管道流量泄漏在线监测中的模糊调度设计

周 鹏
ZHOU Peng

塔里木大学 信息工程学院,新疆 阿拉尔 843300
College of Information Engineering,Tarim University,Alar,Xinjiang 843300,China
E-mail:zpzqxy@163.com

ZHOU Peng.Fuzzy scheduling design in computer on-line monitoring system based on pipeline flux leak.Computer Engineering and Applications,2009,45(27):231-236.

Abstract: Flow against pipeline leakage and the pipe network sudden burst pipe to pipeline leakage flow for the application objects,fuzzy scheduling design flow used in pipeline leak monitoring.In order to improve the control performance of systems and use the limited computing resource reasonably,it is necessary to consider the collaborative design problem between control and real-time scheduling in computer controlled systems synthetically.In this paper,the real-time scheduling problem of multiple control tasks with fuzzy deadlines is studied.The concept of dedication index and the scheduling policy of Largest Dedication First (LDF) are proposed.In order to relieve the overhead caused by the thrashing among tasks,the threshold-based Largest Dedication First (TLDF) is presented.Finally,two proposed scheduling policies LDF and TLDF are compared by simulation.The scheduling of control tasks with fuzzy deadlines is implemented,and the utilization of computing resource gets increased mean while the control performance cost gets decreased and trade off.

Key words: computer controlled systems;control tasks;real-time scheduling;fuzzy;collaborative design

摘 要:针对管道流量泄漏和管网突发性的爆管,以管道流量泄漏为应用对象,将模糊调度设计应用于管道流量泄漏监测中,研究具有模糊截止期的多控制任务的实时调度问题,提出奉献度概念和最大奉献优先(LDF)的调度策略。为了减小因任务间频繁切换造成的系统开销,提出基于抢占阈值的最大奉献优先(TLDF)调度策略。最后通过仿真比较 LDF 和 TLDF 两种调度策略,实现具有模糊截止期的控制任务调度,能够减少并均衡控制性能的损失,同时提高系统计算资源的使用率。

关键词:计算机控制系统;控制任务;实时调度;模糊;协同设计
DOI:10.3778/j.issn.1002-8331.2009.27.070 **文章编号**:1002-8331(2009)27-0231-06 **文献标识码**:A **中图分类号**:TP273

1 引言

管道流量运输是一种新兴经济的运输方式,尤其在输送气体、液体、浆体等散装物品方面具有天然优势。目前全世界大型输油管总长超过 200 万公里,并且以每年约 5 万公里的速度递增,经过一百多年的发展,管道运输业已成为与铁路、公路、航空、水运并行的五大运输手段之一。随着管线的增多、管龄的增长、施工缺陷、腐蚀及人为破坏的存在,管道事故频频发生,给人们的生命财产和生存环境造成巨大的威胁。目前我国已经到了泄漏事故多发时期,对管道泄漏实时监测中的模糊调度设计的需求非常迫切,对此该文研究具有模糊截止期的多控制任务的实时调度问题,提出奉献度的概念和最大奉献优先(LDF)的调度策略,实现具有模糊截止期的控制任务调度。

在实时调度领域人们仅仅关注实时调度技术的学术研究,却很少把实时调度技术与控制技术结合在一起综合考虑。计算机控制系统传统上是由控制系统工程师和实时系统工程师各自独立设计的,控制系统工程师负责开发对象的被控模型,设计控制规律,进行仿真和测试,在进行控制设计时不考虑计算机是如何调度计算的;而实时系统工程师则负责控制算法的调度实现,通过给控制任务分配优先级和截止期来配置实时系统,在进行调度设计时不考虑具体的控制算法是如何实现的[1]。在计算资源有限的计算机控制系统中,则存在资源共享问题。一方面控制任务可以和其他任务一起并行执行,或多个控制任务在同一处理器上运行,此时需要考虑控制性能与计算资源利用之间的均衡问题,而控制任务之间的调度将是这种均衡的关键。另一方面实时系统工程师往往是根据控制系统工程师提供有关确定的采样周期、最坏情况执行时间和固定截止期进行调度设计,由于控制系统工程师提供的时间需求往往留有很大余量并都是非随机、非模糊、确定的数据,因此这种调度设计往往偏于保守的,最终可能导致计算资源的利用率偏低,并影响到整个控制系统的性能,所以需要综合考虑控制与调度的协同设

