $C_{\text{wrap}}-C^*_{\max}\le\delta$.

证明:这里先说明算法 LPT-Swap 的 Step2 可以得到一个合法的任务安排表.一方面,按照算法得到的任务安排表中至多有 $m-1$ 个任务发生中断,且每个发生中断的任务仅中断一次,所以需要总的执行时间不超过$\sum_{j=1}^{n}p_j+(m-1)\delta$,又因为每台机器的总执行时间均可以达到 D,由 $mD\ge\sum_{j=1}^{n}p_j+(m-1)\delta$知,所有的任务都可以在时间 D 内执行完毕;另一方面,每个发生中断的任务的两个执行部分分别在两台机器上首先执行和最后执行,而$D\ge\max_{1\le j\le n}p_j+\delta$,所以它们不会在一个时刻同时执行,并且算法也保证了后执行的任务中断部分的执行时间大于δ.由此可知,算法得到的是一个合法的任务安排表$\sum_{\text{wrap}}$,相应的停机时间 $C_{\text{wrap}}=D=\max\{[(\sum_{j=1}^{n}p_j)+(m-1)\delta]/m,\max_{1\le j\le n}p_j+\delta\}$.

(1) 若 $C_{\text{wrap}}=\max_{1\le j\le n}p_j+\delta$,由于$\sum^*$中在 $C^*_{\max}$ 时刻,任一任务均执行完毕,所以 $C^*_{\max}\ge\max_{1\le j\le n}p_j$,由此可得 $C_{\text{wrap}}=D\le C^*_{\max}+\delta$.

(2) 若 $C_{\text{wrap}}=[(\sum_{j=1}^{n}p_j)+(m-1)\delta]/m$,由于$\sum^*$中在 $C^*_{\max}$ 时刻,所有的任务均执行完毕,所以有 $C^*_{\max}\ge(\sum_{j=1}^{n}p_j)/m$ 成立,由此可得 $C_{\text{wrap}}=[(\sum_{1\le j\le n}p_j)+(m-1)\delta]/m<C^*_{\max}+\delta$.

由上面的(1)和(2)知,$C_{\text{wrap}}-C^*_{\max}\le\delta$成立. □

引理 **3**. 若$(1/\sqrt{6})C^*_{\max}<\delta\le(1/2)C^*_{\max}$,则 $C^*_{\max 0}\le C^*_{\max}+(\sqrt{6}-2)\delta$.

证明:$(1/\sqrt{6})C^*_{\max}<\delta\le(1/2)C^*_{\max}$,即 $2\delta\le C^*_{\max}<\sqrt{6}\,\delta$.下面对任务中断次数最少的最优任务安排表$\Sigma^*$中的任务中断情况进行考察.首先,若在$\Sigma^*$中任务 J_j 发生中断,由命题 2 知 $C^*_{\max}\ge p_j+\delta$,可得所有在Σ^*中发生中断的任务的执行时间为

$$p_j \le C^*_{\max}-\delta<(\sqrt{6}-1)\delta\ . \tag{*}$$

设在最优任务安排表Σ^*中任务 J_j 被中断了 k_j 次.由命题 1 知在Σ^*中发生中断的任务执行时间 $p_j>\delta$,由命题 2 知 $C^*_{\max}\ge p_j+k_j\delta$,所以 $C^*_{\max}\ge(1+k_j)\delta$.又由 $C^*_{\max}<\sqrt{6}\,\delta$,知 $k_j=0$ 或 1,即Σ^*中的所有任务至多中断一次.设执行时间为 p_j 的任务 J_j 产生中断,分成两个部分执行,执行时间分别为 p_j'和 p_j'',有 $p_j'+p_j''=p_j+\delta$.若 $p_j'\le\delta$或者 $p_j''\le\delta$,则 $p_j''\ge p_j$ 或者 $p_j'\ge p_j$.此时,可以对任务安排表Σ^*进行调整,使得任务 J_j 不发生中断,在p_j'或p_j''执行时间内完成,其余任务的执行安排不变.这样,中断次数减少而停机时间没有增加,这与Σ^*中任务中断次数最少矛盾.所以 $p_j'>\delta$且 $p_j''>\delta$,即所有中断产生的部分执行时间均大于δ.

又由于Σ^*是任务中断次数最少的最优任务安排表,且 $C^*_{\max}<\sqrt{6}\,\delta<3\delta$,所以一个任务中断后产生的两个部分不会在同一台机器上执行,Σ^*中每台机器上至多也只能执行两个中断产生的执行部分.

