一种改进的 RM 可调度性判定算法*

刘军祥[1,2+], 王永吉[1], Matthew Cartmell[3]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](中国科学院 研究生院,北京 100039)

[3](Department of Mechanical Engineering, University of Glasgow, G12 8QQ, UK)

An Improved Rate Monotonic Schedulability Test Algorithm

LIU Jun-Xiang[1,2+], WANG Yong-Ji[1], Matthew Cartmell[3]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate School, The Chinese Academy of Sciences, Beijing 100039, China)

[3](Department of Mechanical Engineering, University of Glasgow, G12 8QQ, UK)

+ Corresponding author: Phn: +86-10-82620803 ext 812, E-mail: junxiang03@iscas.cn, http://www.ios.ac.cn

Received 2003-11-28; Accepted 2004-05-09

Liu JX, Wang YJ, Cartmell M. An improved rate monotonic schedulability test algorithm. *Journal of Software*, 2005,16(1):89−100. http://www.jos.org.cn/1000-9825/16/89.htm

Abstract: One of the key issues in real-time theory is the schedulable analysis of a given task set with fixed priority. All the proposed schedulability test methods can be classified into two types: polynomial time methods and exact methods. Polynomial time methods use a sufficient schedulability condition, and several least upper bounds of the processor utilization under ideal assumptions have been developed. Exact methods based on the necessary and sufficient schedulability condition guarantee that the test result for every task set is correct. However, the pseudo-polynomial complexity of the exact tests is too complex to be executed online for large task sets. This paper presents a novel ISTA (improved schedulability test algorithm) for analyzing the schedulability of periodic task sets under Rate Monotonic priority assignment. A pruning theorem for the Task_*i* space is derived, the schedulability correlation among tasks and its effect on the schedulability test are investigated, and the related theorems are proven. Based on the above results, a new improved pseudo-polynomial schedulability test algorithm is developed, and

* Supported by the National Natural Science Foundation of China under Grant No.60373053 (国家自然科学基金); the Hundred Talents of the Chinese Academy of Sciences (中国科学院"百人计划"); the Chinese Academy of Sciences and the Royal Society of the United Kingdom under Grant Nos.20030389, 20032006 (中国科学院与英国皇家学会联合资助项目); the State Education Ministry Scientific Research Foundation for the Returned Overseas Chinese Scholars under Grant No.[2003]406 (留学回国人员科研启动基金资助项目); the National High-Tech Research and Development Plan of China under Grant No.2003AA1Z2220 (国家高技术研究发展计划(863))

作者简介: **刘军祥**(1975−),男,湖北潜江人,博士生,主要研究领域为实时系统,流媒体传输协议;**王永吉**(1962−),男,博士,研究员,博士生导师,主要研究领域为实时系统,网络优化,智能软件工程,优化理论,机器人,控制理论;**Matthew Cartmell**(1958−),男,教授,博士生导师,主要研究领域为非线性结构动力学,计算技术.

several comparative simulation experiments among the three methods are conducted. Extensive simulations show that the average schedulability test performance as a function of number of the tasks can be significantly improved.

Key words: real-time system; schedulability; real-time scheduling; RM (rate monotonic) algorithm; hard real-time system

摘 要: 固定优先级任务可调度性判定是实时系统调度理论研究的核心问题之一.目前已有的各种判定方法可归结为两大类:多项式时间调度判定和确切性判定.多项式时间调度判定通常采用调度充分条件来进行,为此,许多理想条件下基于 RM(rate monotonic)调度算法的 CPU 利用率最小上界被提了出来.确切性判定利用 RM 调度的充要条件,保证任何任务集均可被判定,并且判定结果是确切的.但是由于时间复杂度较差,确切性判定方法难以实现在线分析.提出了一种改进的 RM 可调度性判定方法(improved schedulability test algorithm,简称 ISTA).首先介绍了任务调度空间这一概念,并提出了二叉树表示,然后进一步提出了相关的剪枝理论.在此基础上,研究了任务之间可调度性的相关性及其对判定任务集可调度性的影响,提出并证明了相关的定理.最后基于提出的定理,给出了一种改进的伪多项式时间可调度性判定算法,并与已有的判定方法进行了比较.仿真结果表明,该算法平均性能作为任务集内任务个数的函数具有显著提高.

关键词: 实时系统;调度;实时调度;RM 算法;硬实时系统

中图法分类号: TP316 **文献标识码**: A

与非实时系统相比,实时系统的一个显著特点是它们试图同时实现计算在逻辑上和时间上的正确性.在实时系统中,计算的正确性不仅取决于计算的逻辑结果,也取决于结果产生的时间.1973 年,Liu 和 Layland 提出了一种适用于可抢占的硬实时周期性任务调度的静态优先级调度算法——速率单调(rate monotonic,简称 RM)调度算法,并对其可调度性判定问题进行了研究[1].所谓 RM 调度,就是为每一个周期任务指定一个固定的优先级,该优先级按照任务周期的长短顺序排列,任务周期越短,其优先级越高,调度总是试图最先运行周期最短的任务.Liu 指出了 RM 算法的最优性:一切可以用其他静态优先级调度算法调度的任务集均可以被 RM 调度.Liu 和 Layland 给出了 RM 可调度性判定的一个充分条件,这个充分条件就是后面所要讨论的 CPU 利用率最小上界.

任务可调度性判定是实时系统调度理论研究的核心问题之一,现已经发表了大量关于单调速率(RM)和各种扩展情况下的调度算法以及实时任务在这些算法下的可调度性判定研究的文章.目前已有的各种判定方法可归纳为两大类:多项式时间调度判定和确切性判定.多项式时间调度判定通常采用调度充分条件来进行,学者们研究了最坏情况下的调度特性,提出了一系列的CPU利用率最小上界.对一个任意大小的任务集,如果它总的 CPU 利用率小于或等于这个最小上界,则此任务集是可以调度的.基于此出发点,文献[2]提出了另一个改进的CPU利用率最小上界,文献[3,4]提出了一个双曲线的CPU利用率最小上界,文献[5]提出了一种多项式时间判定算法,文献[6]利用线性规划来求解任务执行时间不确切条件下的 CPU 利用率最小上界.然而,对于一个含有 n 个任务的任务集来说,如果CPU利用率大于CPU利用率最小上界,则用多项式时间调度判定无法判断该任务集合是否可调度,我们把这种特性称为判定的不确定性.这是因为判定条件 CPU 利用率最小上界是在最坏情况下考察其可调度性而得出的,它只是一个充分但不必要条件.因此,多项式时间调度判定常被称为“悲观的”,这是其不足之处.