基金项目:新疆生产建设兵团工业科技攻关计划(Xinjiang Production and Construction Corps Industrial Technology Research Plans No.2007GG15);塔里木大学校长基金青年资助项目(Tarim University Principal Youth Fund Grant No.TDZKQN05002)。
作者简介:周鹏(1970-),男,副教授,主要研究领域:计算机应用技术。CCF 会员,会员证:E200011153M。
收稿日期:2008-05-26 **修回日期**:2008-08-04

计问题。

在计算机控制系统中,有些任务特征通常是模糊性的或不确定性的,如控制器任务的截止期等,它可以是在采样周期的某个范围内随机波动。由于计算硬件、控制算法的执行时间性能、实时操作系统、调度算法和网络延时等都有可能引起控制系统中各种大小的延迟和抖动。在实时调度理论中,已有一些针对具有不确定任务特征的调度问题的研究成果,如将任务截止期定义为模糊数,用概率模型来描述模糊约束,用来确定哪个任务执行,哪个任务在系统过载时被抑制,但这种解决方案是基于经验规则的,不能保证完整性和正确性;考虑模糊空闲时间和模糊关键性等特征,构造一个多准则优化问题,当系统过载时,选择重要任务进行调度;用梯形隶属函数来描述任务的处理时间和截止期,把满意度指标作为代价函数描述任务的完成时间与可调度性的满意度之间的关系;用 Sigmoid 函数来描述截止期约束规则和截止期调度规则,考虑用多个约束规则的满意指标来构造任务和约束的相关矩阵,但这种描述对于动态变化的任务特征是不适合的。如何从有限的计算资源得到最好的控制性能是目前需要解决的问题,该文主要考虑基于时间触发采样的计算机控制系统,研究具有模糊截止期的多控制任务的单处理器调度设计问题。

2 控制任务模型

在计算机控制系统中每经过采样周期 P 秒,控制器将采样系统的输出和参考输入进行比较,重新计算新的控制量,然后将新的控制量作为被控系统的输入。控制的目的是使得系统输出尽可能地接近参考输入,控制量的状态更新与系统输出的计算构成整个系统的闭环回路,在该回路中控制器状态的更新也称为控制器,控制器是由控制算法程序实现的,通常看成是周期的、硬实时任务的例子。

2.1 特征参数

考虑由多个控制器任务组成的任务集,记 T_i 为第 i 个控制器任务,T_{ij} 为 T_i 的第 j 个任务实例,即第 j 次采样控制循环中的控制器任务。下面列出该文讨论中将涉及到的有关 T_i 的一些基本任务时态参数,每个控制器任务实例 T_{ij} 都有相同的属性。

r_i:释放时间或就绪时间,定义为 T_i 可被调度和执行的开始时刻,该文中把任务的到达时间和释放时间看成是相同的,即假设任务到达以后就可以执行。

P_i:采样周期,定义为在理想情况下的控制器采样周期。

$WCET_i$:最坏情况执行时间。

C_i:执行时间,定义为在理想情况下 T_i 完成执行所需要的时间。在仿真中 C_i 就是指 $WCET_i$,并假定 C_i 是已知的。

d_i:截止期,定义为 T_i 必须完成的最终时刻。

s_i:空闲时间,定义为 T_i 在离 d_i 前完成可以等待的时间,即经 s_i 时间等待以后,T_i 仍能赶在 d_i 时刻完成,初始时刻的空闲时间为 $s_i=d_i-r_i-C_i$。

p_i:优先级,定义为任务被调度的优先指标,取决于采用的调度策略。假设优先级值越大,优先级等级就越高,任务也就越早被调度。

h_i:抢占阈值,定义为 T_i 被抢占的优先级界,即当 $p_j>h_i$ 时,T_i 才可被 T_j 抢占。一般情况下,$h_i\in[p_i,\infty]$。假定截止期 d_i 是模糊不确定的,从而空闲时间也是模糊不确定的,其他属性参数假设是已知的或可计算得到的。

2.2 模糊截止期

关于截止期的模糊表达形式,可以采用离散型或连续型模糊表示;用于描述模糊截止期可供选择的连续型隶属函数形式也有多种,比如梯形模糊数或三角形模糊数等,也可以是截尾正态形模糊数等,这里考虑连续梯形模糊数的表达形式[2]。