这样,可以将Σ^*中的任务中断情况用一单图 G 表示.图 G 中每个顶点代表一台机器,若一任务发生中断在两台机器上执行,则图 G 中的两台机器相对应的顶点有一条边相连.由于每台机器上至多只能执行两个中断产生的部分,因此图 G 中所有顶点的度至多为 2.

因为单图 G 中的顶点最大度数小于等于 2,所以 G 的每个连通片只可能是圈或者路径.若图 G 中存在长度为 k 的圈 $H(k\ge2)$,即在Σ^*中,存在 k 个发生中断的任务并且中断后的部分在 k 台机器上执行.此时调整任务安排表,取消这 k 个任务的中断,直接将这 k 个任务分别置于 k 台机器上执行.取消中断后的每台机器,相当于减少了两个执行时间大于δ的中断产生的执行部分,加上一个执行时间小于$(\sqrt{6}-1)\delta$的任务(由(*)式),停机时间会减少.这与Σ^*是最优的任务安排表矛盾.所以在图 G 中没有为圈的连通片,它所有的连通片都是一条简单路径.

(1) 对应于图 G 中连通片为长度为 k 的路径$(k\ge2)$,即 k 个任务发生中断,在 $k+1$ 台机器上执行,且首尾两台机器上只执行一个中断产生的部分.不妨设任务 $J_1,J_2,\ldots,J_k$ 发生中断,在机器 $M_1,\ldots,M_{k+1}$ 上执行.J_j 中断为两个执行部分 J_j'和 $J_j''(1\le j\le k)$,J_1'在 M_1 上执行,J_k''在 M_{k+1} 上执行,J''_{i-1}和 J_i'在机器 $M_i(2\le i\le k)$上执行.现在调整任务安排表,取消原有的 k 次中断.未发生中断的任务依然在原机器上执行,将发生中断的任务 J_j 置于 M_j 上执行$(1\le j\le k)$.

下面考虑取消中断后,这 $k+1$ 台机器停机时间的变化情况.对于机器 $M_i(2\le i\le k)$,相当于去掉两个执行时间大于δ的中断执行部分 J''_{i-1}和 J_i',加上 1 个执行时间小于$(\sqrt{6}-1)\delta$的任务 J_i(由(*)式),停机时间会减少;机器 M_{k+1} 上仅仅去掉了一个中断执行部分 J_k'',停机时间是减少的;最后考虑机器 M_1 的停机时间变化情况.M_1 上原来处理 1 个执行时间 $p'_1>\delta$的中断产生的执行部分 J_1',现在将整个未中断的任务 J_1 在其上执行,该机器的停机时间增加了 $p_1-p'_1$,即 $p''_1-\delta$.又原先执行 J_1''的机器 M_2 上除 J_1''外,还有另一个执行时间大于δ的中断产生的执行部分 J_2',于是,由 $C^*_{\max}<\sqrt{6}\,\delta$知,$p''_1<(\sqrt{6}-1)\delta$.所以,机器 M_1 的停机时间增加值小于$(\sqrt{6}-2)\delta$,即这 $k+1$ 台机器总的停机时间的增加值小于$(\sqrt{6}-2)\delta$.

(2) 对应于图 G 中的连通片为长度为 1 的路径,即 1 个任务发生中断在两台机器上执行,且这两台机器只执行一个任务中断后的执行部分.设中断的任务为 J_j,其执行时间为 p_j,两个中断后部分的执行时间为 p_j'和 p_j'',分别在机器 M_i和 M_{i+1} 上执行,由前面的分析知 $p_j'+p_j''=p_j+\delta$,又由(*)式知,$p_j'+p_j''\le\sqrt{6}\,\delta$,所以 p_j'和 p_j''中必有一个不大于 $\sqrt{6}/2\delta$,不妨设 $p_j'\le\sqrt{6}/2\delta$.现取消这次中断,将整个任务置于机器 M_{i+1} 上执行,机器 M_i 的停机时间减少 p_j',M_{i+1} 停机时间增加了 $p_j-p_j''=p_j'-\delta\le(\sqrt{6}/2-1)\delta<(\sqrt{6}-2)\delta$.

