软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(2):411−428 [doi: 10.13328/j.cnki.jos.005025]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

偶发实时系统可调度性分析问题的整数规划方法*

孙景昊[1], 孙景昶[2], 关 楠[1], 邓庆绪[1]

[1](东北大学 信息科学与工程学院,辽宁 沈阳 110004)

[2](四川大学 计算机学院,四川 成都 610207)

通信作者: 邓庆绪, E-mail: dengqx@mail.neu.edu.cn

摘 要: 偶发实时任务最早截止期优先(earliest deadline first,简称 EDF)可调度分析是实时系统领域经典的 NP 困难问题.现有的伪多项式时间判定算法(pseudo-polynomail time decision algorithm,简称 PTDA)均局限于利用率 U 严格小于 1 的同步任务系统.对于 $U\leqslant 1$ 的同步系统或更加困难的异步系统,现有 PTDA 则不再适用.针对以上问题,为同步和异步两类实时系统建立了统一的整数规划模型,其规模并不依赖于利用率 U 的取值.基于多面体理论证明了模型维数和极大诱导不等式,进而提出了同/异步系统上 EDF 可调度性分析问题统一的多项式时间线性松弛求解方法.实验结果表明,该方法能够获得较紧的问题解下界,在异步和同步系统中,线性松弛解与最优解之间的平均百分界差 *gap* 分别为 0.78%和 1.27%.另外,随机生成了大量同步和异步系统的算例,用于该算法和传统算法进行性能比较.对于同步算例,实验结果表明,在 $U>0.99$ 时,该算法能够对 70%的算例给出判定结果,算法性能与 QPA 算法相比有指数级提升.对于异步算例,实验结果表明,该算法能够对近 96%的算例给出可调度性判定.与传统算法相比,该方法将不能判定可调度性的算例比例平均降低了 29.27%.对于剩余的 4%的算例,该算法将可调度上界的值平均降低了近 10^4 倍.

关键词: 截止期优先;可调度性分析;整数规划;多面体分析;线性松弛

中图法分类号: TP316

中文引用格式: 孙景昊,孙景昶,关楠,邓庆绪.偶发实时系统可调度性分析问题的整数规划方法.软件学报,2017,28(2):411−428. http://www.jos.org.cn/1000-9825/5025.htm

英文引用格式: Sun JH, Sun JC, Guan N, Deng QX. Integer programming approach for schedulability of sporadic real-time systems. Ruan Jian Xue Bao/Journal of Software, 2017,28(2):411−428 (in Chinese). http://www.jos.org.cn/1000-9825/5025.htm

Integer Programming Approach for Schedulability of Sporadic Real-Time Systems

SUN Jing-Hao[1], SUN Jing-Chang[2], GUAN Nan[1], DENG Qing-Xu[1]

[1](School of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

[2](College of Computer Science, Sichuan University, Chengdu 610207, China)

Abstract: Schedulability test for EDF (earliest deadline first) systems is one of the classical NP-hard problems in the study of real-time system. Current researches mainly focus on the synchronous systems with the utilization U strictly less than 1, which can be decided exactly in pseudo-polynomial time. However, these results cannot be easily extended to the synchronous systems with $U\leqslant 1$ or to the asynchronous systems even with $U<1$. In this paper, a unified integer programming formulation, where the associated scale is independent of utilization U, is proposed for the EDF schedulability problems in both of the synchronous and asynchronous systems. The polyhedral structure of the formulation is investigated and a kind of facet inequalities is derived, resulting in a linear relaxation approach with polynomial-time complexity. Numerical results on a large scale randomly generated asynchronous and synchronous instances show that

* 基金项目: 国家重点基础研究发展计划(973) (2014CB360509); 国家自然科学基金(61672147, 61300194, 61300022, 61472072)

Foundation item: National Program on Key Basic Research Project of China (973) (2014CB360509); National Natural Science Foundation of China under Grant (61672147, 61300194, 61300022, 61472072)

收稿时间: 2014-06-17; 修改时间: 2014-12-22; 采用时间: 2015-12-28

the proposed method can obtain a tight gap (0.78% and 1.27% respectively on average) between the relaxation and the optimal integer solutions. Furthermore, the comparison with the QPA exhibits that the new method is available for 70% synchronous instances and exponentially reduces the calculation time especially in situations when U>0.99. Finally, experiments on asynchronous systems find that nearly 96% instances can be exactly solved by the method, which is 29.27% lesser than the traditional method. For the rest of the instances, the upper bound of the schedulability test can be sharply reduced. For most instances, the new bound is 10^4 smaller than the traditional ones.

Key words: earliest deadline first; schedulability; integer programming; polyhedral analysis; linear relaxation

在实时系统领域,周期性实时任务调度是最核心的理论问题之一[1].最早截止期优先(earliest deadline first,简称 EDF)作为单处理器系统中最优的调度算法[2],得到众多学者的广泛关注和研究.当前研究主要关注同步和异步两大类实时系统.所谓同步系统,是指系统中所有任务释放首个作业的时刻均相同.与之相反,异步系统则不要求任务同步地释放首个作业,而是允许为每个任务释放首个作业定义一个特定的时刻,称为相位.特殊地,当异步系统中所有任务的初始相位均相同时,该异步系统即退化为同步系统.因此,同步系统可看作是异步系统的一个特例.同步和异步系统上的 EDF 可调度性分析问题十分经典又极具挑战性,有些早已得到解决,成为实时系统理论中里程碑式的成果;更多问题则在提出后数十年间都未能突破,直到近期才得以解决,同时又衍生出新的开放性问题.

同步系统的 EDF 可调度性分析问题在 20 世纪 70 年代得到研究.Liu 等人率先研究了任务截止期与周期相等(D_i=P_i)的隐含截止期任务模型,并给出了多项式时间的判定算法[3].注意到,隐含截止期任务模型是实时系统的理想模型.在实际应用中,实时任务的截止期和周期之间并没有必然联系.很多偶发性任务,例如中断响应任务,其特点是触发频率较低,并且任务一旦触发就需要在很短时间内完成.这类任务可抽象为截止期小于周期的任务模型.对于截止期大于周期的任务模型,常见于多媒体应用等事件流的处理中.这类系统通常设有缓存来暂存中间结果,任务的截止期取决于缓存区的大小.例如,缓存如果能够暂存 3 个中间结果,则任务的截止期可以是周期的 3 倍.显然,这类系统允许任务具有大于周期的截止期.另外,在类似于基于 Stateflow[4]等同步语言的代码生成中,也存在类似的任意截止期模型的应用.

对于任意截止期的同步任务系统(D_i≠P_i),自 20 世纪 80 年代起就有学者着手研究[5,6],并发现这类一般化模型上的 EDF 可调度性分析异常困难,问题的计算复杂性长期以来难以确定.直到 2010 年,Eisenbrand 等人才最终从理论上证明同步系统上的 EDF 可调度分析问题是弱 coNP 困难的[7],这意味着同步系统存在伪多项式时间的 EDF 可调度判定算法(pseudo-polynomail time decision algorithm,简称 PTDA).目前,PTDA 的研究主要集中在利用率严格小于 1 的同步系统.Zhang 等人[8]提出的 QPA 算法是当前最优秀的精确判定算法之一.尽管 QPA 在理论上仍属于伪多项式时间算法,但大量实验结果表明,与传统算法相比,QPA 能够成指数级地降低计算时间,具有接近常数的计算复杂度[8].本文应用文献[8]的方法生成了更大范围的随机算例样本,并对 QPA 算法做进一步的实验(详见第 5.3.1 节).研究发现,QPA 算法对于利用率 U≤0.99 的算例可保持常数的时间复杂度,计算次数均不超过 100,但当 U>0.99 时,却迅速退化为指数计算时间.Baruah 等人指出,利用率 U 趋于 1 的同步系统可调度性判定仍然属于很困难的问题.该问题是否存在 PTDA 至今仍被列为实时系统调度领域的 5 个开放性课题之一[1].

更困难的是,对于异步系统而言,可调度性分析属于强 coNP 困难问题.该结论甚至对利用率 U 严格小于 1 的异步系统同样成立[5].这从理论上否定了异步系统存在 PTDA 算法的可能.同步系统中所有 PTDA 算法(包括 QPA)均不再适用.目前异步系统的可调度性分析问题仅有充分判定条件的研究[5],即当算法给出“可调度”的结论时,异步系统一定可调度.但是,当算法返回“不可调度”时,系统的可调度性则不确定.

综上,U 趋于 1 的同步系统以及异步系统中的可调度性分析问题均属于很困难的问题.为了更高效地求解这两类困难问题,本文同时开展了同步和异步两类实时系统的整数规划模型和线性松弛方法的研究.首先,本文为异步系统上的可调度性分析问题建立了一般化的整数规划模型.该模型中的变量和约束不等式具有多项式规模,不依赖于利用率 U 的取值,从根本上消除了 U 对算法复杂性的影响.在理论上,本文对模型约束集合展开多面体分析,证明了模型维数和极大诱导不等式,并基于这些结果为可调度性分析问题提出了多项式时间的线性

松弛求解方法.其次,由于同步系统是异步系统的简单特例.因此,异步系统的模型和方法很容易扩展到同步系统中,并且异步系统中问题解空间的多面体结论在同步系统中同样成立.另外,在同步系统中,本文的线性松弛算法可以嵌入 QPA 算法中缩减解空间的策略,这使得算法的计算性能又进一步得到提升.

本文随机生成了大量同步和异步系统的算例,通过实验验证了本文方法的有效性.首先验证了极大诱导不等式的有效性.实验结果表明,本文模型能够获得较紧的问题解下界.在异步和同步系统中,线性松弛解与最优整数解之间的平均百分界差分别为 0.78%和 1.27%.另外,分别从同步和异步两方面开展实验,将本文算法与传统算法进行性能比较.一方面,对于同步系统算例,将本文算法与 QPA 算法进行比较.实验结果表明,对于 U>0.99 的算例,本文算法能够对 70%的算例给出可调度性的判定结果,算法的迭代次数较 QPA 算法有指数级的降低.例如,一个具有 30 个任务的算例,应用 QPA 算法要迭代 33 653 次,而本文算法只需迭代 3 次.另一方面,对于异步系统,通过实验比较了本文算法与文献[5]中传统算法的计算效率.实验结果表明,对于整个随机异步算例集合,本文算法能够对近 96%的算例给出可调度性判定,将传统算法[5]不能判定可调度性的算例比例平均降低了 29.27%.对于剩余的 4%的算例,本文算法能够降低可调度性算法的时间上界.与传统上界结果相比,新上界的值平均降低了近 10^4 倍.

本文第 1 节介绍可调度性分析问题的形式化定义.第 2 节介绍同步/异步系统 EDF 可调度分析的相关研究.第 3 节给出问题的整数规划模型以及相关的多面体理论结果.第 4 节提出用于可调度性判定的多项式时间线性松弛方法.第 5 节给出实验结果.第 6 节对全文进行总结,并对未来工作进行展望.

1 问题的数学定义

本节给出实时任务系统的形式化定义,并阐述一些相关的基本术语和概念.实时调度理论与“传统”调度理论[9]研究的一个主要差异在于实时任务具有周期性复发的特征.一般地,实时任务系统通常定义为由一组有限数量且相互独立的实时任务构成的集合 $T=\{\tau_1,\ldots,\tau_n\}$.其中,每个实时任务 τ_i 都将产生无限数量的作业序列.本文考虑偶发实时任务系统,即每个任务 τ_i 中任意两相邻作业的释放时间间隔不确定,但两个释放动作间存在一个最小的时间间隔,称为周期.