确切性判定利用充要条件进行判定.文献[7]给出了 RM 可调度的充分必要条件,文献[8]研究了 RM 可调度在各种扩展条件下的充分必要条件.确切性判定保证了判定结果的确定性,任何任务集均可被判定,并且判定结果是确切的.但是,确切性判定方法的算法的时间复杂度是假多项式的,不能被用来进行大任务集的在线判定及准入控制.文献[9]提出了一种新颖的方法来分析 RM 调度下周期性任务的可调度性,并给出了具体的HET(hyperplanes δ-exact test)可调度性判定算法,是目前所能搜集到的最好结果.

对实时单调速率及其扩展算法的可调度性判定,文献[10]给出了详细的介绍.本文提出了一种新的改进的RM 可调度性判定算法,并与现有的同类算法进行了比较.

本文第 1 节具体介绍与本文相关的在实时系统领域的研究结果,这部分的介绍是后边论文工作的必要数

学基础.本文的主要工作由第 2~4 节组成.第 2 节提出任务调度空间的概念,并给出任务调度空间的二叉树表示.利用二叉树来组织调度空间元素,可以在二叉树成熟的理论基础上进行任务调度空间的剪枝理论研究,同时方便了各任务间可调度相关性研究.第 3 节给出任务间可调度相关性的判定定理、实例及改进的 RM 可调度性判定算法.第 4 节给出具体的仿真对比实验,对 3 种不同算法的性能进行了比较分析.结果表明,改进后的算法具有运行更快、结果更准确的优点.最后指出进一步的研究方向.

1 研究基础

1.1 RM调度模型

假设 $\tau=\{\tau_1,...,\tau_n\}$ 是一个含有 n 个周期性任务的集合,集合中的任务用 $\tau_i=(C_i,T_i)$ $(i=1,...,n)$ 来表示,其中,T_i 表示 τ_i 的周期,C_i 表示 τ_i 的最坏执行时间,且有 $0\le C_i\le T_i$ $(i=1,...,n)$.$U_i=C_i/T_i$ 表示 τ_i 的 CPU 利用率,整个任务集的 CPU 利用率定义为 $U=\sum_{i=1}^{n}U_i$.对任务集 $\tau=\{\tau_1,...,\tau_n\}$,RM 算法中通常假定: $T_1<T_2<...<T_n$,意味着,若 $1\le i<j\le n$,则 τ_i 的优先级高于 τ_j ,也就是任务集中任务按照优先级由高到低的顺序排列.需要特别提出的是,理想的 RM 调度模型是指隐含截止期(即任务 τ_i 的截止期限 D_i 等于 T_i)的同步单机实时系统,具体的 7 个基本假设请参考文献[10].若一个任务集利用 RM 算法进行调度是可行的,则称该任务集 RM 可调度.反之,则称 RM 不可调度.

1.2 CPU利用率最小上界

CPU 利用率最小上界意味着对一个包含任意数量,每个任务取任意周期的任务集,如果其总的 CPU 利用率不大于这个最小上界,则此任务集是可以调度的.在第 1.1 节基本假设的基础上,文献[1~4]对 RM 算法的可调度性判定进行了研究,提出了多个 RM 算法下的 CPU 利用率最小上界.然而,这些最小上界来自于最坏情况下的估计.使用上述 CPU 利用率最小上界进行调度判定,具有多项式时间复杂度,但却带有很大的悲观性.如图 1 所示.

Fig.1 Physical meaning of schedulability test with CPU utilization least up bounds

图 1 利用 CPU 利用率最小上界进行可调度性判定的物理意义

图 1 中 3 种情况所代表的物理意义如下:

(1) 整个 CPU 利用率小于 bound,任务集是 RM 可调度的;

(2) 整个 CPU 利用率大于 1,任务集是 RM 不可调度的;

(3) 整个 CPU 利用率介于 bound 和 1 之间,无法判定.

可见,对于图中处于灰色区域内的任务集,有可能可调度,但利用 CPU 利用率最小上界来判定却无能为力.

1.3 RM调度的充要条件

由于利用 CPU 利用率最小上界进行可调度性判定是一种悲观的判定,文献[7]提出了 RM 可调度的充要条件.

定理 1. 对任务集 $\tau=\{\tau_1,...,\tau_n\}$,有

(1) 任务 τ_i 是可调度的当且仅当

$$L_i=\min_{t\in S_i}\frac{\lfloor t/T_j\rfloor C_j}{t}\le 1 \tag{1}$$

这里, $S_i=\{rT_j\mid j=1,...,i,r=1,...,\lfloor T_i/T_j\rfloor\}$.

(2) 整个任务集是可调度的当且仅当

$$\max_{i=1,...,n} L_i \le 1 \tag{2}$$

然而,利用 RM 可调度的充要条件来进行判定虽然能够保证所有的任务集均可以被判定,而且判定的结果是确切的,但是它是一种假多项式时间的算法,不适合实时在线判定.文献[9]提出了一种新颖的分析周期性任务 RM 可调度性的方法.它把任务集 $\tau=\{\tau_1,...,\tau_n\}$ 看做是一个由任务参数 C_i 构成的 n-维空间中的点,那么可调度性判定则可以看做是检查此点是否在某一个 RM 可调度的区域 M_n 内. M_n 的定义如下:

$$M_n(T_1,...,T_n)=\{(C_1,...,C_n)\in R^n,\forall i,C_i\ge 0\mid \tau \text{ is schedulable by } RM\} \tag{3}$$

在上式中,周期 T_i 是参数,各个任务执行时间 C_i 是自由变量.因此,我们可以得到一组关于 C_i 变量的约束条件,这组约束条件同时也是各个任务周期 T_i 的函数.

事实上,尽管没有明确表示,在不等式(1)中各个式子的关系是一种逻辑“或”的关系,而在不等式(2)中各个式子的关系是一种逻辑“与”的关系,如果用符号“∨”来表示逻辑“或”操作,用符号“∧”表示逻辑“与”操作,这样定理 1 可以表示为

$$\max_{i=1,...,n}\min_{t\in S_i}\sum_{j=1}^{i}\left\lceil t/T_j\right\rceil C_j\le t \Leftrightarrow \bigwedge_{i=1,...,n}\min_{t\in S_i}\sum_{j=1}^{i}\left\lceil t/T_j\right\rceil C_j\le t \Leftrightarrow \bigwedge_{i=1,...,n}\bigvee_{t\in S_i}\sum_{j=1}^{i}\left\lceil t/T_j\right\rceil C_j\le t \tag{4}$$

结合式(3)中 M_n 的定义与式(4),我们可以给出如下定理.

定理 2. RM 可调度的区域 M_n 为

$$M_n(T_1,...,T_n)=\{(C_1,...,C_n)\in R^n,\forall i,C_i\ge 0\mid \bigwedge_{i=1,...,n}\bigvee_{t\in S_i}\sum_{j=1}^{i}\left\lceil t/T_j\right\rceil C_j\le t\} \tag{5}$$

这里, $S_i=\{rT_j\mid j=1,...,i,r=1,...,\lfloor T_i/T_j\rfloor\}$.

为方便理解,我们给出例 1.