由上述(1)和(2)的讨论知,将Σ^*中任务中断全部取消,停机时间最多增加$(\sqrt{6}-2)\delta$,而 $C^*_{\max\ 0}$ 是当所有任务均不中断时的最优停机时间,所以 $C^*_{\max\ 0}\le C^*_{\max}+(\sqrt{6}-2)\delta$成立. □

定理 **2**. $C_{\text{LPT-Wrap}}\le(1+1/\sqrt{6})C^*_{\max}\approx1.40825\ C^*_{\max}$.

证明:当$\delta>1/2\ C^*_{\max}$ 时,由引理 1 知 $C_{\text{LPT}}\le4/3\ C^*_{\max}$;当$\delta\le1/\sqrt{6}\ C^*_{\max}$ 时,由引理 2 知 $C_{\text{Wrap}}\le\delta+C^*_{\max}\le(1+1/\sqrt{6})C^*_{\max}$.下面考虑 $1/\sqrt{6}\ C^*_{\max}<\delta\le1/2\ C^*_{\max}$ 时的情况.

考虑Σ_{LPT}中的最后一个执行完毕的任务 J_k,其执行时间为 p_k.

(1) 若 $p_k>1/3\ C^*_{\max 0}$,考察没有任务中断的最优的任务安排表Σ^*_0,可知至多只有 $2m$ 个执行时间大于等于 p_k 的任务存在.这些任务在Σ_{LPT}中均在 J_k 之前执行,不妨记为 $J_1,J_2,\ldots,J_{k-1}(p_1\geq p_2\geq\ldots\geq p_{k-1}\geq p_k)$,且$\Sigma_{\mathrm{LPT}}$中每台机器至多执行其中的两个.易见,这不多于 $2m$ 个的任务在Σ_{LPT}中的次序安排就是最优的次序安排(J_1 和 J_k,J_2 和 J_{k-1},…,两两在一台机器上执行),可知 $C_{\mathrm{LPT}}\leq C^*_{\max 0}$,故有 $C_{\mathrm{LPT}}=C^*_{\max 0}$.由引理 3 及$\delta\leq 1/2\ C^*_{\max}$ 可得:$C_{\mathrm{LPT}}=C^*_{\max 0}\leq C^*_{\max}+(\sqrt{6}-2)\delta\leq C^*_{\max}+(\sqrt{6}-2)(1/2\ C^*_{\max})=\sqrt{6}/2\ C^*_{\max}$.

(2) 若 $p_k\leq 1/3\ C^*_{\max 0}$,则因为在任务 J_k 执行之前所有的机器均没有空闲,所以 J_k 开始执行的时刻 $s_k<(\sum^n_{j=1}p_j)/m\leq C^*_{\max}$,这样有 $C_{\mathrm{LPT}}=s_k+p_k\leq(\sum^n_{j=1}p_j)/m+p_k\leq C^*_{\max}+1/3\ C^*_{\max}$.再由引理 3 及$\delta\leq 1/2\ C^*_{\max}$ 可得:$C_{\mathrm{LPT}}\leq C^*_{\max}+1/3\ C^*_{\max}+(\sqrt{6}-2)\delta]=4/3\ C^*_{\max}+(\sqrt{6}-2)/3\delta\leq(4/3+(\sqrt{6}-2)/6)\ C^*_{\max}=(1+\sqrt{6}/6)C^*_{\max}$.

综上,$C_{\text{LPT-Wrap}}=\min\{C_{\mathrm{wrap}},C_{\mathrm{LPT}}\}\leq(1+1/\sqrt{6})C^*_{\max}\approx 1.40825\ C^*_{\max}$ 成立. □

由定理 2 知,算法 LPT-Swap 的近似比小于等于 1.408 25.

3 在线算法

LS 算法是调度问题中的常见算法.它按照所给任务序列,若某台机器空闲,即将当前任务表中的第 1 个任务安排在上面执行,直到所有任务均执行完毕为止.注意到该算法的时间复杂度为 $O(n)$,且是一个在线算法.

定理 **3**. 对于 $P|ptmn(\delta)|C_{\max}$,LS 算法的竞争比为 2.

证明:考虑 LS 算法得到的任务安排表Σ_{LS}中的最后执行完的那个任务 J_k,设其执行时间为 p_k.由于任务 J_k 执行前没有机器空闲,所以 J_k 开始执行的时间 $s_k<(\sum^n_{j=1}p_j)/m$.又在最优任务安排表中,任务 J_k 在 $C^*_{\max}$ 时刻已执行完毕,可知 $C^*_{\max}\geq p_k$,所以,$C_{\mathrm{LS}}=s_k+p_k<(\sum^n_{j=1}p_j)/m+p_k<C^*_{\max}+C^*_{\max}=2\ C^*_{\max}$.