记 $d_i\equiv Trapezoid(a_i,a_i',b_i',b_i)$,其中$(a_i,a_i',b_i',b_i)$为梯形模糊数的 4 个顶点,且 $r_i<a_i\leqslant a_i'\leqslant b_i'\leqslant b_i\leqslant p'$。记 $\mu_i(t)$ 为 d_i 的 $[a_i,b_i]$截尾隶属函数,即满足

$$\mu_i(t)=\begin{cases}\dfrac{(t-a_i)h_i}{a_i'-a_i}, & a_i<t\leqslant a_i'\\ h_i, & a_i'<t\leqslant b_i'\\ \dfrac{(b_i-t)h_i}{b_i-b_i'}, & b_i'<t\leqslant b_i\\ 0, & \text{其他}\end{cases}\tag{1}$$

且称 d_i 为 T_i 的$[a_i,b_i]$截尾模糊截止期,其中 a_i 和 b_i 在该文分别被称为 T_i 的最早截止期和最晚截止期。当 $h_i=2/[(b_i-a_i)+(b_i'-a_i')]$时,$\int\mu_i(t)\mathrm{d}t=1$。当 $a_i'=b_i'$时变为三角形模糊数;当 $a_i=a_i'$、$b_i'=b_i$ 时则变为平均分布形模糊数。记 d_{ij} 为实例 T_{ij} 的截止期,则有 $d_{ij}=(j-1)P_i+d_i$,于是 d_{ij} 的梯形模糊数可表示为 $d_{ij}\equiv Trapezoid(a_{ij},a_{ij}',b_{ij},b_{ij})$,其中,$a_{ij}\equiv(j-1)P_i+a_i$,$a_{ij}'\equiv(j-1)P_i+a_i'$,$b_{ij}'\equiv(j-1)P_i+b_i'$和 $b_{ij}\equiv(j-1)P_i+b_i$。同样 d_{ij} 是 T_{ij} 的$[a_{ij},b_{ij}]$截尾模糊截止期,其中 a_{ij} 和 b_{ij} 分别是 T_{ij} 的最早截止期和最晚截止期。另外记 $\mu_{ij}(t)$为 d_{ij} 的$[a_{ij},b_{ij}]$截尾隶属函数,则有 $\mu_{ij}(t)=\mu_i(t-(j-1)P_i)$。

2.3 模糊空闲时间

记 s_{ij} 为 T_{ij} 的空闲时间,$C_{ij}(t)$为 T_{ij} 在当前时刻 t 的剩余执行时间值,则由空闲时间的定义,T_{ij} 在 t 时刻的空闲时间定义为 $s_{ij}(t)=d_{ij}-t-C_{ij}(t)$,其中 $t\geqslant r_{ij}$,$r_{ij}=(j-1)P_i+r_i$ 为 T_{ij} 的释放时间,$C_{ij}(r_{ij})=C_i$;T_{ij} 的初始空闲时间为 $s_{ij}=d_{ij}-r_{ij}-C_i$。由于 d_{ij} 是$[a_{ij},b_{ij}]$截尾模糊数,因此 $s_{ij}(t)$则是$[s_{ij}^{\min}(t),s_{ij}^{\max}(t)]$截尾模糊数,其中 $s_{ij}^{\min}(t)\equiv a_{ij}-t-C_{ij}(t)$是 T_{ij} 在 t 时刻允许的最小空闲时间,而 $s_{ij}^{\max}(t)\equiv b_{ij}-t-C_{ij}(t)$是 T_{ij} 在 t 时刻允许的最大空闲时间。可见空闲时间变化范围的区间长度始终为定常值 $b_{ij}-a_{ij}=(b_i-a_i)$。

(1)当 $t-C_{ij}(t)<a_{ij}$ 时,$s_{ij}^{\min}(t)>0$ 表示 T_{ij} 仍然有富裕的空闲时间在最早截止期前执行完成;

(2)当 $t-C_{ij}(t)=a_{ij}$ 时,$s_{ij}^{\min}(t)=0$ 表示 T_{ij} 刚好可以在最早截止期时刻执行完成;

(3)当 $a_{ij}<t-C_{ij}(t)<b_{ij}$ 时,$s_{ij}^{\min}(t)<0<s_{ij}^{\max}(t)$表示 T_{ij} 已经没有足够的空闲时间让其在最早截止期前执行完成,但仍然有富裕的空闲时间让其在最晚截止期前执行完成;

(4)当 $t-C_{ij}(t)=b_{ij}$ 时,$s_{ij}^{\max}(t)=0$ 表示 T_{ij} 刚好可以在最晚截止期时刻执行完成;