例 1:设有一任务集,见表 1,先让我们看看每个任务的 RM 可调度性.用代数式表示,我们有

Table 1 A periodic task set

表 1 周期性任务

i	T_i	S_i
1	3	{3}
2	8	{3,6,8}
3	20	{3,6,8,9,12,15,16,18,20}}

- 对任务 τ_1,如果 $C_1\le 3$,则 τ_1 RM 可调度.
- 对任务 τ_2,如果不等式(6)成立,则 τ_2 RM 可调度.

$$C_1+C_2\le 3\vee 2C_1+C_2\le 6\vee 3C_1+C_2\le 8 \tag{6}$$

- 对任务 τ_3,如果不等式(7)成立,则 τ_3 RM 可调度.

$$\begin{aligned}&C_1+C_2+C_3\le 3\vee 2C_1+C_2+C_3\le 6\vee 3C_1+C_2+C_3\le 8\vee 3C_1+2C_2+C_3\le 9\vee 4C_1+2C_2+C_3\le\\ &12\vee 5C_1+2C_2+C_3\le 15\vee 6C_1+2C_2+C_3\le 16\vee 6C_1+3C_2+C_3\le 18\vee 7C_1+3C_2+C_3\le 20\end{aligned} \tag{7}$$

采用文献[11]的方法,我们可以得到一种简洁而直观的表示.用 S_{ij} 表示集合 S_i 的第 j 个元素,并且假设

$$h_{ij}\Leftrightarrow \sum_{k=1}^{i}\left\lceil S_{ij}/T_k\right\rceil C_k\le S_{ij}.$$

那么不等式(6)和(7)分别等价于如下具有逻辑关系“或”的不等式约束:

$$h_{21}\vee h_{22}\vee h_{23} \tag{8}$$

$$h_{31}\vee h_{32}\vee h_{33}\vee h_{34}\vee h_{35}\vee h_{36}\vee h_{37}\vee h_{38}\vee h_{39} \tag{9}$$

从文献[9]中我们已经得知如下结论:

(1) RM 可调度的区域 M_n 由 h_{ij} 限定,但是在不等式(8)和(9)中,许多 h_{ij} 实际上是无用的;

(2) 阻碍利用充要条件进行在线调度判定的一个很重要的因素就在于 h_{ij} 的数量比较大,特别是对于任务数很多的任务集.所有 h_{ij} 的数量等于所有 S_i 中元素个数的总和;

(3) 实际上通过消除一些具有逻辑“或”关系的 h_{ij},我们可以得到一个与 M_n 等价的子集.相比较而言,在这个子集上进行可调度性判定具有更简单的时间复杂性.

从这些结论出发,Bini[9]给出并证明了如下定理.

定理 3. RM 可调度的区域 M_n 可以由简化后的区域定义如下:

$$M_n(T_1,...,T_n)=\{(C_1,...,C_n)\in R^n,\forall i,C_i\ge 0 \mid \bigwedge_{i=1,...,n}\ \bigvee_{t\in p_{i-1}(T_i)}\sum_{j=1}^{i}\left\lceil t/T_j\right\rceil C_j\le t\}.$$

这里, $p_{i-1}(T_i)$ 由循环表达式(10)定义:

$$p_0(t)=\{t\},\ p_{i-1}(T_i)=p_{i-2}(\lfloor T_i/T_{i-1}\rfloor T_{i-1})\cup p_{i-2}(T_i) \tag{10}$$

同时不失一般性,对任意的时间间隔[0,*b*],有

$$p_0(t)=\{t\},\ \ p_i(b)=p_{i-1}(\lfloor b/T_i\rfloor T_i)\cup p_{i-1}(b) \tag{11}$$

文献[9]证明了集合 S_i 等价于 $p_{i-1}(T_i)$,即 $S_i\Leftrightarrow p_{i-1}(T_i)$.

对于例 1,我们有

$$S_3\Leftrightarrow p_2(T_3)=p_2(20)=p_1(16)\cup p_1(20)=p_0(15)\cup p_0(16)\cup p_0(18)\cup p_0(20)=\{15,16,18,20\},$$
$$S_2\Leftrightarrow p_1(T_2)=p_1(8)=p_0(6)\cup p_0(8)=\{6,8\},$$
$$S_1\Leftrightarrow p_0(T_1)=p_0(3)=\{3\}.$$

与此同时,设 $S_{36}=15,S_{37}=16,S_{38}=18,S_{39}=20$,式(9)等价于

$$h_{36}\vee h_{37}\vee h_{38}\vee h_{39} \tag{12}$$

定理 3 通过减少充要条件中不等式的数目,显著地增加了利用充要条件进行可调度性判定算法的性能.文献[9]中提出了 HET 算法.尽管性能显著提高,但依然存在如下不足:

(1) 从式(10)可以看出,在 $p_{i-1}(T_i)$ 计算的过程中,可能存在大量相同的元素,这增加了算法不必要的计算开销.

(2) 定理 3 主要用来减少具有逻辑“或”关系间的处理,具有逻辑“与”的任务之间可调度相关性研究则被忽略了.

(3) 当与任务集内某一个任务具有相同周期的任务加入到任务集中时,HET 算法失效.

针对上述不足,本文将对其加以改进.

2 任务调度空间及其二叉树表示

2.1 二叉树基本理论

二叉树(binary tree)是 $n(n\ge 0)$ 个结点的有限集,它或者是空集 $n=0$,或者由一个根结点及两棵互不相交的、分别称作这个根的左子树和右子树的二叉树组成.二叉树中,每个结点最多只能有两棵子树,并且有左右之分.没有左右子树的结点称为叶结点.组成树的各结点的最大层次,称为树的深度.树形结构是一类重要的非线性结构,而二叉树是树形结构的一个重要类型.许多实际问题抽象出来的数据结构往往是二叉树的形式,即使是一般的树也能简单地转换为二叉树,而且二叉树的存储结构及其算法都较为简单,因此二叉树显得特别重要.

遍历是二叉树上最重要的运算之一,是二叉树上进行其他运算的基础.所谓遍历(traversal)是指沿着某条搜索路线,依次对树中每个结点均做 1 次且仅做 1 次访问.访问结点所做的操作依赖于具体的应用问题.通常根据访问结点操作发生位置来命名,有 3 种遍历算法:前序遍历、中序遍历和后序遍历.本文仅考虑前序遍历的递归算法定义,算法的复杂度为 $o(n)$.

若二叉树非空,则依次执行如下操作:

(1) 访问根结点;

(2) 遍历左子树;

(3) 遍历右子树.

使用二叉树来组织信息是一种广泛使用的技术,由于非本文的重点,有关二叉树更多的理论介绍,请参考文献[12]及有关书目.

2.2 任务调度空间及其二叉树表示

为便于以后的问题描述,首先给出如下定义:

定义 1. 称 $p_{i-1}(T_i)$ 为任务 τ_i 的 RM 调度空间.