1.1 偶发性实时任务系统

系统 T 中每个偶发实时任务通常定义为一个四元组 $\tau_i=(\varphi_i,C_i,D_i,P_i)$,其中元素的具体含义描述如下.

- $\varphi_i\in\mathbf{R}^+$:任务 τ_i 中第 1 个作业的释放时刻(又称相位),令 $\Phi=\max\{\varphi_1,\ldots,\varphi_n\}$.
- $C_i\in\mathbf{R}^+$:任务 τ_i 中作业的最坏执行时间(WCET),规定任务 τ_i 释放的所有作业具有相同的 C_i.
- $D_i\in\mathbf{R}^+$:任务 τ_i 的相对截止期,规定任务 τ_i 释放的所有作业具有相同的 D_i;若 τ_i 的某作业 J 在 t 时刻释放,则该作业必须在 $t+D_i$ 时刻之前完成.换言之,该作业在$[t,t+D_i]$时间段内需占用 C_i 个时间单位的处理器资源.$t+D_i$ 称为作业 J 的绝对截止期.
- $P_i\in\mathbf{N}^+$:任务 τ_i 的周期,表示 τ_i 中任意两个连续释放作业间的最小时间间隔;若 τ_i 在 t 时刻释放了一个作业,则 τ_i 的下一个作业需在 $t+P_i$ 时刻之后释放(含 $t+P_i$ 时刻).

若任务系统 T 关联的最大相位 $\Phi=0$,则 T 称为同步系统,否则,T 称为异步系统.显然,同步系统是异步系统的特例.另外,若 T 中每个任务 τ_i 的相对截止期 D_i 和周期 P_i 相等($D_i=P_i$),则称 T 为隐含截止期任务系统;若 T 中每个任务 τ_i 关联的 $D_i\leqslant P_i$,则称 T 为约束截止期任务系统;本文研究任意截止期的任务系统,即 T 中每个任务 τ_i 关联的 D_i 与 P_i 之间没有任何约束关系,D_i 可小于、等于甚至大于 P_i.

接下来,进一步给出文中将会用到的其他术语和相关定义.

- $U_i=C_i/P_i$:任务 τ_i 的利用率.
- $U=\sum_{k=1}^{n}U_i$:任务系统 T 关联的总利用率.
- $lcm(P_1,\ldots,P_j)$:周期 $P_1,\ldots,P_j$ 的最小公倍数.
- $H=lcm(P_1,\ldots,P_n)$为系统 T 的超周期.

• $\eta_i(t_1,t_2)=\max\{\lfloor(t_2-\varphi_i-D_i)/P_i\rfloor-\lceil(t_1-\varphi_i)/P_i\rceil+1,0\}$:任务 τ_i 在时间段$[t_1,t_2)$内释放的作业数,其中每个作业的释放时刻均大于等于 t_1,其绝对截止期均小于等于 t_2[5].

1.2 基于EDF调度的实时任务系统运行行为

对实时任务系统 T 的调度行为可抽象为一个指派函数$\sigma:R\to T\cup\{\varnothing\}$.对于任意时间点 t:(1) $\sigma(t)=\tau_i$ 表示调度算法指派处理器在 t 时刻执行任务 τ_i 释放的作业;(2) $\sigma(t)=\varnothing$说明在 t 时刻处理器不执行任何作业,处于空闲状态.本文考虑单处理器上可抢占式调度方法,即处理器可以随时中断当前正在执行的任务作业,转而处理其他任务释放的作业,被中断的作业会在未来的某个时间点恢复执行.在当前的研究中,通常假设处理器上这种任务间的切换不产生额外的代价和开销.

目前,实时系统中的调度算法主要分为静态优先级调度和动态优先级调度.本文关注一类应用最为广泛的动态优先级调度算法——最早截止期优先(EDF)算法[3,10,11].根据 EDF 调度策略,在每个时间点,处理器都会从那些已经释放但未完成的若干作业中挑选出绝对截止期最小的那个优先执行.直观上,EDF 调度策略倾向于让那些最先可能错失截止期的作业抢占处理器资源,从而使尽可能多的作业能够在其相应的绝对截止期之前完成.在 EDF 调度策略控制下,如果系统中的每个任务释放的所有作业都能在其相应的绝对截止期之前完成,则称系统可被 EDF 成功调度.1974 年,Dertouzos[2]从理论上证明了 EDF 是最优的单处理器可抢占式调度算法.这意味着,任意一个实时任务系统如果不能被 EDF 策略成功调度,则该任务系统也不能被其他算法成功调度.因此,将能够被 EDF 策略成功调度的任务系统简称为可调度的系统;反之,则称为不可调度的系统.

1.3 可调度性分析问题

可调度性分析问题是指,对于给定的任务系统 T,是否存在一个判定算法 A:若 T 能够被某种策略成功调度,则算法 A 回答“是”;否则,若 T 不能被任何调度策略成功调度,则算法 A 回答“否”.根据第 1.2 节所述,在单处理器系统中,EDF 是最优的调度策略.因此,单处理器上任务系统 T 的可调度性分析问题又等价为系统 T 是否可被 EDF 策略成功调度的判定问题.

可调度性判定问题的传统分析方法框架均建立在一种被称为处理器需求范式(processor demand criterion)的基础之上.下面首先给出处理器需求函数的定义.

定义 1. 处理器需求函数(processor demand function,简称 PDF)[5]$df(t_1,t_2)$表示任务系统 $T=\{\tau_1,\ldots,\tau_n\}$在给定的时间段$[t_1,t_2)$内可能产生的最大执行时间需求,具体计算公式如下:

$$df(t_1,t_2)=\sum_{i=1}^{n}\eta_i(t_1,t_2)C_i.$$

易知,在时间段$[t_1,t_2)$内,处理器至少要提供 $df(t_1,t_2)$个时间单位的计算资源用来执行任务系统 T 中已释放的作业.否则,T 中的任务就有错失截止期的危险.根据 PDF 的定义,能够很自然地推导出系统可调度的一个必要条件,即任务系统在任意时间段产生的时间需求量不能超过该时间段的长度,其形式化描述如下:

$$\forall 0\leqslant t_1<t_2: df(t_1,t_2)\leqslant t_2-t_1.$$

接下来分别从异步和同步系统两个方面介绍可调度性分析理论的基本原理.

定理 1[5]. 异步任务系统 T 在单处理器上是可调度的,当且仅当 $U\leqslant 1$,且$\forall 0\leqslant t_1<t_2\leqslant \Phi+2H: df(t_1,t_2)\leqslant t_2-t_1$.

以上定理实际上蕴含着一个异步系统可调度性分析算法[12]:在时间段$[0,\Phi+2H]$中,检查每个可能有作业释放的时间点 t_1,计算 t_1 对应的所有需求函数 $df(t_1,t_2)$,其中时间点 t_2 满足 $t_2>t_1$,代表系统可能释放的所有作业关联的绝对截止期.注意到,超周期 H 是关于规模 n 的指数函数,因此以上可调度性分析算法是指数时间的.Baruah 等人[5]已经从理论上证明了异步系统可调度性分析问题是强 NP 困难的,并且该结论对于利用率 U 严格小于 1 的系统也同样成立.目前,异步系统的可调度性分析理论主要围绕着充分性判定条件展开研究[7],致力于提出一些(伪)多项式时间的近似判定算法,对于任意异步系统,若算法回答“是”,则说明系统可被 EDF 成功调度;然而,当算法回答“否”时,却无法断定系统的可调度性.异步系统可调度性分析算法的相关研究详见第 2 节的介绍.

对于同步系统,定理 1 中的结论可进一步简化,如定理 2 所述.

定理 2[5]. 同步任务系统 T 在单处理器上是可调度的,当且仅当$\forall L\leqslant L^*$:$df(0,L)\leqslant L$.

与定理 1 相比,定理 2 将需求函数由原先的二元参变量 t_1 和 t_2 降为一元参变量 L,并给定变量 L 的上界 L^*.根据定理2,同步系统的可调度分析算法只需检查每个小于等于L^*的L,计算相应的一元需求函数$df(0,L)$,并判断$df(0,L)\leqslant L$是否成立.在之前的文献中,也有学者将L^*称为可调度性分析的上界[8],将需求函数$df(0,L)$重新定义为需求上界函数(demand bound function,简称 DBF) $dbf(L)=\sum_{i=1}^{n}\max\left\{\left\lfloor (L-D_i)/P_i\right\rfloor+1,0\right\}C_i$ [1].易知,计算 $dbf(L)$仅需要 $O(n)$的时间复杂度,可调度性分析问题的复杂性主要取决于上界 L^*的值.Baruah 等人[5]研究发现,对于利用率 U 严格小于 1 的同步系统,L^*的取值具有伪多项式上界,这意味着在 U<1 时,定理 2 蕴含的可调度性分析算法是 PTDA 算法.之后的研究均围绕着如何进一步优化 PTDA 算法性能而展开,或者缩小可调度性分析上界 L^*,或者减少 $dbf(L)$的计算次数.在接下来的一节将重点介绍这部分的研究进展.

2 相关工作

在过去的几十年间,众多学者为实时任务系统建立了各式各样的形式化模型,为调度算法分析提供了理论基础[1].这些形式化模型,从偶发性任务模型[3]到任务图模型[13,14],再到时间自动机[15,16],无一不在易解性和可表达性之间寻求一个折中.之前的研究通常把是否具有 PTDA 算法作为衡量一个系统模型是否易解的标准.针对每一类系统模型,专家学者大都致力于探究 PTDA 算法的存在性,并努力尝试提出 PTDA 算法.本文所研究的偶发性实时任务模型(具体定义见第 1.1 节)是最经典的形式化模型之一,迄今为止,至少有数十位学者为此类系统上 PTDA 算法的研究做出贡献.

2.1 同步系统可调度性分析问题研究

1990 年,Baruah 等人[5]研究发现,利用率 U<1 的任意截止期同步系统存在 PTDA 算法,并最先确定了可调度分析上界 L^*的范围:

$$L^*<L_a^1=\max\left\{D_1,...,D_n,\max_{1\leqslant i\leqslant n}\{P_i-D_i\}\frac{U}{1-U}\right\}.$$

1996 年,Ripoll 等人[17]在基于约束截止期的假设下,进一步缩小了可调度性分析上界,该上界很快被扩展到任意截止期的同步系统中[18]:

$$L^*<L_a^2=\max\left\{D_1,...,D_n,\frac{\sum_{i=1}^{n}(P_i-D_i)U_i}{1-U}\right\}.$$

另外,Spuri[19]和 Ripoll 等人[17]也给出了可调度性分析上界的递推方程形式:

$$\omega^0=\sum_{i=1}^{n}C_i,\ \omega^{m+1}=\sum_{i=1}^{n}\left\lceil\frac{\omega^m}{P_i}\right\rceil C_i,$$

以上公式从 ω^0 开始迭代,直到求得不动点 $\omega^{m+1}=\omega^m$ 时停止,并令可调度性分析上界 $L^*<L_b=\omega^{m+1}$.