(5)当 $t-C_{ij}(t)>b_{ij}$ 时,$s_{ij}^{\max}(t)<0$ 表示 T_{ij} 已经没有足够的空闲时间让其在最晚截止期前执行完成。

3 控制任务的调度设计

考虑由 n 个伺服系统 $G_i(s)$构成的多控制任务系统,每个受控对象由各自控制器所控制,由 n 个控制器任务$(T_i,i=1,2,\cdots,n)$竞争计算资源参预调度。记 Q_0 是由所有控制器任务实例混合组成任务队列,按优先级值从大到小顺序排队;Q_i 是由第 i 个控制器任务实例组成的子任务队列,按 FIFO(First-In First-Out)顺序排队($i=1,2,\cdots,n$)。

3.1　调度步骤

(1)对到达的控制器任务,调度器根据调度策略配置优先级,将任务插入 Q_0 队列中,最前面的任务 T_i 优先级最高,优先得到 CPU 调度执行。

(2)如果考虑完全抢占式调度的话,在 Q_0 中等待执行的任务可与正占用 CPU 执行的任务 T_i 进行竞争。若有某个等待任务 T_j 的优先级更高,即 $p_j>p_i$,则 T_j 可抢占 CPU,而 T_i 则退出并插回到 Q_0 队列中。

(3)由于不知道任务的实际截止期,调度器将根据任务的最晚截止期来确定任务是否错失截止期。当 T_i 还未错失其最晚截止期时,它可以继续占用计算资源或继续排在 Q_0 队列中等待 CPU 调度;当 T_i 或 Q_0 中其他任务错失他们的最晚截止期时,调度器将及时中止这些任务的执行或等待。

(4)调度器将完成任务实例按不同的控制任务分别送入各自不同的任务队列中,控制器 i 的任务实例送入 Q_i 队列。执行完成的任务实例将产生新的控制量,而错失最晚截止期的任务实例将不更新控制量。

3.2　控制环的计算

表 1 给出了计算机控制系统中控制环计算的伪码形式,其中不考虑调度设计时的控制环计算是指在理想情况下的单个控制环的计算过程,而基于调度设计时的控制环计算是指在多控制任务竞争计算资源情况下的每个控制环的计算过程。

在基于调度设计的控制环计算方案中:假设控制任务的采样周期维持理想情况下的设计不变;在每次采样循环内,增加判断当前控制任务实例是否能在其最晚截止期前完成或是错失最晚截止期而放弃。若当前任务实例完成执行,说明在当前采样环内控制算法在最晚截止期前完成计算,并得到新的控制量,从而对控制量进行更新;若当前任务实例错失最晚截止期,说明在当前采样环内控制算法不能在最晚截止期前完成计算,将维持上一采样环内控制量不变。

表 1　控制环的计算伪码

不考虑调度设计	基于调度设计
t=Current Time	t=Current Time
LOOP	LOOP
A/D-Conversion	A/D-Conversion
UpdateStateAlgorithm	IF task T finishes
D/A-Conversion	UpdateStateAlgorithm
CalculateSystemOutput	END
$t=t+P$	D/A-Conversion
WaitUntil(t)	CalculateSystemOutput
END	$t=t+P$
	WaitUntil(t)
	END

4　最大奉献优先调度策略

4.1　奉献度

对于一个任务来讲,越早调度就越早完成,从单个任务来讲,希望任务尽早完成,完成得越早就越满意。但对于一个任务集来讲,如果每个任务都争着要求尽早调度的话,将有可能会影响到整个任务集的调度成功率。为此,希望每个任务都能够发扬奉献精神,在各自许可的范围内尽量推迟调度,以便让其他更加紧急的任务优先调度,这种奉献的精神是基于该任务本身在最坏情况下能够确保调度完成的前提下才加以考虑的[3]。事实上对于计算机控制系统来说,控制任务往往允许具有一定的"软截止期"性质,允许在其截止期的一定范围内完成,从这个角度,奉献精神既能让更多的任务得到调度,又能保证任务在一定精度范围内实现调度,因此,这种奉献的概念可应用于计算机控制系统的任务调度设计中。

对于模糊截止期任务来说,截止期不是指某个特定的值,其变化范围可以是从最早截止期到最晚截止期所确定的时间区间。下面给出奉献度指标 DIN(Dedication Index)的定义。记 F_i 为任务 T_i 的完成时刻,$\forall x\in R^+$,T_i 的奉献度函数可以定义为[4]:

$$Ded_i(x)\equiv Ded_i\{F_i\leqslant x\}\equiv\frac{\int_{-\infty}^{x}\mu_i(F)\mathrm{d}F}{\int_{-\infty}^{+\infty}\mu_i(F)\mathrm{d}F} \tag{2}$$