实时在线可调度性判定时,根据理想的RM调度模型假设(A5),计算任务RM调度空间的时间开销是可以忽略的.对式(10)或式(11)采用递归算法,然而由于任务 RM 调度空间有其自身的特点,性能并不是很好,它需要我们给出一定的优化方法来减少计算开销.对式(11)

$$p_0(t)=\{t\},\quad p_i(b)=p_{i-1}(\lfloor b/T_i\rfloor T_i)\cup p_{i-1}(b).$$

如果把 $p_i(b)$ 看做是 $p_{i-1}(\lfloor b/T_i\rfloor T_i)$ 和 $p_{i-1}(b)$ 的父结点,并且 $p_{i-1}(\lfloor b/T_i\rfloor T_i)$ 是它的左子结点, $p_{i-1}(b)$ 是它的右子结点,那么 $p_i(b)$ 可以表示为一棵二叉树.

图 2 中,结点 $p_j(t)$ 由带有 t 的圆圈表示,左边的数字表示树中各结点所在层(深度),右边的列表示与每个结点深度相关联的任务周期. T_i 越小的任务所对应的结点深度越大,周期最短的任务在树的叶结点层.为了符号的方便表示,我们用结点 t 来意指结点 $p_j(t)$.由图 2 可见, $p'=\lfloor b/T_i\rfloor T_i, p=b$.如例 1 所示,例 1 中 $p_2(T_3)$ 的二叉树表示如图 3 所示.当结点深度为 2 时,相关的任务周期为 T_2 ;当结点深度为 3 时,相关的任务周期为 T_1 .

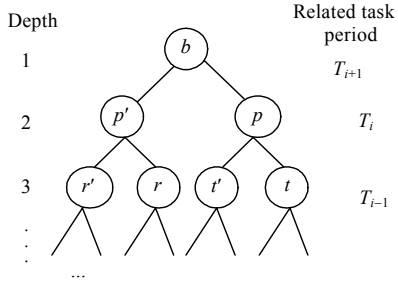


Fig.2 Bintree expression of RM schedulability space $p_i(b)$
图 2 任务 RM 调度空间 $p_i(b)$ 的二叉树

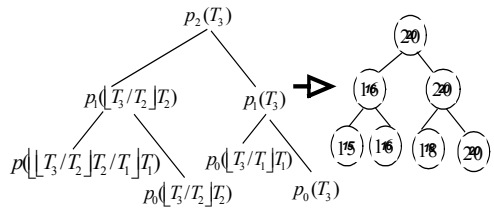


Fig.3 Bintree expression of $p_2(T_3)$ in Example 1
图 3 例 1 中 $p_2(T_3)$ 的二叉树表示

结合式(12),二叉树表示实际上就是用叶子结点来表示具有逻辑"或"关系的一组约束条件.更进一步地,我们给出如下定义.

定义 2. 前序遍历二叉树 $p_i(b)$,如果具有同样深度的结点 t 在结点 t' 之后被搜索到,则称 t 晚于结点 t' ,或者说 t' 早于 t .

2.3 剪枝理论

定理 4. 设 n_1,n_2 是二叉树 $p_i(b)$ 中具有相同深度的任意两个结点,并且结点 n_2 要晚于结点 n_1 ,那么当且仅当 $n_2>n_1$ 时,结点 n_2 才有意义.否则,分枝 n_2 可以被剪切掉.

证明:有两种情况存在:一是结点 n_1,n_2 有相同的父结点,二是 n_1,n_2 的父结点不同.当结点 n_1,n_2 有相同的父结点时,以图2为例,设 $n_1=t', n_2=t$, n_1,n_2 的父结点是 p ,根据定义则有: $t=p, t'=\lfloor p/T_{i-1}\rfloor T_{i-1}$.容易得知, $t\geq t'$, 当 $t=t'$ 时,分枝 t 可被剪切掉,定理成立.综合两种情况,对结点深度 d 采用数学归纳法来证明定理 4.

(1) 当结点深度为 2 时,属于第 1 种情况,得证.

(2) 当 $d=n$ 时,假设对所有的具有同一树深度的任意两结点 n_1,n_2,n_2 晚于 n_1 ,定理 4 成立,即有 $n_2>n_1$.那么我们需要证明当 $d=n+1$ 时,如果 n_2 晚于 n_1 ,且 $n_2\leq n_1$,分枝 n_2 可以被剪切掉.同时,对结点 n_1,n_2 有相同父结点的这种情况已得证,因此我们只需证明第 2 种情况.

当 n_1,n_2 的父结点不同时,以图 4 为例,设 $n_1=r, n_2=t'$, r 的父结点是 o ,且 o 的左子结点是 r', t' 的父结点是 p ,且 p 的右子结点是 t. o,p 是有相同深度 n 的任意两个结点,且 $o<p$.并且与当前结点 r', r , t 和 t' 相关的任务周期为 T_j .此时我们有: $r'=\lfloor o/T_j\rfloor T_j$, $r=o$, $t'=\lfloor p/T_j\rfloor T_j$, $t=p$.因为 $t'\leq t, t>r$,所以只需证明对 t', 定理成立即可.此时有:

(1) 对 $r'=\lfloor o/T_j\rfloor T_j$,在这种情况下,由于 $o<p$,显然直接可得 $t'\geq r'$, 若 $t'=r'$, 分枝 t' 可以被剪切,定理成立.

(2) 对 $r=o$,采用反证法.假设 $t'<r$,分枝 t' 不能被剪切,设 $\lfloor p/T_j\rfloor=k$,那么有 $t'=kT_j$.根据假设, $t'<r=o$,可得 $r'=\lfloor o/T_j\rfloor T_j\geq\lfloor kT_j/T_j\rfloor T_j=kT_j$,即 $r'\geq kT_j$.又根据 $p_i(b)$ 定义, $p-t'<T_j\Rightarrow p<t'+T_j$ 又 $o<p, t'=kT_j$,可得

$o<(k+1)T_j$.因此 $r'\le o<(k+1)T_j$.这样便有 $kT_j\le r'<(k+1)T_j$.故有 $r'=t'$,可见分枝 t' 被覆盖,能被剪切,与假设矛盾.

综合上述情况,定理 4 得证. □

为了更好地理解定理 4,我们来看下面的例子.

例 2:用剪枝理论对 $M_5(9,15,16,30,100)$ 中 $p_4(T_5)$ 的二叉树表示进行剪枝,所有被剪去的分枝不再被显示.如图 5 所示,几乎一半的结点在最底层被剪掉了.