2009 年,Zhang 等人[8]又改进了 L^*的上界值:

$$L^*<L_a^3=\max\left\{(D_1-P_1),...,(D_n-P_n),\frac{\sum_{i=1}^{n}(P_i-D_i)U_i}{1-U}\right\}.$$

注意到,以上可调度性分析上界的值均限定在一个关于 U 的伪多项式界内.传统判定算法在最坏情况下需要检查小于等于 L^*的所有 L,并逐个计算需求上界函数 $dbf(L)$.尽管 L^*是伪多项式时间界的,但实际计算时间依然庞大.Zhang 等人[8,20]充分利用 DBF 函数的非递减性质,提出了一种能够跳跃搜索的快速处理器需求分析(QPA)方法,从而避免对 1~L^*上界中所有可能的 L 进行枚举式的检查.实验结果表明[8],QPA 算法能够成指数级地降低 $dbf(L)$的迭代次数.如,对于某个包含 16 个任务的算例,传统枚举式算法需要迭代计算 $dbf(L)$高达 858 331 次,QPA 算法只需 12 次迭代即可完成可调度性判定.QPA 算法是目前判定同步系统可调度性最优秀的算法.

我们研究发现,QPA 算法的性能受到利用率 U 取值的影响,且算法的时间复杂性存在一个常数-指数突变的临界点.对于 $U\leqslant$0.99 的系统,QPA 算法中需求上界函数 $dbf(L)$的计算迭代次数不会超过 100 次,算法具有接近

常数时间的复杂度.然而,当 U>0.99 后,QPA 计算 $dbf(L)$的迭代次数呈指数增长,算法迅速退化为指数时间的复杂度.有研究表明,利用率 U 趋于 1 的同步系统可调度性分析属于弱 coNP 困难问题[7],理论上存在 PTDA 算法,但至今未有学者明确给出此类算法.该问题仍被列为实时调度领域的 5 大开放性课题之一[1].本文即针对这些困难问题开展研究,实验结果表明,第 3 节和第 4 节提出的模型和算法对大部分 U>0.99 的困难算例都有效(约 70%),与 QPA 算法相比,能够成指数倍地降低算法迭代次数.

2.2 异步系统可调度性分析问题研究

异步实时任务中的可调度性分析问题是强 coNP 困难问题,该结论甚至对于利用率 U 严格小于 1 的系统也成立[5].这意味着,同步系统中所有在 U<1 约束下建立起来的 PTDA 算法(包括 QPA 算法)均不再适用.目前,异步系统可调度性分析的研究主要关注充分性判定条件[12].其中,代表性的充分判定算法将每个任务 τ_i 的初始相位 φ_i 直接忽略,进而将原系统当作同步系统考虑[5].Baruah 等人[5]指出,给定一个异步系统 T,若其相对应的同步系统 T'是可调度的,则原系统 T 也是可调度的.但是,若 T'是不可调度的,T 的可调度性则不能确定.

注意到,同步系统中 PTDA 的研究成果扩展到异步系统所面临的主要困难在于,异步系统中不存在一个伪多项式界的可调度分析上界 L^*.即使将系统利用率限定为 U<1,上界 L^*在理论上依然是指数规模(Φ+2H)[5].但是,如果能从理论和实验上证明,对于大多数的异步系统,可调度性分析上界 L^*的取值实际上远远小于 Φ+2H,落在 QPA 等 PTDA 算法能够快速处理的范围内,则同步系统中的众多 PTDA 算法结果均可应用于求解异步系统的可调度性分析问题.本文应用整数规划理论,尝试为异步系统找到一个较小的可调度分析上界 L^*,可看作是将同步系统 PTDA 算法扩展到异步系统中的第 1 步研究.

3 整数规划模型及多面体分析

根据第 1.3 节中介绍的可调度判定基本原理,可调度性判定问题本质上是对解空间的搜索问题.即在一个时间域内不断搜索可能发生溢出(overflow)的时间段或时间点,直到找到溢出点,判定问题回答为“否”(不可调度);或者穷尽了整个时间域也未发现溢出,判定问题回答为“是”(可调度).为了更深刻地理解此类搜索问题,本节应用整数规划(integer programming,简称 IP)模型重新刻画可调度性判定问题.整数规划理论在对问题解空间的描述和分析层面都具有独特优势.本节的工作将从优化理论角度给出可调度性判定问题的一些新性质.这些新的性质和实时调度领域的既有结论和算法相结合,将有助于提出新的可调度性判定算法,提高算法的求解效率.

与以往研究不同,本文并非为可调度性判定问题直接建立 IP 模型,而是基于分而治之的思想,先对整个时间域[0,Φ+2H](异步系统)或[0,L^*](同步系统)进行划分,得到若干互不相交的时间子域[Q_1,Q_2),[Q_2,Q_3),…,[Q_l,Q_{l+1}],其中 Q_i 和 Q_{i+1} 分别表示第 i 个子域的下界和上界(1≤i≤l).易知,Q_1=0,Q_{l+1}=Φ+2H(或 L^*).另外, Q_i 和 Q_{i+1} 的选取还需满足:不存在任何任务 τ_k,使得 Q_i<D_k+φ_k<Q_{i+1} 成立.在每个时间子域[Q_i,Q_{i+1})中,判断是否存在时间点违背定理 1(在同步系统中违背定理 2)的问题将对应一个最优化问题.最优化问题的解与 0 的小于和大于关系分别蕴含原判定问题回答“是”或“否”.若所有时间域关联的最优解均小于 0,则说明系统可调度;否则,只要存在一个时间域关联的最优解大于等于 0,则说明系统不可调度.因此,原可调度性判定问题被分解为若干个时间子域对应的最优化问题.这些子问题具有相似的解空间结构,本文为所有子问题建立统一的整数规划模型.接下来分别从异步和同步系统两方面入手,给出时间子域[Q_i,Q_{i+1})上可调度性判定问题的 IP 模型.

3.1 异步系统中可调度判定问题的IP模型和理论分析

在异步系统中,给定时间域对应的可调度性判定子问题定义如下.

定义 2. 给定时间域[Q_i,Q_{i+1}),异步系统中相应的可调度性判定问题是:在时间域中寻找两个时间点 t_1 和 t_2,满足(Q_i≤t_1<t_2<Q_{i+1}),使得 $df(t_1,t_2)$>t_2–t_1 成立.若成功地查找到 t_1 和 t_2,则系统不可调度;否则,系统可调度.

接下来为定义 2 中的判定问题建立相应的 IP 模型,即异步系统可调度性分析问题(schedulability test for asychronous system,简称 STAS)的 IP 模型,简记为 STAS-IP 模型.首先,STAS-IP 模型中用到的所有常量在本文第 1.1 节均已定义,在此不再赘述.下面主要介绍 STAS-IP 模型中用到的 3 类变量,及其相关的约束条件.

首先,给出时间变量 t_1 和 t_2.定义 2 中在时间域 $[Q_i,Q_{i+1})$ 内要寻找的两个时间点分别定义为 t_1 和 t_2.在 STAS-IP 模型中,时间变量被定义为非负实数变量:$t_1,t_2\in R^{\geqslant 0}$.另外,根据定义 2 可知,$t_1$ 和 t_2 需满足约束 $Q_i\leqslant t_1<t_2<Q_{i+1}$.因此,构造以下约束:

$$t_1 < t_2 \tag{1}$$

$$t_2 < Q_{i+1} \tag{2}$$

$$t_1 \geqslant Q_i \tag{3}$$

约束条件(1)限定了时间点 t_1 和 t_2 的序关系,即 t_1 要严格小于 t_2.约束条件(2)保证了时间点 t_2 在 Q_{i+1} 之前取值.约束条件(3)保证 t_1 取 Q_i 之后的值.约束式(1)-(3)联立,共同保证了 t_1 和 t_2 可以取遍时间域 $[Q_i,Q_{i+1})$ 内的所有时间点,且满足 $t_1<t_2$.

注意到,在划分时间域时,本文规定 Q_i 和 Q_{i+1} 需满足条件:不存在任何任务 τ_k,使得 $Q_i<D_k+\varphi_k<Q_{i+1}$ 成立.换句话说,对于给定的 Q_i 和 Q_{i+1},系统中任何任务 τ_k 均不满足 $Q_i<D_k+\varphi_k<Q_{i+1}$.因此,系统中的任务被划分为两个集合 $T_1=\{\tau_k|D_k+\varphi_k\leqslant Q_i\}$ 和 $T_2=\{\tau_k|D_k+\varphi_k\geqslant Q_{i+1}\}$.易知, T_2 中任务所释放的作业其截止期落在时间点 Q_{i+1} 之外,一定不会影响需求函数 $df(t_1,t_2)$ 的值.故 T_2 可以直接从 STAS-IP 模型中排除,而只需考虑 T_1 中的任务.

接下来,给出 x_k 变量.为 T_1 中的每个任务 $\tau_k(1\leqslant k\leqslant n)$ 分配一个非负整数变量 x_k,表示 τ_k 在时间段 $[0,t_2-D_k]$ 内应该完成的作业个数.易知,在时间段 $[0,t_2-D_k]$ 内,任务 τ_k 释放并且必须完成的作业个数至多为 $\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor$.因此,有 $x_k\leqslant\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor$ 成立.更进一步地,可知如下线性不等式成立.

$$P_k x_k + D_k + \varphi_k \leqslant t_2,\ \forall \tau_k : D_k + \varphi_k \leqslant Q_i \tag{4}$$

约束条件(4)给出了 x_k 的线性上界 $x_k\leqslant(t_2-\varphi_k-D_k)/P_k$.由于 x_k 是整数变量,所以 x_k 至多取到 $\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor$.

最后,给出 y_k 变量.为 T_1 中的每个任务 $\tau_k(1\leqslant k\leqslant n)$ 分配一个非负整数变量 y_k,表示 τ_k 在时间段 $[0,t_1]$ 内最多释放的作业个数.易知,任务 τ_k 在 $[0,t_1]$ 内最多释放 $\lceil(t_1-\varphi_k)/P_k\rceil$ 个作业,即 $y_k=\lceil(t_1-\varphi_k)/P_k\rceil$.易知,$y_k\geqslant(t_1-\varphi_k)/P_k$ 成立,如约束条件(5)所示.约束(5)给出了变量 y_k 的线性下界.

$$P_k y_k + \varphi_k \geqslant t_1,\ \forall \tau_k : D_k + \varphi_k \leqslant Q_i \tag{5}$$

注意到,变量 x_k 的整数上界和 y_k 的整数下界分别为 $\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor$ 和 $\lceil(t_1-\varphi_k)/P_k\rceil$.因此,$\max\{x_k-y_k+1\}=\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor-\lceil(t_1-\varphi_k)/P_k\rceil+1$.这恰好与 $\eta_k(t_1,t_2)$ 定义中的第 1 项相对应.以下引理给出了 STAS-IP 模型中变量集合 $\{x_k,y_k|D_k+\varphi_k\leqslant Q_i\}$ 与 $\eta_k(t_1,t_2)$ 的联系.

引理 3. 在时间域 $[Q_i,Q_{i+1})$ 中,对于任意给定的时间段 $[t_1,t_2]$,有 $\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)\leqslant\sum_{k=1}^{n}\eta_k(t_1,t_2)C_k$.