当取式(1)形式的隶属函数时

$$Ded_i(x)=\begin{cases}0, x\leqslant a_i\\ 0.5\dfrac{(x-a_i)^2h_i}{a_i'-a_i}, a_i<x\leqslant a_i'\\ 0.5(a_i'-a_i)h_i+(x-a_i')h_i, a_i'<x\leqslant b_i'\\ 1-0.5\dfrac{(b_i-x)^2h_i}{(b_i-b_i')h_i}, b_i'<x\leqslant b_i\\ 1, x\geqslant b_i\end{cases} \tag{3}$$

记 $C_i(t)$ 为 T_i 在 t 时刻的剩余执行时间,在理想情况下,T_i 将在 $t+C_i(t)$ 执行完成,利用 $t+C_i(t)$ 取代式(2)中的 x,计算 T_i 在当前时刻的奉献度指标:

$$DIN_i(t)=Ded_i(t+C_i(t)) \tag{4}$$

由此可见,奉献度指标不仅与模糊截止期的形式有关,而且也与任务的剩余执行时间有关。对于$[a_i,b_i]$截尾模糊截止期,奉献度函数的定义表示:

(1)当 $x\leqslant a_i$ 时,$Ded_i(x)=0$,即任务 T_i 在最早截止期 a_i 之前完成的奉献度为 0;

(2)当 $x\geqslant b_i$ 时,$Ded_i(x)=1$,即任务 T_i 在最晚截止期 b_i 之后完成的奉献度为 1;

(3)当 $a_i<x<b_i$ 时,即在(a_i,b_i)之间完成的奉献度取决于截止期的隶属函数 μ_i,对于连续型模糊数,积分(2)存在,且奉献度是 x 的连续递增函数;

(4)最大奉献度为 1,最小奉献度为 0,最早和最晚截止期对应的奉献值分别是 0 和 1;

(5)同样地可对任务实例 T_{ij} 的奉献度函数进行类似定义和计算。

4.2　最大奉献优先

4.2.1　优先级配置

最大奉献原则是奉献值较大的任务配置较高的优先级等级。对于任意时刻 t_1 和 $t_2(t_2>t_1)$,则有 $C_i(t_1)-C_i(t_2)\leqslant t_2-t_1$,或有 $t_2+c_i(t_2)\geqslant t_1+C_i(t_1)$。由于 $Ded_i(x)$ 是递增函数,因此,$Ded_i(t_2+C_i(t_2))\geqslant Ded_i(t_1+C_i(t_1))$,或 $DIN_i(t_2)\geqslant DIN_i(t_1)$,即奉献度指标 $DIN_i(t)$ 也是递增函数,这样奉献度指标可作为任务的优先级取值,即 $p_i(t)=DIN_i(t)$,此意味着奉献度指标值越大,任务的优先级就越高,从而优先调度具有高奉献度的任务,满足最大奉献优先调度策略的基本原则[5]。

4.2.2　LDF 调度策略

根据最大奉献原则来配置任务的优先级,由此确定的调度策略称为最大奉献优先调度策略,记为 LDF(Largest Dedica-

tion First)。

(1)采用奉献度驱动的调度策略,高奉献度任务可以抢占低奉献度任务。

(2)如果两个任务 T_i 和 T_j 具有相同的奉献度时,则比较它们的最晚截止期 b_i 和 b_j,具有较小最晚截止期的任务优先调度执行。

(3)当任务还未错失最晚截止期时,它还有继续奉献的空间,因此可以继续等待下一时刻的抢占调度;当任务错失最晚截止期时,它已经没有继续奉献的空间,这时,如果该任务还未完成,则将中止或放弃该任务的执行。

根据奉献度函数的定义(1)当 $t+C_i(t)<a_i$ 时,T_i 仍然有富裕的空闲时间在最早截止期前执行完成,其奉献度为零;(2)当 $a_i<t+C_i(t)<b_i$ 时,T_i 已经错失最早截止期,但仍然有富裕的空闲时间让其在最晚截止期前执行完成,这是 T_i 能够做出真正奉献的时间段;(3)当 $t+C_i(t)>b_i$ 时,表示 T_i 已经错失最晚截止期,其奉献度为 1,这时 T_i 将夭折。利用奉献度指标作为优先级的取值是合理的,当任务到达后,任务的奉献度起初为零,并一直延续到任务的最小空闲时间为零。随着任务的最小空闲时间小于零,其奉献度也越来越大,需要调度的紧迫程度也越来越高。由于任务的奉献度在最早截止期到最晚截止期之间是递增的,因此任务的优先级配置是动态变化的,其开销比较大,这是 LDF 的不足之处。