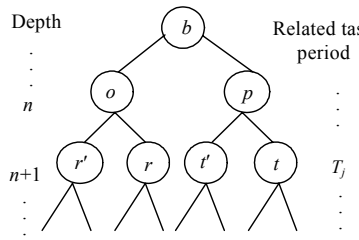


Fig.4 Nodes in bintree

图 4 结点示意图

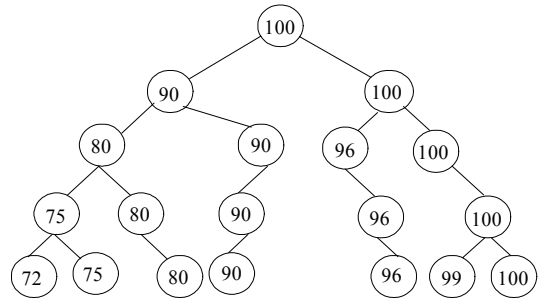


Fig.5 Bintree expression of $p_4(T_5)$ after pruning

图 5 剪枝后 $p_4(T_5)$ 的二叉树表示

定理 5. $\forall i=2,...,n, n>2$, 如果 $\lceil T_n/T_1\rceil\le 2$ 或者 $n=2, T_2<2T_1$,那么 $p_{i-1}(T_i)=p_{i-2}(T_{i-1})\cup\{T_i\}$.

引理 1. 如果 $T_{i+1}/T_i<2$,那么 $p_i(T_{i+1})\cap p_{i-1}(T_i)=p_{i-1}(T_i)$.

证明:如果 $T_{i+1}/T_i<2$,那么 $\lfloor T_{i+1}/T_i\rfloor=1$,这样有

$p_i(T_{i+1})=p_{i-1}(\lfloor T_{i+1}/T_i\rfloor T_i)\cup p_{i-1}(T_{i+1})=p_{i-1}(T_i)\cup p_{i-1}(T_{i+1})$,可见 $p_{i-1}(T_i)\subset p_i(T_{i+1})$,得证. □

定理 5 的证明:当 $n=2$, $T_2/T_1<2$ 时,有 $p_1(T_2)=\{T_1,T_2\}=p_0(T_1)\cup\{T_2\}$,定理 5 满足;

当 $n=3$ 时,因为 $\lceil T_3/T_1\rceil\le 2$,根据定理 4,当 $T_3<2T_1$ 时, $p_1(T_3)$ 的左子结点将被剪切,而当 $T_3=2T_1$ 时,其右子结点将被剪切.依次类推,对二叉树 $p_{n-1}(T_n)$ 结论依然成立,定理 5 得证. □

定理 4 与定理 5 都很重要,后面会被用到.

3 任务间可调度相关性及改进的 RM 可调度性判定算法

3.1 任务间可调度相关性分析

为便于问题的数学描述,首先给出如下定义:

定义 3. 设 $t\in p_{i-1}(T_i)$,定义函数 $f_i(t)$

$$f_i(t)=\begin{cases}T, \text{ if } \sum_{j=1}^{i}\lceil t/T_j\rceil C_j\le t\\ F, \text{ otherwise}\end{cases}.$$

这里, T 意味真(true), F 意味着假(false).该函数的意义为:如果任务调度空间中元素 t 使得函数值为真,即 $f_i(t)=T$,则根据定理 1 有任务 τ_i RM 可调度.注意,如果任务调度空间中元素 t 使得函数值为假,即 $f_i(t)=F$,任务 τ_i RM 不一定不可调度.因为在任务 τ_i 的 RM 调度空间中可能有另一元素 t',使得 $f_i(t')=T$,从而使得任务 τ_i RM 可调度.

定理 6. 如果 $i>j, T_i/T_j<2$, 那么 $p_{i-1}(T_i)\cap p_{j-1}(T_j)=p_{j-1}(T_j)$.

证明: $T_i/T_j<2\Rightarrow\lfloor T_i/T_{j+1...i-1}\rfloor=1$, 由引理 1 有, $p_{j-1}(T_j)\subset p_j(T_{j+1})\subset...\subset p_{i-1}(T_i)$, 因此 $p_{i-1}(T_i)\cap p_{j-1}(T_j)=p_{j-1}(T_j)$, 定理得证. □

定理 7. 如果 $i>j, t\in p_{i-1}(T_i)$, 同时有 $t\in p_{j-1}(T_j)$, 并且 $f_i(t)=T$,那么 $f_j(t)=T$.

证明: $i>j$, 设 $t\in p_{i-1}(T_i), f_i(t)=T$, 由定理 2,任务 τ_i 是可调度的,则有

$$\lceil t/T_1\rceil C_1+\lceil t/T_2\rceil C_2+...+\lceil t/T_j\rceil C_j+...+\lceil t/T_{i-1}\rceil C_{i-1}+C_i\le t.$$

由此可得

$$\lceil t/T_1\rceil C_1+\lceil t/T_2\rceil C_2+\ldots+\lceil t/T_j\rceil C_j\le t.$$

由已知条件 $t\in p_{j-1}(T_j)$, 可见任务 τ_j 可调度.定理得证. □

定义 4. $p_i^{\max}$ 表示 $p_{i-1}(T_i)$ 的最大元素, $p_i^{\min}$ 表示 $p_{i-1}(T_i)$ 的最小元素.

引理 2. $p_i^{\max}=T_i$.

由 $p_{i-1}(T_i)$ 的定义直接可得.

定理 8. 设 t' 整除 t ,记为 $t'|t$,并设 $t=kt'$, k 为正整数,如果有 $t\in p_{i-1}(T_i), t'\in p_{j-1}(T_j)$, $j=1,\ldots,i-1$,设 $t'/T_j=n+\varepsilon_j$, n 是一非负整数, $\varepsilon_j\in(0,1)$, 如果 $\forall j$, 有 $\varepsilon_j\in(1-1/k,1)$ 或者 $\varepsilon_j=0$,而且 $f_i(t)=T$,那么 $f_j(t')=T$.

证明:对 $\forall j$, ε_j 有两种情况:

(1) $\forall j$, $\varepsilon_j\in(1-1/k,1)$,可得

$$\left|t/T_j\right|=\left|kt'/T_j\right|=\left|k(n+\varepsilon_j)\right|=kn+\left|k\varepsilon_j\right|=(n+1)k=k\left|t'/T_j\right|.$$

(2) $\forall j$, $\varepsilon_j=0$,可得

$$\left|t/T_j\right|=\left|kt'/T_j\right|=\lceil kn\rceil=kn=k\left|t'/T_j\right|.$$

根据已知条件 $f_i(t)=T$,任务 τ_i 可调度,由定理 2,则有 $\sum_{k=1}^{i}\lceil t/T_k\rceil C_k\le t$.

由情况 1 和情况 2,有

$$\sum_{k=1}^{i}\lceil kt'/T_k\rceil C_k\le kt'\Rightarrow k\sum_{k=1}^{j}\lceil t'/T_k\rceil C_k+\sum_{k=j}^{i}\lceil t/T_k\rceil C_k\le kt'$$
$$\Rightarrow k\sum_{k=1}^{j}\lceil t'/T_k\rceil C_k\le kt'\Rightarrow\sum_{k=1}^{j}\lceil t'/T_k\rceil C_k\le t'.$$

任务 τ_j 可调度,定理得证. □

定理 9. 如果 $\lceil T_n/T_1\rceil\le 2$, $t\in p_{n-1}(T_n)$, 而且 $f_n(t)=T$, 则任务集 τ 可调度.