证明:根据定义 1,$df(t_1,t_2)$ 计算公式 $\sum_{k=1}^{n}\eta_k(t_1,t_2)C_k$ 的展开式为 $\sum_{k=1}^{n}\max\left\{\left\lfloor(t_2-\varphi_k-D_k)/P_k\right\rfloor-\left\lceil(t_1-\varphi_k)/P_k\right\rceil+1,0\right\}C_k$.其中,$\eta_k(t_1,t_2)$ 的非线性计算公式由一个最大化函数 max()构成,其取值分两种情况.

• 当 $D_k+\varphi_k\leqslant Q_i$ 时,$\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor+1$ 即为任务 τ_k 在时间段 $[0,t_2]$ 内释放并完成的最多作业个数,$\lceil(t_1-\varphi_k)/P_k\rceil$ 为任务 τ_k 在时间段 $[0,t_1]$ 内释放的最多作业个数,例如图 1 中的任务 τ_1.已知 $t_1<t_2$,有 $\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor+1\geqslant\lceil(t_1-\varphi_k)/P_k\rceil$ 成立.故 $\eta_k(t_1,t_2)=\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor-\lceil(t_1-\varphi_k)/P_k\rceil+1$.

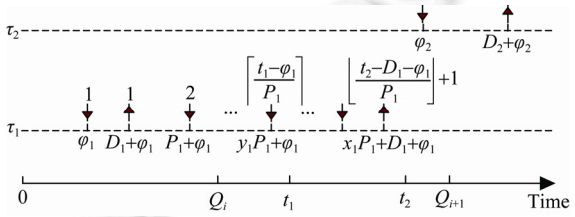


Fig.1 Illustration for the different two values of $\eta_k(t_1,t_2)$

图 1 $\eta_k(t_1,t_2)$两种取值情况的示意图

• 当 $D_k+\varphi_k\geqslant Q_{i+1}$ 时,$\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor+1\leqslant 0$,例如图 1 中的任务 τ_2.易知,$\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor+1-\lceil(t_1-\varphi_k)/P_k\rceil\leqslant 0$.故 $\eta_k(t_1,t_2)=0$.

注意到,由于 Q_i 和 Q_{i+1} 的取值限制,$Q_i<D_k+\varphi_k<Q_{i+1}$ 的情况已被排除,故无需讨论.

综上可得 $\sum_{k=1}^{n}\eta_k(t_1,t_2)C_k=\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}\eta_k(t_1,t_2)C_k$,即 $df(t_1,t_2)=\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor-\lceil(t_1-\varphi_k)/P_k\rceil+1$.根据 STAS-IP 模型中变量定义可知,对于满足 $D_k+\varphi_k\leqslant Q_i$ 的任务 τ_k,x_k+1 是$[0,t_2]$内释放并完成的作业个数,y_k 是 $[0,t_1]$ 内释放的最多作业个数.易知,$x_k-y_k+1\leqslant\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor-\lceil(t_1-\varphi_k)/P_k\rceil+1\leqslant\eta_k(t_1,t_2)$ 成立.故有 $\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)\leqslant\sum_{k=1}^{n}\eta_k(t_1,t_2)C_k$ 成立.

易知,当每个任务都尽可能多地释放作业时,引理 3 中不等式取等.故有 $\max\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)=df(t_1,t_2)$.

基于以上变量和约束条件的定义,给出 STAS-IP 模型的目标函数如下:

$$\min\left(t_2-t_1-\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)\right) \tag{6}$$

目标函数(6)的主体部分可看作两个子式的差.子式①t_2-t_1 是时间段$[t_1,t_2]$的长度;子式②$\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)$是时间段$[t_1,t_2]$对应的处理器需求.由引理 3 可知,$\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)\leqslant df(t_1,t_2)$.注意到,子式②在目标函数(6)中的系数为–1,在最小化函数 min()的作用下,实际等价为计算时间段$[t_1,t_2]$上的最大需求,即 $\max\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)=df(t_1,t_2)$.

由定义 1 可知,$df(t_1,t_2)$由 $\eta_k(t_1,t_2)$的线性加和构成($1\leqslant k\leqslant n$).由于 $\eta_k(t_1,t_2)$是非线性的,故 $df(t_1,t_2)$也是非线性的.在 STAS-IP 模型中,非线性的 $df(t_1,t_2)$被等价为线性式 $\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)$ 的最大化形式.之所以能够实现这种转化,是由 $\eta_k(t_1,t_2)$的定义所决定的.对于任意的任务 $\tau_k(1\leqslant k\leqslant n)$,$\eta_k(t_1,t_2)$被定义为$\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor-\lceil(t_1-\varphi_k)/P_k\rceil+1$ 和 0 两者取大.由引理 3 的证明过程可知,若 $\tau_k\in T_1$,则 $\eta_k(t_1,t_2)=\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor-\lceil(t_1-\varphi_k)/P_k\rceil+1$;若 $\tau_k\in T_2$,则 $\eta_k(t_1,t_2)=0$.又知,$T_1\cup T_2=\{\tau_k|1\leqslant k\leqslant n\}$.因此,$\eta_k(t_1,t_2)$的非线性的离散取值情况对应为任务集合的一个划分 $T_1\cup T_2$.由于 $\eta_k(t_1,t_2)=0$ 对计算需求函数 $df(t_1,t_2)$没有贡献,所以只需考虑 $\eta_k(t_1,t_2)=\lfloor(t_2-\varphi_k-D_k)/P_k\rfloor-\lceil(t_1-\varphi_k)/P_k\rceil+1$ 的情况,即只需考虑 T_1 中的任务.由目标函数(6)可知,STAS-IP 模型恰好只考虑任务子集合 T_1,而将 T_2 排除在外.用线性式 $\max\sum_{\tau_k:D_k+\varphi_k\leqslant Q_i}C_k(x_k-y_k+1)$ 即可替代定义 1 中包含非线性式 $\eta_k(t_1,t_2)$的需求函数 $df(t_1,t_2)$.STAS-IP 模型的线性目标函数(6)和线性约束不等式(1)~(5)使得 STAS-IP 模型成为易于处理的整数线性规划.

注意到,目标函数(6)即保证最优解对应的时间段长度与其相应的最大处理器需求之差最小.易知,若 STAS-IP 模型求得最优解对应的目标值小于 0,即存在一个时间段$[t_1,t_2]$,使得 $df(t_1,t_2)>t_2-t_1$ 成立,则说明系统不可调度;若最优解对应的目标值大于等于 0,则说明不存在产生溢出的时间段,即系统可调度.

定理 4. STAS-IP 模型是多项式规模的整数线性规划模型.

证明:令 $m=|\{\tau_k|D_k+\varphi_k\leqslant Q_i\}|$.易知,STAS-IP 模型中的变量个数是 $2m+2$,约束集合(1)~(5)中不等式的个数之和为 $2m+3$.由于 m 是小于 n 的整数,显然,STAS-IP 模型是多项式规模的.另外,目标函数(6)是线性的,并且约束集合(1)~(5)均为线性不等式,故 STAS-IP 模型属于整数线性规划模型. □

接下来对 STAS-IP 模型的解空间进行精确理论分析.假设 STAS-IP 模型的变量规模为 $2m+2$.STAS 问题的任意可行解均关联一个整数向量 $f=(t_1,t_2,x_1,\ldots,x_m,y_1,\ldots,y_m)$,其中,非负整数 t_1 和 t_2 是 f 关联的时间段下界和上界值,非负整数 x_k 是 f 中第 k 个任务 τ_k 在时间段$[0,t_2-D_k]$内完成的作业个数,非负整数 y_k 是任务 τ_k 在时间段$[0,t_1]$内释放的作业个数($1\leqslant k\leqslant m$).显然,STAS-IP 模型中每个可行解的关联向量 f 满足约束不等式(1)~(5);反之,每个满足约束不等式(1)~(5)的向量也一定是 STAS-IP 模型的可行解.因此,可由约束不等式(1)~(5)定义 STAS-IP 模型的解空间多面体 $P_{\mathrm{STAS}}(T_m)$,其中 $T_m=\{\tau_k|\tau_k\in T\wedge D_k+\varphi_k\leqslant Q_i\}$.首先,多面体 $P_{\mathrm{STAS}}(T_m)$对应的可行解集合 F 定义如下:

$$F=\{f\in N^{2m+2}|f\text{ 满足约束不等式(1)~(5)}\}.$$

以上表达说明,STAS-IP 的多面体可定义为所有可行解构成的凸包(convex hull):$P_{\mathrm{STAS}}(T_m)=conv(F)$.下面给出 $P_{\mathrm{STAS}}(T_m)$的维数和极大诱导不等式的理论结果.

定理 5. 若$\forall\tau_k\in T_m$:$P_k+D_k+\varphi_k\leqslant Q_{i+1}-1$,则 STAS-IP 多面体 $P_{\mathrm{STAS}}(T_m)$的维数为 $2m+2$.

证明:欲证多面体维数等于 $2m+2$,只需在 F 中找出 $2m+3$ 个仿射无关向量.首先,给出基向量 $f_0=(Q_i,Q_{i+1}-1,0,...,0,K,...,K)\in N^{2m+2}$,其中 K 为变量 y_k 的取值,是一个使约束(5)成立的常数.另外,0 和 K 的个数为 m.然后,给出剩余的 $2m+2$ 个可行解关联向量.

• 集合 F_1 由两个关联向量组成:$F_1=\{(Q_i+1,Q_{i+1}-1,0,..,0,K,...,K),(Q_i,Q_{i+1}-2,0,...,0,K,...,K)\}$.

• 集合 F_2 包含 m 个向量:$F_2=\{(Q_i,Q_{i+1}-1,1,0,...,0,K,...,K),(Q_i,Q_{i+1},0,1,0,...,0,K,...,K),...,(Q_i,Q_{i+1}-1,0,...,1,K,...,K)\}$,其中,第 k 个向量关联的 x_k 变量取值为 1.

• 集合 F_3 包含 m 个向量:$F_3=\{(Q_i,Q_{i+1}-1,0,...,0,K+1,K,...,K),(Q_i,Q_{i+1}-1,0,...,0,K,K+1,K,...,K),...,(Q_i,Q_{i+1}-1,0,...,0,K,...,K+1)\}$,其中,第 k 个向量关联的 y_k 变量取值为 $K+1$.

易知,将 F_1,F_2 和 F_3 中的每个向量减去基向量 f_0 后得到 $2m+2$ 个线性无关的向量.所以,向量集合 $\{f_0\}\cup F_1\cup F_2\cup F_3$ 中的 $2m+3$ 个关联向量仿射无关. □

定理 6. 若存在两个时间点 $Q_i\leqslant t_2<t_3<Q_{i+1}$ 和一个任务 $\tau_k\in T_m$,使得 $P_kx_k+D_k+\varphi_k=t_2\wedge P_k(x_k+1)+D_k+\varphi_k=t_3$,并且 $\forall\tau_k\in T_m:P_k+D_k+\varphi_k\leqslant t_2$,则 STAS-IP 模型中约束不等式(4)是 $P_{\mathrm{STAS}}(T_m)$的极大诱导不等式.