4.3 基于抢占阈值的 LDF

4.3.1 颠簸现象

当 T_i 持续执行而不被其他任务抢占时,其奉献度保持不变,而所有等待任务的奉献度是递增的。一旦在某一时刻 t_1,某等待任务(如 T_j)的奉献度大于执行任务(T_i)的奉献度,即 $DIN_j(t_1)>DIN_i(t_1)$,这时 T_j 将抢占 T_i。当 T_j 执行时,其奉献度保持不变,但 T_i 的奉献度将因等待而严格递增,一直到另一时刻 t_2(大于 t_1),T_i 的奉献度又反超 T_j,这时 T_i 反过来再抢占 T_j。如此反复,任务间将不断被抢占,造成任务间的频繁切换,或颠簸现象严重,颠簸现象既增大系统内存的开销,又造成 CPU 带宽的浪费,这种比较严重的颠簸现象可能会限制 LDF 算法的应用,为此在 LDF 策略基础上提出设定抢占的约束,即抢占阈值。

4.3.2 抢占阈值

本小节采用抢占阈值的概念来扩展 LDF 的抢占调度,这种基于抢占阈值的或有条件抢占的 LDF 调度策略记为 TLDF(Threshold-based Largest Dedication First)。下面给出 TLDF 抢占阈值(h_i)的一种与任务的完成率有关的配置方法,任务的优先级(p_i)仍按 4.2.1 小节介绍的方法进行配置,由于 $p_i\in[0,1]$,因此可设定 $h_i\in[p_i,1]$。任务的完成率定义为任务已完成的时间与任务总执行时间之比,对于给定的任务完成率参考值 α,当任务完成率不到 α 时,抢占阈值取为 $h_i=p_i$,即完全抢占方式;否则抢占阈值取为 $h_i=1$,即非抢占方式。LDF 就是 TLDF 在 $h_i=p_i$ 时的完全抢占下的调度策略,当 $h_i=1$ 时的 TLDF 为非抢占调度策略,而当 $h_i\neq1$ 时的 TLDF 称为基于抢占阈值的调度策略。

4.3.3 抢占模式

考虑抢占阈值时,在 3.1 小节中的调度步骤(2)需要修改如下:等待任务 T_j 能够抢占当前执行任务 T_i 的充要条件是 $p_j>h_i$,虽然 T_i 名义上或实际上的优先级值仍然是 p_i,但其优先级等级已提高到抢占阈值 h_i 的层次,从而暂缓 T_j 抢占 T_i 的进度。当 $h_i=p_i$ 时,抢占过程变为完全抢占,而当 $h_i=1$ 时,变为非抢占调度,这时 Q_0 队列中的任务只有等到 T_i 执行完成以后才有机会竞争CPU。

5 仿真

5.1 条件

考虑由压力传感器位置控制系统组成的计算机控制系统,每个伺服系统由传递函数 $G(s)=1\ 000/(s(s+1))$来描述,并由 PD(Proportional Differentiation)控制器所控制,其控制算法离散形式为 $Prop(t)=K(r(t)-y(t))$,$Der(t)=\xi\cdot Der(t-P)+\eta\cdot(y(t-P)-y(t))$,$u(t)=Prop(t)+Der(t)$,其中 $r(t)$为参考输入,$y(t)$为系统输出,$u(t)$为控制量,$\xi=M/(NP+M)$,$\eta=NKM/(NP+M)$,K 和 M 为控制参数,N 为常值,P 为采样周期。表 2 给出两组这两个控制器的控制参数(K,M,N),采样周期 P,执行时间 C 以及梯形模糊截止期 d 的 4 个顶点。这两个控制器作为任务竞争计算资源,仿真时间取为 T_s=2 000。系统输入都取为阶跃函数:当 $t\leqslant500$ 或 $1\ 000<t\leqslant1\ 500$ 时,$r(t)=1$;在其他时间,$r(t)=-1$,单位为ms。