证明:根据定理 5,可分为两种情况:

(1) 当 $n>2$ 或者 $T_n<2T_1$ 时,有 $p_{n-1}(T_n)=p_{n-2}(T_{n-1})\cup\{T_n\}$. 由定理 5,如果 $t\in p_{n-1}(T_n)$, 并且 $t\in p_{n-2}(T_{n-1})$, 由定理 7, $f_n(t)=T\Rightarrow f_{n-1}(t)=T$ 成立.另一方面,如果 $t=T_n$,则有

$$\sum_{i=1}^{n-1}2C_i+C_n\le T_n\Rightarrow\sum_{i=1}^{n-1}C_i+C_n/2\le T_n/2\Rightarrow\sum_{i=1}^{n-1}C_i\le T_{n-1}.$$

由定理 1 可知, τ_{n-1} 可调度.以此类推,可知任务集 τ 可调度.

(2) $n=2$ 且 $T_n=2T_1$,此时 $p_1(T_2)=\{T_2\}$, 这样,如果 $f_2(T_2)=T$, 可得 $2C_1+C_2\le T_2$, 不等式 $C_1\le T_2/2=T_1$ 成立.很显然,任务集 τ 可调度.

综上所述,定理得证. □

定理 10. 假设 $t\in p_{i-1}(T_i)$, 并且 $f_i(t)=T$, 同时 $T_i/T_{i-1}<2$, 如果 $t\le T_{i-1}$, 那么,

$$t\in p_{i-2}(T_{i-1}), f_{i-1}(t)=T.$$

引理 3. $b\in p_i(b),\forall t\in p_i(b), t\le b.$

由定义,有 $p_0(b)=\{b\}\in p_i(b)$, 根据引理 2,得证. □

定理 10 的证明:首先我们证明当 $t\le T_{i-1}$ 成立时, $t\in p_{i-2}(T_{i-1})$.由定理 6,有

$$p_{i-1}(T_i)\cap p_{i-2}(T_{i-1})=p_{i-2}(T_{i-1}).$$

由定理 5,当 $T_i/T_{i-1}<2$ 时,

$$p_{i-1}(T_i)=p_{i-2}(\lceil T_i/T_{i-1}\rceil T_{i-1})\cup p_{i-2}(T_i)=p_{i-2}(T_{i-1})\cup p_{i-2}(T_i).$$

已知 $t\in p_{i-1}(T_i)$,如果 $t\notin p_{i-2}(T_{i-1})$,那么一定有 $t\in p_{i-2}(T_i)$.根据引理 3 和定理 4,有 $t>T_{i-1}$, 否则,分枝 t 将会被剪掉.这与已知条件矛盾.因此, $t\in p_{i-2}(T_{i-1})$ 得证.

另一方面,根据定理 7,如果 $f_i(t)=T$, 则有 $f_{i-1}(t)=T$. 综上,定理 10 得证. □

3.2 示 例

为了便于理解上述定理,我们给出如下例子:

例 3:可调度区域 $M_4(3,8,20,30)$,由定理 3,各个任务调度空间如下:

$$p_3(30)=\{15,16,18,20,24,30\}, p_2(20)=\{15,16,18,20\}, p_1(8)=\{6,8\}, p_0(3)=\{3\}.$$

运用定理 7 和定理 8,可以得到如下任务间的可调度相关性:

$$f_3(15)=T\Rightarrow f_2(15)=T; \quad f_3(16)=T\Rightarrow f_2(16)=T\Rightarrow f_1(8)=T;$$
$$f_3(18)=T\Rightarrow f_2(18)=T\Rightarrow f_1(6)=T; \quad f_3(20)=T\Rightarrow f_2(20)=T;$$
$$f_3(30)=T\Rightarrow f_2(15)=T.$$

举例说来,当任务 τ_3 的调度空间有一元素 16 使得任务 τ_3 RM 可调度时,我们就可知 τ_2 和 τ_1 亦 RM 可调度.

例 4:假设有一和谐任务集 $\tau=\{\tau_1,...,\tau_n\}$,根据和谐任务集的定义[5,13],此时有 $\forall i, i\in[1,n-1]$, $T_i \mid T_{i+1}$, 设 $T_n=k_iT_i+\varepsilon_i$, 很显然 $\forall i, \varepsilon_i=0$. 根据引理 3, $T_i\in p_{i-1}(T_i)$, $T_n\in p_{n-1}(T_n)$, 我们有

$$\frac{\sum_{i=1}^{n}\lceil T_n/T_i\rceil C_i}{T_n}=\frac{\sum_{i=1}^{n}k_iC_i}{T_n}=\sum_{i=1}^{n}C_i/(T_n/k_i)=\sum_{i=1}^{n}C_i/T_i\ .$$

可见,根据定理 1,当 CPU 利用率 $U=\sum_{i=1}^{n}C_i/T_i\le 1$ 时, $f_n(T_n)=T$, 任务 τ_i RM 可调度.由定理 8,因为 $T_{n-1}\mid T_n$, 而且 $\forall i=1,...,n-1, T_i\mid T_{n-1}$, 可得到 $f_{n-1}(T_{n-1})=T$, 任务 τ_{n-1} RM 可调度.以此类推,整个任务集是 RM 可调度的.从而证明了文献[5,13]的理论:对任意的和谐任务集,如果 $U=\sum_{i=1}^{n}C_i/T_i\le 1$,那么和谐任务集是 RM 可调度的.

3.3 改进的RM可调度性判定算法,简称ISTA(improved schedulability test algorithm)

与文献[9]中的 HET 算法不同,ISTA 采用后向推理.伪代码如下:

```
boolean RMTest( τ ) {
    for(int i=n;i>0;i--) {
        if(!isSched(i))          //Test each task's schedulability backward
            return false;        //Task τi is not schedulable
        else if ( Ti / T1 ≤ 2 )
            return true;         //Theorem 9 is applied here
    }
    return true;                 //Task set is schedulable
}
```

函数 isSched(int i)用于确定任务 τ_i 是否错过最后截止期.在这个函数中,为了剪去一切不必要的计算,一个跟踪执行流的全局变量被保存,函数通用的过程如下:

Step 1. $div=T_{i+1}/T_i$;

Step 2. If $div<2$ && $latest\_t<T_i$, **return** true; //Theorem 10 is applied here

Step 3. read one element t from $p_{i-1}(T_i)$; //Theorem 4 is applied here

Step 4. if $latest\_t=kt$, k is an integer and $\forall j=1,...,i, t/T_j=n+\varepsilon_j, \varepsilon_j>1-1/k$ or $\varepsilon_j=0$

return true; //Theorem 8 is applied here

Step 5. if ($\sum_{j=1}^{i}\lceil t/T_j\rceil C_j\le t$){ //Theorem 1 is applied here

$latest\_t=t$; //Global variable to remember the executing flow

Step 6. if all elements of $p_{i-1}(T_i)$ have been tested, **return** false, otherwise, go to Step 3.