证明:根据定理 5,$P_{\mathrm{STAS}}(T_m)$的维数为 $2m+2$.对于约束(4)中的不等式 $P_kx_k+D_k+\varphi_k\leqslant t_2$,只需在 F 中找出 $2m+2$ 个令等号成立的仿射无关向量,即可证明该不等式为极大诱导不等式.首先给出基向量 $f_0=(Q_i,t_2,0,...,x_k,...,0,K,...,K)$.在 f_0 的基础上,构造向量 $f_1=(Q_i+1,t_2,0,...,x_k,...,0,K,...,K)$.易知,$f_0$,$f_1$ 均使不等式取等.接下来,分两步构造剩余的 $2m$ 个使不等式取等的向量.

• 集合 F_1 包含 m 个向量:$F_1=\{(Q_i,t_2,1,0,...,x_k,...,0,K,...,K),...,(Q_i,t_3,0,...,x_k+1,...,0,K,...,K),...,(Q_i,t_2,0,...,x_k,...,1,K,...,K)\}$.

• 集合 F_2 包含 m 个向量:$F_2=\{(Q_i,t_2,0,...,x_k,...,0,K+1,K,...,K),...,(Q_i,t_2,0,...,x_k,...,0,K,K+1,K,...,K),...,(Q_i,t_2,0,...,x_k,...,0,K,...,K+1)\}$.

将 $\{f_1\}$,F_1 和 F_2 中每个向量减去基向量 f_0 后得到 $2m+1$ 个线性无关的向量.所以,向量集合 $\{f_0,f_1\}\cup F_1\cup F_2$ 中的 $2m+2$ 个关联向量仿射无关. □

应用类似的构造证明技术不难推出以下结论,具体证明过程在此不再赘述.

推论 7. 若存在两个时间点 $Q_i\leqslant t_1<t'_1<Q_{i+1}$ 和一个任务 $\tau_k\in T_m$,使得 $P_ky_k+\varphi_k=t_1\wedge P_k(y_k+1)+\varphi_k=t'_1$,并且 $\forall\tau_k\in T_m:P_k+D_k+\varphi_k\leqslant t_1$,则 STAS-IP 模型中约束不等式(5)是 $P_{\mathrm{STAS}}(T_m)$的极大诱导不等式.

定理 6 和推论 7 说明,约束(4)和(5)是可调度性判定问题非常紧的约束条件.在多面体理论中,极大诱导不等式对应解空间的精确边界.根据最优化理论,问题最优解总是在边界交叉点上取得.也就是说,最优解很有可能在约束不等式(4)和(5)取等时取得.本文将在第 5.1 节通过实验验证这些约束不等式的有效性.

3.2 同步系统中可调度判定问题模型和理论分析

上一节以异步系统为例,给出了可调度性判定问题一般化的 STAS-IP 模型.本节应用相似的技术,为同步系统建立 IP 模型.与 STAS-IP 模型相比,本节模型将更加简化.下面先给出同步系统中可调度性判定问题的定义.

定义 3. 给定时间域$[Q_i,Q_{i+1})$,同步系统中相应的可调度性判定问题是:在时间域中寻找时间点 t,$(Q_i\leqslant t<Q_{i+1})$,使得 $dbf(t)>t$ 成立.若成功查找到 t,则系统不可调度;否则,系统可调度.

根据定义 3 可知,同步系统对应的 IP 模型只需两类变量:① 时间点变量 t:定义 3 中要寻找的时间点;② 每个任务 τ_k 对应一个变量 x_k,表示 τ_k 在时间段$[0,t-D_k]$内应该完成的作业个数.

根据定理 2,建立同步系统可调度性分析问题(schedulability test for sychronous systems,简称 STSS)的 IP 模型如下.

Model STSS-IP:

$$\min\left(t-\sum\nolimits_{\tau_k:D_k\leqslant Q_i}C_k(x_k+1)\right) \tag{7}$$

s.t.

$$t<Q_{i+1} \tag{8}$$

$$t \geqslant Q_i \tag{9}$$

$$P_k x_k + D_k \leqslant t,\ \forall \tau_k : D_k \leqslant Q_i \tag{10}$$

$$x_k, y \in N^{\geqslant 0},\ \forall \tau_k : D_k \leqslant Q_i \tag{11}$$

显然,STSS-IP 模型是 STAS-IP 模型在考虑 Φ=0 时的简化.目标函数(7)使得时间段[0,t]的长度与其对应的需求上界 $dbf(t)$的差值最小.约束(8)和(9)保证了时间点变量 t 落在时间域[Q_i,Q_{i+1})中.约束(10)是 STAS-IP 模型中约束(4)考虑 φ_k=0 后的结果,给出了变量 x_k 的上界.易知,STSS-IP 模型的变量和线性约束不等式的个数分别为 m+1 和 m+2.另外,采用类似的证明技术,第 3.1 节中的多面体理论结果不难扩展到同步系统中,具体证明过程略.

推论 8. 若$\forall \tau_k \in T_m$:$P_k+D_k<Q_{i+1}$,则 STSS-IP 多面体 $P_{STSS}(T_m)$的维数为 m+1.

推论 9. 若存在两个时间点 $Q_i \leqslant t<t'<Q_{i+1}$ 和一个任务 $\tau_k \in T_m$,使得 $P_k x_k+D_k=t \wedge P_k(x_k+1)+D_k+\varphi_k=t'$,并且$\forall \tau_k \in T_m$: $P_k+D_k \leqslant t$,则 STSS-IP 模型中约束不等式(10)是 $P_{STSS}(T_m)$的极大诱导不等式.

4 线性松弛算法

上一节将原可调度性判定问题分解为若干子问题,为每个子问题建立了 IP 模型.本节将通过求解这一系列子问题的 IP 模型来给出原问题的判定结果.具体方法如算法 1 所示.

算法 1. 同/异步系统可调度性判定问题的线性松弛算法.

输入:任务系统 $T=\{\tau_1,\ldots,\tau_n\}$及可调度性分析上界:$L$(在同步系统中, $L = L_a^3$; 在异步系统中,$L=\Phi+2H$).

输出:任务系统 T 是否可调度.

BEGIN

1. 将任务按照 $D_k+\varphi_k$ 递增顺序进行排列,得到有序任务序列:$\tau_{[1]},\ldots,\tau_{[n]}$;
2. l:=1; Q_l:=0;
3. REPEAT $1 \leqslant k \leqslant n$:
4. IF $\tau_{[k]}$关联的 $D_{[k]}+\varphi_{[k]}>Q_l$ THEN l:=l+1; Q_l:=$D_{[k]}+\varphi_{[k]}$;
5. Q_{l+1}:=L; *feasible*:=0;
6. FOR each [Q_l,Q_{l+1}):
7. 将时间段[Q_l,Q_{l+1})上关联的 STAS-IP/STSS-IP 模型中整数约束拿掉,求解其线性松弛模型;
8. 得到线性松弛解 f_l,其对应的目标值为 LP_l;
9. 将松弛解 f_l 取整,得到原 IP 模型中的可行解,其对应目标值为 FS_l;
10. IF $LP_l>0$ THEN *feasible*:=*feasible*+1;
11. ELSE IF $FS_l<0$ THEN RETURN 系统 T 不可调度;
12. IF *feasible*=l THEN RETURN 系统 T 可调度;
13. ELSE RETURN 系统可调度性不确定;

END

算法 1 为同步和异步两类实时任务系统的可调度性判定问题提供了统一的求解框架,主要分为两部分.

① 算法在第 1 行~第 5 行将时间域[0,L]划分为 l 个互不相交的子时间域.注意到,每个子域[Q_i,Q_{i+1})的下界和上界取值要么等于某个 $D_k+\varphi_k$(见算法第 3 行和第 4 行),要么等于 0 或 L;显然满足第 3 节中规定的约束条件:不存在任何任务 τ_k,使得 $Q_i<D_k+\varphi_k<Q_{i+1}$ 成立.易知,原可调度性判定问题可转化为 l 个定义在时间域[Q_i,Q_{i+1})上的子问题($1 \leqslant i \leqslant l$).

② 算法在第 6 行~第 11 行首先将每个子问题对应的 IP 模型进行线性松弛处理,即去掉整数变量约束,进而得到一个易于求解的线性规划(linear programming,简称 LP)模型.求解这个线性松弛模型,得到一个松弛最优解 f_l,其对应的解向量中可以包含小数.通过对 f_l 向量取整,可以得到原 IP 模型的一个可行解.根据最优化理论,线性松弛解 f_l 的值 LP_l 一定小于等于 IP 模型的最优解,可行解的值 FS_l 一定大于等于 IP 模型的最优解.因此,

- 若 $LP_l>0$,则 IP 模型最优解一定大于 0,可得系统 T 在时间域[Q_i,Q_{i+1})内可调度.在算法中,逐个验证每个子

时间域,只要发现系统 T 在该时间域内可调度,变量 *feasible* 就会自增 1(见算法第 10 行).如果系统 T 在 l 个子时间域内均可调度,则 *feasible* 的值等于 l,即可判定系统 T 在整个时间域上可调度(见算法第 12 行).

• 否则,若 $LP_l \leqslant 0$,则 IP 模型的最优解与 0 的关系不确定,此时,若 f_l 取整得到的可行解的值 $FS_l<0$,则 IP 模型的最优解一定小于 0,可知系统 T 不可调度(见算法第 11 行).

定理 10. 算法 1 的计算复杂性是多项式时间的.

证明:由算法第 3 行和第 4 行可知,原可调度性判定问题至多被划分为 n 个子问题,每个子问题对应一个线性松弛问题.又知,线性松弛问题是多项式时间可解的,因此,算法 1 是多项式时间的. □

根据定理 10,算法 1 的迭代次数被限定在多项式规模内,在最坏情况下,至多需要迭代 n 次.注意到,同步系统中存在很多优良的算法理论结果,如 QPA 算法,与这些已有理论成果相结合,能够进一步减少算法 1 的迭代次数,提高算法的执行效率.

4.1 同步系统中算法性能的优化策略

在同步系统中,QPA 算法能够在时间域$[0,L]$中实现跳跃式搜索.这主要是基于以下结论[8]:若时间点 t 关联的需求上界函数值 $dbf(t)<t$,则在时间域$(dbf(t),t]$内不可能存在溢出点.因此,可以跳过这一区域,在搜索过 t 点后,直接对时间点 $dbf(t)$进行搜索.可将该结论应用到算法 1 中,以减少需求求解的子问题个数.改进后的算法如下.

算法 2. 同步系统中改进的可调度性判定算法.

输入:任务系统 $T=\{\tau_1,\ldots,\tau_n\}$及可调度性分析上界 $L=L_a^3$.

输出:任务系统 T 是否可调度.

```
BEGIN
1.  将任务按照 D_k 递增顺序进行排列,得到有序任务序列:τ_[1],…,τ_[n];
2.  Q_low:=D_[n]; Q_high:=L; uncertain:=false;
3.  WHILE Q_high≥D_[1] DO
4.      求解时间段[Q_low,Q_high]关联的 STSS 线性松弛模型,得到松弛解 f,其对应的目标值为 LP;
5.      将松弛解 f 取整,得到原 STSS-IP 模型中的可行解,其对应目标值为 FS;
6.      IF LP<0 && FS<0 THEN RETURN 系统 T 不可调度;
7.        ELSE IF LP<0 && FS>0 THEN uncertain:=true;
8.      Q_high:=min{Q_low,dbf(Q_low)+1}; Q_low:=max{D_k|D_k<Q_high};
9.  IF uncertain=false THEN RETURN 系统 T 可调度;
10. ELSE RETURN 系统可调度性不确定;
END
```

算法 2 的第 8 行实现了本文方法与文献[8]中结论的结合.与算法 1 相比,算法 2 排除了那些一定不存在溢出点的时间段.如图 2 所示,算法 1 会对 l 个时间段对应的线性松弛模型逐个进行求解;算法 2 则可能在执行过程中跳过一些时间段,或者缩小时间段的长度,从而减少算法的迭代次数,并缩小问题的解空间.在第 5.3 节,将给出算法 2 与 QPA 算法的性能比较.