表 2 控制器参数和其它任务特征参数

		K	M	N	P/ms	C/ms	d/ms
第一组	T_1	1.0	0.042	100	12	6	(7,8,9,10)
	T_2	1.1	0.035	50	10	5	(6,7,7,8)
第二组	T_1	1.0	0.040	100	14	7	(7,8,9,11)
	T_2	1.2	0.030	80	12	6	(6,7,8,9)

5.2 控制性能损失指标

多任务控制系统的综合控制性能可选择下面的指标来衡量:

$$cpc=\sum_{i=1}^{n}w_i cpc_i,\ cpc_i=\int_0^{T_s}[Yideal,i(t)-Yactual,i(t)]^2\,dt$$

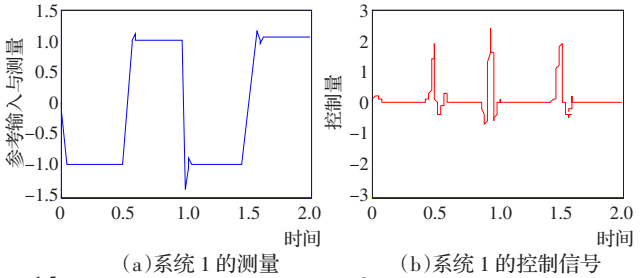
cpc_i 和 w_i 分别为第 i 个控制子系统的控制性能损失及其加权系数,后者可根据控制器任务的重要性或价值来设计;$Yideal,i$ 为表 1 中不考虑调度设计时第 i 个控制子系统的理想输出,$Yactual,i$ 为表 1 中基于调度设计时第 i 个控制子系统的实际输出。这里 CPC(Control Performance Cost)也可看成是由于系统过载时调度额外的总损失。

5.3 控制性能比较

考虑表 2 的第 1 组参数,T_1 共有 166 个任务实例,T_2 共有 200 个任务实例。由于任务的截止期是个模糊数,因此不能用传统的公式来计算 CPU 利用率。可用最晚截止期 b_i 来代替模糊截止期 d_i,得到最乐观情况下的利用率下界:$U\geqslant(C_1/b_1+(C_2/b_2)=49/40>1$,由此可见,在最乐观情况下系统仍然过载,没有任何调度算法能够保证所有任务实例满足最晚截止期。图 1 和图 2 分别是在 LDF 调度和在 TLDF 调度下得到的控制性能比较图,这两个图中所有子图的 X 轴都是刻画时间变化。图(a)和图(c)分别比较两个子系统的测量,虚线为参考输入信号,实线为实际测量,点划线为理想测量;图(b)和图(d)分别比较两个子系统的控制信号,实线为实际控制量,虚线为理想控制量。理想控制量与理想测量是在理想条件下控制系统工程师期待的理想结果;而实际控制量与实际测量是在实时系统工程师根据控制系统工程师提供的时间约束进行实时设计下得到的实际结果。

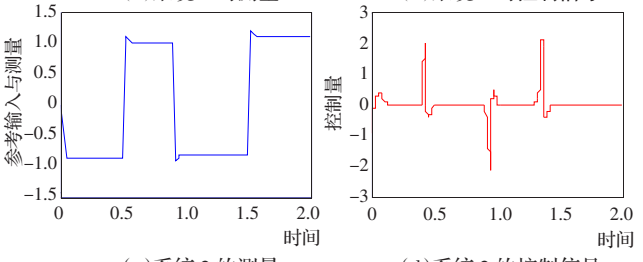
记 EUR(Efficient Utilization Rate)为 CPU 的有效利用率,定义为满足最晚截止期的实例所占用的 CPU 时间/总的仿真时间×100%。CSN(Context Switching Number)为任务未完成但被抢占的次数。在 LDF 调度下系统 2 得到理想的设计结果,其控

制性能损失为零,而系统 1 的控制性能损失达到 CPC_1=55.271 5。此外,EUR 达到 79.70%,而 CSN 达到 333 次。在 TLDF 调度下(任务完成率的参考值取为 α=50%),系统 1 的控制性能损失大大减小,为 CPC_1=3.823 3,而系统 2 的控制性能损失稍稍有所上升,为 CPC_2=6.700 2,但系统的总体控制性能损失得到大大降低。此外 EUR 也提高到 81.35%,而 CSN 下降到 34 次。