3.4 算法性能分析

普通实时系统的调度判定通常是一个强 co-NP-hard 问题[14],也就是说,如果 $P\ne NP$,则不可能存在多项式时间算法和伪多项式时间算法.在第 1.1 节理想的 RM 调度模型中,假设所有任务在 0 时刻发出请求,并且任务截止期与请求周期相等,这是一种隐含截止期的同步实时系统.BARUAH 等人[14]指出了这种特殊实时系统的调度判定问题的复杂性目前仍然未知,现阶段人们所发现的最好的算法是伪多项式时间算法.这里给出 ISTA 的性能分析及实现开销.

设有周期任务集 $\tau=\{\tau_1,...,\tau_n\}$,对任务 $\tau_n=(C_n,T_n)$, 根据二叉树理论,其调度空间最多有 2^{n-1} 个元素即叶结点数量,对每一个结点元素进行调度判定需执行 n 次操作,任务 τ_n 调度判定的算法复杂度为 $o(2^{n-1}n)$,为指数时间算法.实际上,其调度空间元素个数最多为 T_n,而且 $T_n<<2^{n-1}$,所以真实复杂度应为 $o(nT_n)$,即为伪多项式时间算法.考虑一共有 n 个任务,ISTA 时间复杂度为 $o(n^2T_n)$,该算法执行时间不仅与问题规模相关,而且与任务集 τ 中最大的任务周期 T_n 相关,所以是一种伪多项式时间算法.同时,在实现 ISTA 时,还存在两方面的实现开销:一是需要 $o(n\log n)$ 的时间开销对任务集 τ 进行排序,使得 $T_1<T_2<...<T_n$;二是需要 $o(T_i)$ 的时间对任务 τ_i 的调度空间进行二叉树遍历.

4 对比仿真性能比较

针对文献[7]给出的未简化的 RM 可调度性判定充要条件,文献[9]提出的 HET 算法以及本文提出的 ISTA,这里给出 3 种算法的平均性能比较.在文献[9]中,衡量算法复杂度的性能指标是算法内部的循环次数.本文结合 ISTA 的实际情况,算法性能指标定义为各算法占用的 CPU 处理时间.通常可以认为算法占用 CPU 的时间是算法内总循环次数的递增函数,即有

$$CPUElapsedTime=f(nTimes).$$

这里,*nTimes* 就是每种算法内循环的总次数.同时,当 $n_1>n_2$ 时,有 $f(n_1)>f(n_2)$.这样,本文定义的性能指标就与文献[9]中的性能指标相一致.

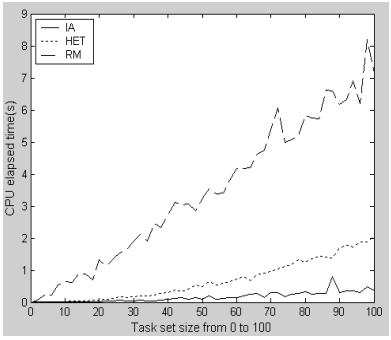
实验随机产生各个任务,各任务的周期 T_i 随机地从[1,10000]中产生,各任务的执行时间 C_i 随机地从 $[0,\varsigma T_i](\varsigma=1/(\psi m),\psi\in(0,1))$ 产生得到,这里 ψ 是一个调整系数,n 是任务集内任务的数量.实验发现,如果 C_i 是一个在 $[0,T_i]$ 内的随机数,那么整个任务集绝大部分是不可调度的,所以我们定义了一个与任务集所含任务个数成反比例的影响因子 ς,使得可调度任务集的数量可以控制.调整系数 ψ 越小,影响因子 ς 就越大,可调度的任务集越少.反之,调整系数 ψ 越大,影响因子 ς 就越小,可调度的任务集越多.为了得到一个平均性能,避免由于不同大小的相邻任务集因特性不一样带来平均性能的剧烈波动,对同样大小的任务集选取 5 个样本进行测试,不同大小的相邻任务集所含任务个数相差为 2,每次实验任务集大小从 0~100,这样每次实验一共随机产生 250 个任务集.仿真环境是在 Window 2000,Pentium III 600,256M RAM 环境下进行的,同时对每一个实验,各算法运行环境相同,以保证算法性能的可比性.

所有实验均假设,算法所占用 CPU 处理时间同时也是任务集所含任务个数的函数.实验给出不同影响因子情况下的平均性能比较结果.在下面的所有比较图中,横坐标是采样任务集内所含任务个数,在实验中,任务集所包含的最大任务个数是 100.纵坐标是各算法在判定相同的任务集时所占用的 CPU 时间,*num* 表示实验中不可调度的总的任务集的个数.RM 代表利用未简化的 RM 充要条件进行可调度性判定分析的算法,我们用实线表示 ISTA 算法的性能,点虚线表示 HET 算法的性能,破折线表示利用未简化的 RM 充要条件算法的性能.

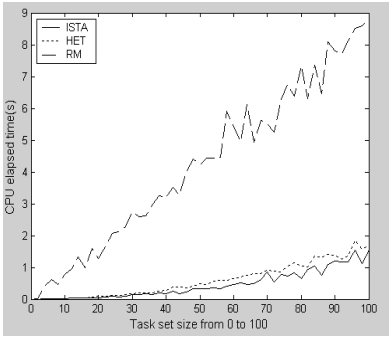
对具有属性 $T_n/T_1\le 2$ 的任务集,过去已有许多学者做了很多研究工作.可以证明除了 $n=2,T_2=2T_1$ 这种特殊情况外,对所有的任务集,都有 $S_i=p_{i-1}(T_i)=\{T_1,T_2,...,T_i\}$.也就是说,各任务调度空间无法被减少,任务集可调度区域 M_n 无法被简化.此时 3 种算法性能一致,这里不再进行比较.

在下面的实验中,我们不要求 $T_n/T_1\le 2$,每个任务的请求周期 T_i 在[1,10000]范围内,这种情况更符合大多数的实际需求,具有更好的现实意义.此时,$p_{i-1}(T_i)$ 是 S_i 的子集,各个任务调度空间被简化,各个任务集可调度区域 M_n 减少.实验结果如图 6 所示.