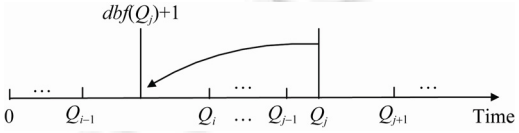


Fig.2 Algorithm 2 will search in $[Q_{i-1},dbf(Q_j)+1]$ after the completion of searching in $[Q_j,Q_{j+1}]$

图 2 算法 2 在搜索完区域$[Q_j,Q_{j+1}]$之后,搜索$[Q_{i-1},dbf(Q_j)+1]$

4.2 异步系统中对不确定情况的处理

注意到,本文算法的返回结果分为两种情况,一是明确指出系统 T 可调度或不可调度,二是不能确定系统的

可调度性.对于第 1 种返回情况的算例,本文算法是精确判定算法;对于第 2 种情况,本文算法虽然不能判定算法的可调度性,却能在一定程度上缩小可调度性分析上界.这在异步系统的可调度性分析中尤为明显.对于不能确定系统可调度性的算例,进一步应用以下算法,将给出一个较小的可调度性分析上界.

算法 3. 异步系统中对不确定情况的处理算法.

输入:可调度性不确定的任务系统 $T=\{\tau_1,\ldots,\tau_n\}$及可调度性分析上界 $L=\Phi+2H$.

输出:缩小的可调度性分析上界 L.

BEGIN

1. 将任务按照 D_k 递增顺序进行排列,得到有序任务序列:$\tau_{[1]},\ldots,\tau_{[n]}$;
2. $Q_{\text{low}}:=D_{[n]}$; $Q_{\text{high}}:=L$;
3. WHILE $Q_{\text{low}}<Q_{\text{high}}$ DO
4. 求解时间段$[Q_{\text{low}},Q_{\text{high}}]$关联的 STSS 线性松弛模型,得到松弛解 $f=(t,x_1,\ldots,x_m)$,其目标值为 LP;
5. IF $LP\geqslant 0$ Then $Q_{\text{high}}:=\lfloor(Q_{\text{low}}+t)/2\rfloor$;
6. ELSE $Q_{\text{low}}:=\lfloor(t+Q_{\text{high}})/2\rfloor$;
7. $L:=\max\{Q_{\text{low}},Q_{\text{high}}\}$;

END

算法 3 应用折半法思想,在时间域$[D_{[n]},\Phi+2H]$中确定离 $D_{[n]}$最近的时间点 L,使得时间段$[L,\Phi+2H]$对应的线性松弛解的值 $LP\geqslant 0$.易知,当 $LP\geqslant 0$ 时,时间段$[L,\Phi+2H]$中一定不存在溢出点,即可排除出搜索范围.因此,只有时间段$[0,L]$中才可能出现溢出点,L 即为新的可调度性分析上界.易知,算法 3 的计算复杂性为 $O(\log(\Phi+2H))$,即使 H 为指数规模,算法 3 也是多项式时间界的.第 5.4 节将通过实验验证算法 3 在降低可调度性分析上界方面的有效性.

5 实验结果及分析

为了验证本文算法的有效性,本节针对大量随机任务测试集进行实验.本文算法由 C++程序调用 Gurobi 5.6[21]函数库实现,程序运行环境为具有 2.2GHz 主频和 1G 内存的奔腾处理器.另外,随机任务集算例的生成方法详见第 5.1 节.

5.1 随机任务集生成算法

本节采用文献[8]中生成随机同步任务集的方法,并将其扩展到异步系统中.

• 任务利用率的生成策略:本节按照 0-1 间的均匀分布为任务集中的每个任务分配一个利用率 U.具体生成算法参见文献[22]中的 UUniFast 算法.该算法已被证明可以有效地生成呈均匀分布的任务利用率[22].

• 任务周期的生成策略:本文采用文献[23]的方法生成满足指数分布的任务周期.令 $P_{\max}:=\max_{1\leqslant i\leqslant n}\{P_i\}$和 $P_{\min}:=\min_{1\leqslant i\leqslant n}\{P_i\}$.为使所有任务的周期均匀分布在一个给定范围(如 $P_{\max}/P_{\min}$ 的最大值)内,首先将这个范围划分为 k 个子区域:$e^0\sim e^1,e^1\sim e^2,\ldots,e^{k-1}\sim e^k$;然后,在每个子区域中,随机生成$\lfloor(n-1)/k\rfloor$个任务周期.另外,剩余的$(n-1)\bmod k$ 个任务周期则在 $e^0\sim e^k$ 内随机分配.

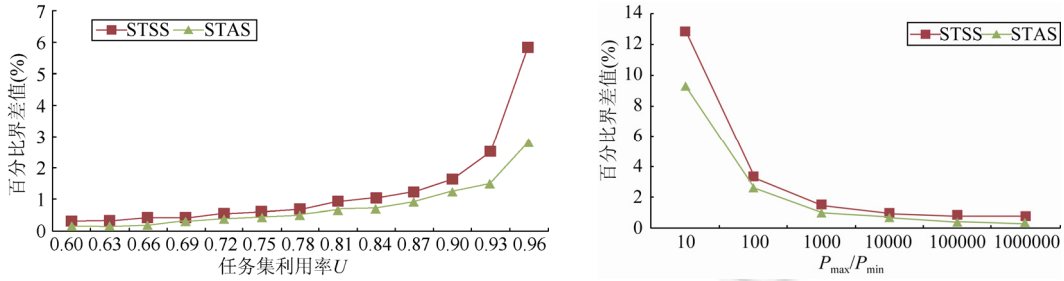
• 任务截止期和相位的生成策略:每个任务 τ_i 的截止期 D_i 在区域$[a,b]$中随机生成.其中,区域下界 a 的取值规则如下:当 τ_i 的执行时间 $C_i<10$ 时,$a:=C_i$;当 $10\leqslant C_i<100$ 时,$a:=2\times C_i$;当 $100\leqslant C_i<1000$ 时,$a:=3\times C_i$;当 $C_i\geqslant 1000$ 时,$a:=4\times C_i$.区域上界 b 的默认值为 $1.2\times P_i$.另外,在异步系统中,每个任务 τ_i 的相位 φ_i 通过在区域$[0,D_i]$中取随机数获得.

在本节所有实验中,对于每种问题参数取值,均生成了大约 6 000 个随机算例,并统计算法运行的平均情况.

5.2 极大诱导不等式的有效性分析

图 3 给出了本文的 STSS 和 STAS 线性松弛模型对各自最优整数解的逼近程度.这种逼近程度可由百分比界差 gap 进行评估.gap 的计算公式为$|(OPT-LP)/LP|\times 100\%$,其中,$OPT$ 为 IP 模型求得的最优整数解,LP 为 IP 模

型去掉整数约束后,得到的松弛最优解.百分比界差是数学规划领域常用的评价参数[24,25],具体可视为在去除问题参数取值的影响后,问题松弛解与最优解之间的绝对差距.注意到松弛模型中主要起作用的是极大诱导不等式约束,*gap* 值能够在一定程度上反映极大诱导不等式的有效性.



(a) 利用率 U 对 *gap* 的影响　　(b) $P_{\max}/P_{\min}$ 最大值对 *gap* 的影响

Fig.3　Values of *gap* affected by the problem parameters

图 3　问题参数对 *gap* 值的影响

本节对具有 30 个任务的测试集进行实验,分别调用 Gurobi 的单纯形算法和分支切割算法求解$[D_{[n]},L]$对应的线性松弛模型和 IP 模型.图 3(a)给出了同步和异步系统在 $P_{\max}/P_{\min}$:=1000,b=1.2×P_i 以及 U=0.6~0.96 时,*gap* 的取值情况.平均情况下,异步和同步系统的 *gap* 能够保持在 0.78%和 1.27%左右,这说明,线性松弛解距离最优解的偏差能够限制在一个很小的范围内.对于不确定的算例,直接依据松弛解判定系统可调度性的可信度为 99.23%(异步)和 98.73%(同步).另外,图 3(b)给出了同步和异步系统在 U=0.9,b=1.2×P_i 以及 $P_{\max}/P_{\min}$:=10~1000000 时,*gap* 的取值情况.异步和同步系统的平均 *gap* 值分别为 2.41%和 3.37%.

5.3　同步系统中算法性能分析

5.3.1　问题参数对 QPA 算法的影响

本节重现了 QPA 算法,研究了可调度分析上界 L 取值和利用率 U 的变化对算法性能的影响.我们发现,当 QPA 算法中 L 分别取 L_a^2 和 L_a^3 时,算法迭代次数呈现出一定的差异.如图 4 所示,当 $L=L_a^3$ 时,算法迭代次数不超过 100;然而,当 $L=L_a^2$ 时,算法迭代次数会超过 200 次,这在 U>0.99 后更为明显.根据定理 10 易知,本文算法的迭代次数不依赖于 L 的取值.

另外,对于 U≤0.99 的算例,QPA 算法的迭代次数不超过 100;当 U>0.99 时,QPA 算法的迭代次数呈指数增长,如图 5 所示(本节任务测试集的相关参数设定为:n=30,$P_{\max}/P_{\min}$:=1000,b=1.2×P_i).