对于实例 $I:\delta=m$,任务序列由 $m(m-1)$个执行时间为 1 的任务和一个执行时间为 m 的任务组成.易知 $C_{\mathrm{LS}}(I)=2m-1,C^*_{\max}(I)=m$.可见 LS 算法的竞争比为 2. □

4 总 结

本文提出了 $P|ptmn(\delta)|C_{\max}$ 问题,它考虑任务中断时的时间代价,在实际的各种调度应用问题中,都存在这种中断的时间代价.文中给出了该问题的一个近似比不大于 1.408 25 的脱线近似算法 LPT-Swap,接着说明 LS 算法作为该问题的在线近似算法的竞争比为 2.寻找近似比小于 2 的在线近似算法是我们下一步的研究课题.

References:

[1] Hall, L.A. Approximation algorithms for scheduling. In: Hochbaum, D., ed., Approximation Algorithms for NP-Hard Problems. Boston: PWS, 1996. 1~45.

[2] Karger, D., Stein, C., Wein, J. Scheduling algorithms. In: Atallah, M.J., ed., Handbook on Algorithms and Theory of Computation. CRC Press, 1998.

[3] Graham, R.L. Bounds for certain multiprocessing anomalies. Bell System Technical Journal, 1966,45:1563~1581.

[4] Engels, D.W., Feldman, J., Karger, D.R. Parallel processor schedluing with delay constraints. In: Proceeding of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms. ACM Press, 2001. 577~585.

[5] Pressman, R.S. Software Engineering: a Practitioner's Approach. McGraw-Hill, 1997.

[6] Attiya, H., Welch, J. Distributed Computing: Fundamentals, Simulations and Advanced Topics. London: McGraw-Hill, 1998.

[7] Green, P.E. Fiber-Optic Networks. Cambridge, MA: Prentice-Hall, 1992.

[8] Deng, Xiao-tie, Gu, Nian, Brecht, T., *et al*. Preemptive scheduling of parallel jobs on multiprocessors. In: Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms. ACM Press, 1996. 159~167.

[9] Goemans, M., Wein, J., Williamson, D.P. A 1.47-approximation algorithm for a preemptive single-machine scheduling problem. Operations Research Letters, 2000,26:149~154.

[10] Garey, M.R., Johnson, D.S. Computers and Intractability: A Guild to the Theory of NP-Completeness. New York: W.H. Freeman and Company, 1979.

[11] Sgall, J. On-Line scheduling——a survey. In: Fiat, A., Woeginger, G.J., eds, Online Algorithms: the State of the Art. Lecture Notes in Computer Science 1442, Springer-Verlag, 1998. 196~231.

Preemptive Scheduling Problem with Interrupted Time Cost on Identical Parallel Machines*

SUN Guang-zhong[1,2], CHEN Guo-liang[1,2], XU Yin-long[1,2], GU Jun[3]

[1](*National High Performance Computing Center*, *Hefei* 230027, *China*);

[2](*Department of Computer Science and Technology*, *University of Science and Technology of China*, *Hefei* 230027, *China*);

[3](*Department of Computer Science and Technology*, *HongKong University of Science and Technology*, *HongKong*, *China*)

E-mail: sungz@mail.ustc.edu.cn

http://www.ustc.edu.cn

Abstract: A preemptive scheduling problem $P|ptmn(\delta)|C_{\max}$ is proposed in this paper, in which an additional time cost δ is needed if a job is interrupted once. The problem has wide applications such as job allocation in projects, distributed computing and communication in networks. It is proved that $P|ptmn(\delta)|C_{\max}$ is an NP-hard problem. An off-line approximation algorithm LPT-Wrap with time complexity of $O(n\log n+m)$ and a performance ratio no more than 1.408 25 is presented. The on-line property of $P|ptmn(\delta)|C_{\max}$ is also studied and an on-line algorithm with linearity time complexity and a competitive ratio of 2 is proposed.

Key words: scheduling problem; preemption; combinatorial optimization; time cost; NP-hard; approximation algorithm; approximation ratio

* Received April 29, 2001; accepted July 12, 2001

Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1998030403