从图 6(a)、图 6(b)可以看出,对于含有任意周期任务的任务集,ISTA 要好于未简化的 RM 可调度性判定,同时优于 HET 算法.其原因是 ISTA 与 HET 算法的各个任务调度空间均少于未简化的 RM 可调度性判定的调度空间,而 ISTA 一方面利用剪枝理论,优化了任务调度空间的搜索过程,另一方面,分析了具有逻辑“与”关系的任务间的可调度性相关性,简化了部分判定分析,克服了 HET 算法的部分不足,相对于同是伪多项式时间的 HET 算法而言,具有更小的时间开销,也就表现出更好的优越性.表 2 给出了当 $num=99$ 时部分任务集所占用 CPU 处理时间的比较结果.



(a) $\psi = 0.65, \varsigma = 1/(0.65n), num = 173$



(b) $\psi = 0.75, \varsigma = 1/(0.75n), num = 12$

Fig.6 Average performance comparison among three algorithms with any periodic task sets

图 6 任意周期任务时 3 种算法平均性能比较

Table 2 Comparison on CPU elapsed time among different task sets (s)

表 2 不同任务集所占 CPU 时间比 (秒)

Task set size	20	30	40	50	60	70	80	90	100
RM	1.842 3	2.333 3	3.151 0	4.726 7	4.773 7	5.892 0	7.050 0	8.001 3	8.696 0
HET	0.067 0	0.136 7	0.247 0	0.507 3	0.637 7	0.841 0	1.135 3	1.582 3	1.752 3
ISTA	0.060 3	0.113 7	0.227 0	0.424 0	0.350 0	0.767 7	0.927 7	0.971 7	1.442 0
Ratio impr. by ISTA vs. HET (%)	10	16.83	8.10	16.42	45.12	8.72	18.29	38.59	17.71

从表 2 可以看出,ISTA 平均性能比 HET 算法的平均性能要好,最大平均性能提高接近 45.12%,性能提高率平均可达 19.98%.而且任务集越大,平均性能提高越多.同时从图 6(b)还可以看出,不可调度的任务集数量越少,平均性能提高也越大.

此外,从图 6 可以看出,不同大小的相邻任务集之间平均性能的变化对 3 种算法都会出现波动现象,这是因为算法对不同属性的任务集特性不一.比如,有两个相邻任务集,其中一个任务集包含 80 个任务,另一个任务集包含 82 个任务.如果第 1 个任务集可调度,第 2 个任务集不可调度,并且第 2 个任务集的第 10 个任务不可调度,很显然,判定的顺序以及不可调度的任务出现的时间对可调度性判定算法占用的处理机时间有很大程度的影响.同时,任务集越大,差别也将越大.实验还发现,HET 算法不能很好地处理任务集中包含具有相同周期任务的特殊情况,而 ISTA 和未简化的 RM 可调度性判定却能得到正确的结果.

5 结 论

本文在传统的 RM 调度充要条件的基础上,结合二叉树理论,给出了任务可调度空间的概念,用二叉树形式表示了任务调度空间元素之间逻辑“或”的关系,给出了相应的剪枝理论.在此基础上,进一步研究了任务集内具有逻辑“与”关系的单个任务之间的可调度相关性,提出了基于未简化的 RM 调度算法的一种新的改进的周期性任务集的可调度性判定算法.这一算法在减少调度空间的同时,兼顾了各个任务之间的可调度相关性,有效地克服了传统的 RM 算法和 HET 算法的不足.在本文中进行的仿真对比实验结果显示了 ISTA 的优越性.我们相信,这一结果将会吸引更多的学者对 RM 可调度性判定多项式算法或伪多项式算法进行更深入的研究.

作为更进一步的研究工作方向,实时系统在不完全信息情况下的优化设计是一项有意义的深入点.以前的工作大多基于每个任务的属性是已知的这一理想情况,然而这并不能完全代表所有的实际应用情况.比如,每个任务的执行时间与多个因素相关,如硬件、软件的实现等,它不是固定不变的.如何利用现有的理论指导实时系统的优化设计将是一个有意义的工作.

致谢 作者十分感谢中国科学院软件研究所互联网软件技术实验室实时系统组的淮晓永博士、邢建生博士生以及实验室其他研究生.感谢他们与作者的有益讨论.

References:

[1] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard real time environment. Journal of ACM, 1973,20(1): 46−61.

[2] Burchard A. New strategies for assigning real-time tasks to multiprocessor systems. IEEE Trans. on Computer, 1995,44(12): 1429−1442.

[3] Bini E, Buttazzo GC, Buttazzo G. A hyperbolic bound for the rate monotonic algorithm. In: Proc. of the 13th Euromicro Conf. on Real-Time Systems. Delft: IEEE Computer Society Press, 2001. 59−68.

[4] Bini E, Buttazzo GC, Buttazzo G. Rate monotonic analysis: The hyperbolic bound. IEEE Trans. on Computers, 2003,52(7): 933−942.

[5] Han C-C, Tyan HY. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithm. In: Proc. of the 18th IEEE Real-Time Systems Sym. Madrid: IEEE Computer Society Press, 1997. 36−45.

[6] Park DW, Natarajan S. A generalized utilization bound test for fixed-priority real-time scheduling. In: Proc. of the 2nd Int'l Workshop on Real-Time Computing Systems and Applications. Tokyo, 1995. 73−77.

[7] Lehoczky JP, Sha L, Ding Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Proc. of the Real-Time Systems Symp. Santa Monica: IEEE Computer Society Press, 1989. 166−171.

[8] Katcher DI, Arakawa H, Strosnider JK. Engineering and analysis of fixed priority schedulers. IEEE Trans. on Software Engineering, 1993,19(9):920−934.

[9] Bini E, Buttazzo GC. The space of rate monotonic schedulability. In: Proc. of the 23rd IEEE Real-Time Systems Symp. Austin Texas: IEEE Computer Society Press, 2002. 169−178.

[10] Wang YJ, Chen QP. On schedulability test of rate monotonic and its extendible algorithms. Journal of Software, 2004,15(6): 799−814 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/799.htm

[11] Wang YJ, Lane DM. Solving a generalized constrained optimization problem with both logic AND and OR relationships by a mathematical transformation and its application to robot path planning. IEEE Trans. on Systems, Man and Cybernetics, Part C: Application and Reviews, 2000,30(4):525−536.

[12] Yan WM, Wu WM. Data Structure. 2nd ed., Beijing: Tsinghua University Press, 1992. 120−128 (in Chinese).

[13] Naghibzadeh M, Kim KH. A modified version of rate-monotonic scheduling algorithm and its efficiency assessment. In: Proc. of the 7th IEEE Int'l Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2002). San Diego: IEEE Computer Society Press, 2002. 289−294.

[14] Baruah S, Howell R, Rosier L. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. Real-Time Systems: The Int'l Journal of Time-Critical Computing, 1990,2(4):301−324.

附中文参考文献:

[10] 王永吉,陈秋萍.单调速率及其扩展算法的可调度性判定.软件学报,2004,15(6):799−814 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/799.htm

[12] 严蔚敏,吴伟民.数据结构.第 2 版,北京:清华大学出版社,1992.120−128.