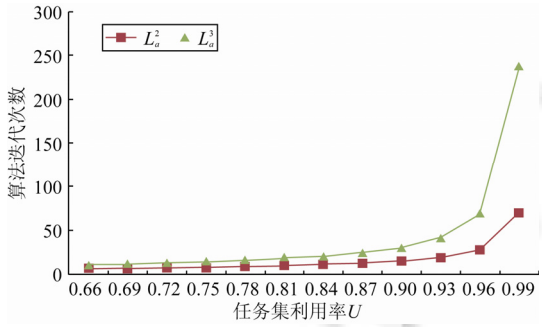


Fig.4　Iteration times of QPA affected by values of upper bound L

图 4　上界 L 不同取值对 QPA 算法迭代次数的影响

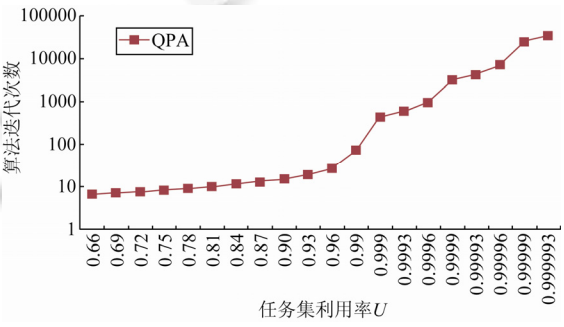


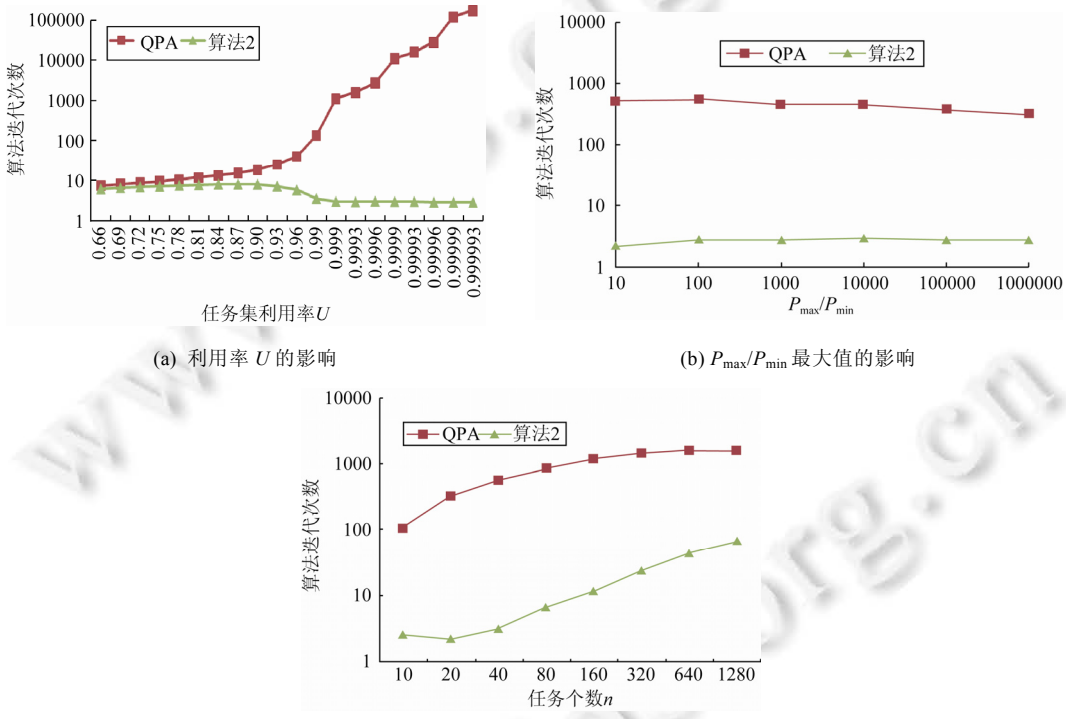
Fig.5　Iteration times of QPA affected by utilization U

图 5　利用率 U 对 QPA 算法迭代次数的影响

5.3.2 本文算法与 QPA 算法的性能比较

本文通过调用 gurobi 库函数中的单纯形算法实现了算法 2.注意到,若 STSS-IP 模型中目标函数值小于 0,则说明系统不可调度.因此,在 STSS-IP 模型中增加约束 $t-\sum_{\tau_k:D_k\leqslant Q_l}C_k(x_k+1)<0$, 使得模型仅关注小于 0 的最优解.加入约束后的松弛模型若无解,则说明在相应时间域内一定可调度.加入约束是为了进一步提高 gurobi 的求解效率.首先,新的约束能够缩小解空间.其次,加入新的约束有可能和已有约束构成矛盾,从而使得松弛模型无解.gurobi 善于快速处理此类无解情况.

本文调用 linux 系统中的计时器接口 clock_gettime(),分别测量了算法 2 求解一次松弛问题的时间以及 QPA 算法求解一次需求函数 $h(t)$的时间(任务测试集的相关参数设定为:n=30,U=0.9,$P_{\max}/P_{\min}$:=1000,b=1.2×P_i).我们发现,两者耗时均在μs 量级.因此,本文将算法 2 求解一次松弛问题定义为一次迭代.图 6 给出了算法 2 与 QPA 算法的迭代次数随问题参数的变化.其中,图 6(a)给出了测试集的参数设定为 n=30,$P_{\max}/P_{\min}$:=1000,b=1.2×P_i 时,利用率 U 对算法性能的影响.对于算法 2 可判定调度性的算例:当 $U\leqslant 0.99$ 时,算法 2 和 QPA 算法的迭代次数相差不大;当 $U>0.99$ 后,QPA 算法的迭代次数迅速增多,平均情况下 QPA 算法迭代次数多达 1 000 次以上,是本文算法的指数倍.



(a) 利用率 U 的影响

(b) $P_{\max}/P_{\min}$ 最大值的影响

(c) 任务个数 n 的影响

Fig.6 Iteration times of QPA and Algorithm 2 affected by problem parameters

图 6 问题参数对 QPA 和算法 2 迭代次数的影响

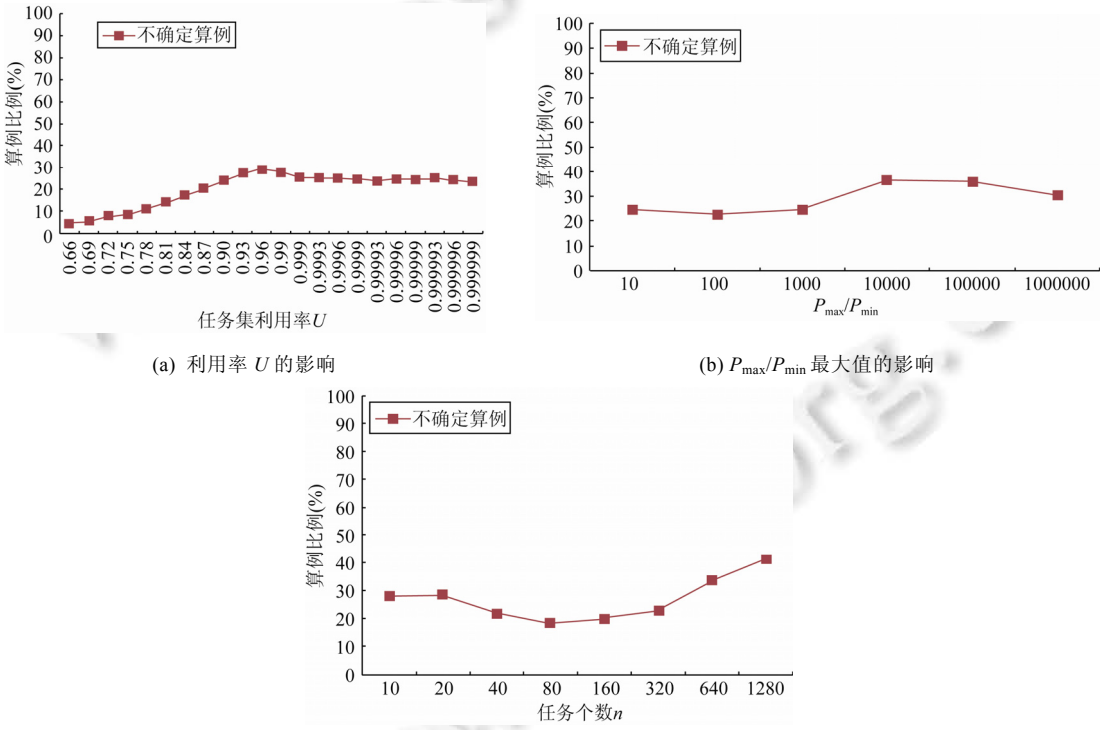
QPA 算法迭代次数随 U 增大的原因是,QPA 算法跳跃式搜索的步长直观上与利用率 U 成反比.在 QPA 算法执行过程中,假设当前检查的时间点为 t,则下一个时间点为 $\min\{t-1,dbf(t)\}$[8].QPA 算法 1 次迭代的步长即为两相邻检查点的距离 $\max\{t-dbf(t),1\}$.在时间域长度一定的条件下,QPA 算法的步长越大,则迭代次数越少,反之亦然.U 可看作任务在时间段内释放作业负载的密度.在时间段 t 固定时,负载密度越大,需求函数 $dbf(t)$就可能越大,进而 $t-dbf(t)$的值越小,步长也越短.这使得 QPA 算法的迭代次数随 U 增大而增多.同时注意到,QPA 算法的迭

代次数并非随 U 增大以线性速度增长.实验结果表明,U=0.99 是 QPA 算法迭代次数的增长速度由线性到指数的分界点,如图 6(a)所示.

另外,在算法 2 中,线性松弛方法的迭代次数与利用率 U 无关,而只与任务数 n 相关.因此,算法 2 的迭代次数并不会随着 U 增大而增大.注意到,算法 2 的迭代次数在 U>0.99 后不但没有增多,反而有所下降,如图 6(a)所示.经过统计得知,在 U>0.99 的算例集合中,不可调度的算例比例接近 80%.算法 2 在判定不可调度的算例时更加高效.算法 2 将整个时间域划分成 n 个子时间域.算法一次迭代就可以判定一个时间子域内是否存在溢出点.给定一个任务系统,在 n 个时间子域中,只要有一个存在溢出点,就可以判定该系统不可调度.因此,对于不可调度的算例,算法 2 往往能够在最初的几个时间子域中迅速找到溢出点,从而返回判定结果,终止程序.

综上,在 U 取值很大时,QPA 算法的搜索步长变得很小,算法退化为在整个时间域上逐个时间点检查溢出情况.然而,U 增大也使得不可调度的算例比例增加,算法 2 在判定不可调度的算例时,往往能够经过最初几次迭代迅速终止.基于以上两方面考虑,在 U 很大时,QPA 算法的迭代次数迅速上升,而算法 2 的迭代次数却有所下降,如图 6(a)所示.

图 6(b)给出在测试集参数设为 n=30,U=0.999,b=1.2×P_i 时,算法性能随 P_{max}/P_{min} 取值的变化.图 6(c)给出了在测试集参数 U=0.999,P_{max}/P_{min}:=1000,b=1.2×P_i 时,任务个数对算法性能的影响趋势.最后,图 7 给出了算法 2 不能确定调度性的算例比例随问题参数的变化.我们发现,对于小规模的算例,能够判定可调度性的算例比例平均不低于 72%,如图 7(a)所示.当任务个数增多或者 P_{max}/P_{min} 值增大时,可判定算例比例略有下降,但仍不低于 60%,如图 7(b)和图 7(c)所示.



(a) 利用率 U 的影响　(b) P_{max}/P_{min} 最大值的影响

(c) 任务个数 n 的影响

Fig.7 Proportion of uncertain instances affected by problem parameters

图 7 问题参数对不确定算例比例的影响

5.4 异步系统中算法性能分析

针对异步系统,本节将算法 1 和 Baruah 等人提出的传统算法[5]进行比较.注意到,文献[5]中的方法回答“可

调度”时,说明异步系统一定是可调度的.但是,当算法返回“不可调度”时,则异步系统的可调度性不能确定.本节比较了本文算法和文献[5]中方法能够确切给出判定结果的算例比例.在问题参数为 n=30,P_{max}/P_{min}:=1000,b=1.2×P_i,U=0.66~0.96 时,图 8 给出了本文算法和传统算法[5]对同一随机算例集合给出的判定结果.实验结果表明,本文算法对近 96%的算例都能给出精确的判定结果,将传统算法[5]不能给出判定结果的算例比例平均降低了29.27%.注意到,当 U≤0.9 时,本文算法不能给出判定结果的算例比例保持在 6%以下.当 U>0.9 之后,该比例有明显上升趋势.这种现象产生的原因是,算法 1 不能给出判定结果的算例比例与 U 成正比关系.由算法 1 第 10 行~第 13 行可知,算法返回“可调度性不确定”结果的必要条件是:存在一个子问题的线性松弛解 LP_l≤0.不妨令 l=n,则

$$\begin{aligned} LP_i &= \min\left(t_2 - t_1 - \sum_{\tau_k:D_k+\varphi_k\le Q_l} C_k(x_k - y_k + 1)\right) \\ &\quad \text{(经过线性松驰,}x_k\text{和}y_k\text{允许取小数)} \\ &= t_2 - t_1 - \sum_{\tau_k:D_k+\varphi_k\le Q_l} C_k((t_2 - t_1 - D_k)/P_k + 1) \\ &\quad \text{(根据约束(4)和(5),将}x_k \le (t_2 - \varphi_k - D_k)/P_k \text{ 和}y_k \ge (t_1 - \varphi_k)/P_k \text{ 代入)} \\ &= (1-U)(t_2 - t_1) - \sum_{\tau_k:D_k+\varphi_k\le Q_l} u_k(P_k - D_k). \end{aligned}$$

观察上式可知,当利用率 U 增大时,等号右边第 1 项$(1-U)(t_2-t_1)$减小,同时,第 2 项 $\sum_{\tau_k:D_k+\varphi_k\le Q_l} u_k(P_k - D_k)$ 增大.LP_l 作为两项的差值,易知,U 越大,LP_l 的值就越小,其小于 0 的概率也就越大.另外,根据算法 1 第 10 行~第 13 行可知,只要存在一个 l 使得 LP_l≤0,算法就有可能返回“可调度性不确定”.因此,LP_l≤0 的概率越大,算例可调度性不确定的可能性就越大.实验结果表明,不确定算例比例的增长速度并不是 U 的线性函数,而是存在一个临界点 U=0.9,如图 8 所示.

针对判定结果不确定的算例,应用算法 3 可将可调度性分析上界 L 降低到一个合理的范围.如图 9 所示,算法 3 得到的 L 较之原上界值 Φ+2H 有指数级的降低,平均降低了近 10^4 倍(为了表述方便,图 9 直接将新的 L 值与超周期 H 进行比较).

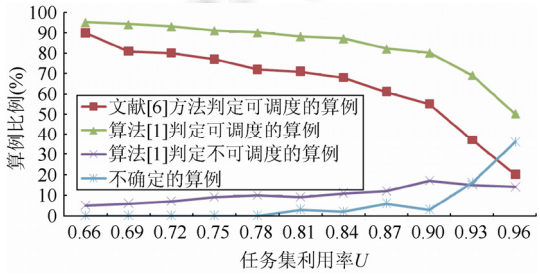


Fig.8 Comparison of performance between Algorithm 1 and method in Ref.[5]

图 8 算法 1 与文献[5]方法的性能比较

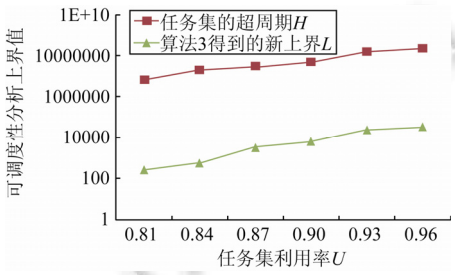


Fig. 9 Comparison between the new bound obtained by Algorithm 3 and the hyper-period H

图 9 算法 3 获得的界与超周期 H 的比较

6 总结与展望

本文研究了偶发性实时任务系统可调度性分析问题的整数规划方法,为同步和异步两类任务系统提出了统一的求解模型和算法框架.第 5 节的实验结果表明,该方法在利用率 U 较大的同步任务系统和较为困难的异步系统中更具优势.而且,本文算法的性能比较稳定,受测试集参数变化的影响较小.注意到,本文算法尽管对某些困难算例能够指数倍地提升求解效率,但并不是对所有算例都能给出明确的判定结果.尤其是对异步系统而言,本文方法对 U≤0.9 的算例更加有效.因此,下一步的工作是更加深入地分析本文 IP 模型的解空间,提出更多的强有效不等式和更多能够从理论上严格证明的极大诱导不等式,进一步缩小算法的 *gap* 值,从而有效降低不确定算例的比例.另外,偶发实时任务模型是实时系统领域中最基本的形式化模型之一,绝大多数复杂实时系统

模型的可调度性分析均基于偶发实时任务模型的基本原理和框架.偶发任务调度新的问题模型和算法框架的提出,极有可能扩展到更复杂的实时系统模型中.因此,充分结合图论和网络优化理论,为 DRT 等更复杂的实时系统模型建立整数规划方法求解框架,也是很有希望的未来工作方向之一.

References:

[1] Baruah S, Pruhs K. Open problems in real-time scheduling. Journal of Scheduling, 2010,13(6):577−582. [doi: 10.1007/s10951-009-0137-5]

[2] Davis RI, Burns A. A survey of hard real-time scheduling for multiprocessor systems. ACM Computing Surveys, 2011,43(4):88−87. [doi: 10.1145/1978802.1978814]

[3] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973,20(1):46−61. [doi: 10.1145/321738.321743]

[4] Hamon G, Rushby J. An operational semantics for Stateflow. Int'l Journal on Software Tools for Technology Transfer, 2004,9(5-6):447−456. [doi: 10.1007/s10009-007-0049-7]

[5] Baruah SK, Rosier LE, Howell RR. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. Real-Time Systems, 1990,2(4):301−324. [doi: 10.1007/BF01995675]

[6] Leung JYT, Merrill ML. A note on preemptive scheduling of periodic, real-time tasks. Information Processing Letters, 1980,11(3):115−118. [doi: 10.1016/0020-0190(80)90123-4]

[7] Eisenbrand F, Rothvoβ T. EDF-Schedulability of synchronous periodic task systems is coNP-hard. In: Proc. of the ACM-SIAM Symp. on Discrete Algorithms. 2010. 1029−1034.

[8] Zhang F, Burns A. Schedulability analysis for real-time systems with EDF scheduling. IEEE Trans. on Computers, 2009,58(9):1250−1258. [doi: 10.1109/TC.2009.58]

[9] Sun JH, Deng QX, Meng YK. Two-Stage workload scheduling problem on GPU architectures: Formulation and approximation algorithm. Ruan Jian Xue Bao/Journal of Software, 2014,25(2):298−313 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4527.htm [doi: 10.13328/j.cnki.jos.004527]

[10] Zhang DS, Wu T, Chen FY, Jin SY. Global EDF-based on-line energy-aware real-time scheduling algorithm in multi-core systems. Ruan Jian Xue Bao/Journal of Software, 2012,23(4):996−1009 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4054.htm [doi: 10.3724/SP.J.1001.2012.04054]

[11] Liu H, Fei SM. A fault-tolerant scheduling algorithm based on EDF for distributed control systems. Ruan Jian Xue Bao/Journal of Software, 2003,14(8):1371−1378 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1371.htm

[12] Pellizzoni R, Lipari G. A new sufficient feasibility test for asynchronous real-time periodic task sets. In: Proc. of the 16th Euromicro Conf. on Real-Time Systems (ECRTS 2004). IEEE, 2004. 204−211. [doi: 10.1109/EMRTS.2004.1311022]

[13] Stigge M, Ekberg P, Guan N, Yi W. The digraph real-time task model. In: Proc. of the 17th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS). IEEE, 2011. 71−80. [doi: 10.1109/RTAS.2011.15]

[14] Stigge M, Ekberg P, Guan N, Yi W. On the tractability of digraph-based task models. In: Proc. of the 23rd Euromicro Conf. on Real-Time Systems (ECRTS 2004). IEEE, 2011. 162−171. [doi: 10.1109/ECRTS.2011.23]

[15] Fersman E, Krcal P, Pettersson P, Yi W. Task automata: Schedulability, decidability and undecidability. Information and Computation, 2007,205(8):1149−1172. [doi: 10.1016/j.ic.2007.01.009]

[16] Tan GZ, Sun JH, Wang BC, Yao WH. Solving Chinese postman problem on time varying network with timed automata. Ruan Jian Xue Bao/Journal of Software, 2011,22(6):1267−1280 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4033.htm [doi: 10.3724/SP.J.1001.2011.04033]

[17] Ripoll I, Crespo A, Mok AK. Improvement in feasibility testing for real-time tasks. Real-Time Systems, 1996,11(1):19−39. [doi: 10.1007/BF00365519]

[18] Hoang H, Buttazzo G, Jonsson M, Karlsson S. Computing the minimum EDF feasible deadline in periodic systems. In: Proc. of the 12th IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications. IEEE, 2006. 125−134. [doi: 10.1109/RTCSA.2006.22]

[19] Spuri M. Analysis of deadline scheduled real-time systems. Technical Report, 2772, INRIA, 1996.

[20] Zhang F, Burns A. Schedulability analysis of EDF-scheduled embedded real-time systems with resource sharing. ACM Trans. on Embedded Computing Systems, 2013,12(3):1−19. [doi: 10.1145/2442116.2442117]

[21] Optimization G. Gurobi optimizer reference manual. 2012. http://www.gurobi.com

[22] Bini E, Buttazzo GC. Measuring the performance of schedulability tests. Real-Time Systems, 2005,30(1-2):129−154. [doi: 10.1007/s11241-005-0507-9]

[23] Davis RI, Zabos A, Burns A. Efficient exact schedulability tests for fixed priority real-time systems. IEEE Trans. on Computers, 2008,57(9):1261−1276. [doi: 10.1109/TC.2008.66]

[24] Tan G, Sun J, Hou G. The time-dependent rural postman problem: Polyhedral results. Optimization Methods and Software, 2013, 28(4):855−870. [doi: 10.1080/10556788.2012.666240]

[25] Sun JH, Meng YK, Tan GZ. An integer programming approach for the Chinese postman problem with time-dependent travel time. Journal of Combinatorial Optimization, 2014,29(3):565−588. [doi: 10.1007/s10878-014-9755-8]

附中文参考文献:

[9] 孙景昊,邓庆绪,孟亚坤.GPU 上两阶段负载调度问题的建模与近似算法.软件学报,2014,25(2):298−313. http://www.jos.org.cn/1000-9825/4527.htm [doi: 10.13328/j.cnki.jos.004527]

[10] 张冬松,吴彤,陈芳园,金士尧.多核系统中基于 Global EDF 的在线节能实时调度算法.软件学报,2012,23(4):996−1009. http://www.jos.org.cn/1000-9825/4054.htm [doi: 10.3724/SP.J.1001.2012.04054]

[11] 刘怀,费树岷.基于 EDF 的分布式控制系统容错调度算法.软件学报,2003,14(8):1371−1378. http://www.jos.org.cn/1000-9825/14/1371.htm

[16] 谭国真,孙景昊,王宝财,姚卫红.时变网络中国邮路问题的时间自动机模型.软件学报,2011,22(6):1267−1280. http://www.jos.org.cn/1000-9825/4033.htm [doi: 10.3724/SP.J.1001.2011.04033]

孙景昊(1985－),男,河北沧州人,博士,副教授,主要研究领域为实时系统调度算法,最优化理论,时间自动机.

孙景昶(1993－),男,学士,主要研究领域为数学规划理论和算法.

关楠(1981－),男,博士,教授,CCF 专业会员,主要研究领域为实时系统,嵌入式系统.

邓庆绪(1970－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为实时嵌入式系统,可重构计算,物联网.