(a)系统 1 的测量

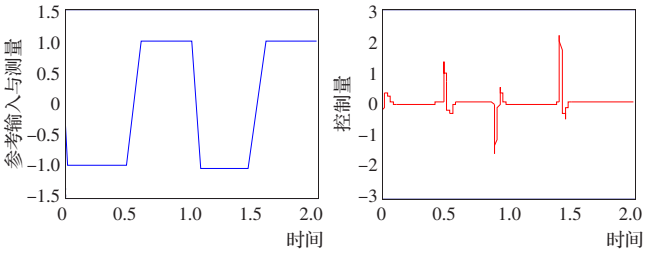
(b)系统 1 的控制信号



(c)系统 2 的测量

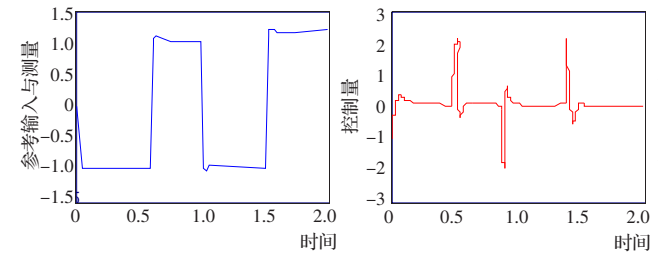
(d)系统 2 的控制信号

图 1　在 LDF 调度下的控制性能比较



(a)系统 1 的测量

(b)系统 1 的控制信号



(c)系统 2 的测量

(d)系统 2 的控制信号

图 2　在 TLDF 调度下的控制性能比较

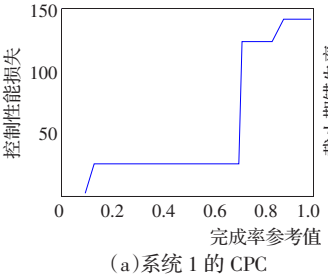
5.4　不同完成率下的 TLDF 性能

记 MDR(Missed Deadline Ratio)为截止期错失率,定义为错失最晚截止期的任务个数与总任务实例个数之比。考虑表 2 中的第二组参数。在最乐观情况下的利用率为 $U\geqslant(C_1/b_1)+(C_2/b2)$=43/33>1,表明系统仍然是过载的,同样没有调度算法可以保证所有实例都满足其最晚截止期。图 3 给出两个子系统的CPC 和 MDR 比较,图 4 给出 EUR、CSN、总 CPC 和总 MDR 的仿真结果,所有子图的 X 轴都是刻画任务完成率参考值(α)的变化。图 4 的比较仿真结果表明:

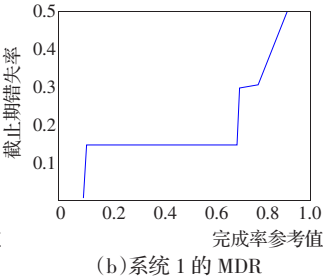
(1)对于较小的(或较大的)α,总 CPC 是相对较大的,然而子系统的控制性能损失变化不同,系统 1 的控制性能损失是随 α 单调递增的,而系统 2 的控制性能损失是单调递减的。

(2)对于完全抢占或较大的 α,即 α>0.9,系统 2 没有控制性能损失,T_2 的所有实例都得到完成,但系统 1 的 CPC 和 MDR 都达到最大。

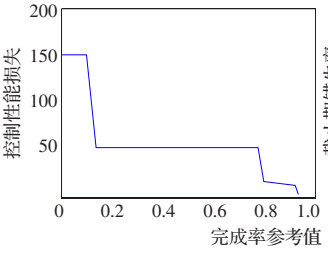
(3)对于非抢占或较小的 α,即 $\alpha\leqslant$0.1,系统 1 没有控制性能损失,T_1 的所有实例都得到完成,但系统 2 的 CPC 和 MDR 都达到最大。



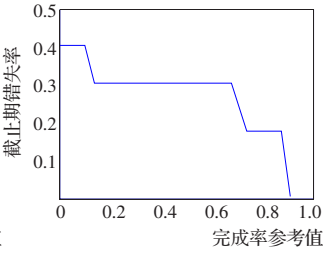
(a)系统 1 的 CPC



(b)系统 1 的 MDR

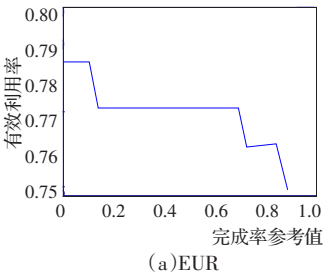


(c)系统 2 的 CPC

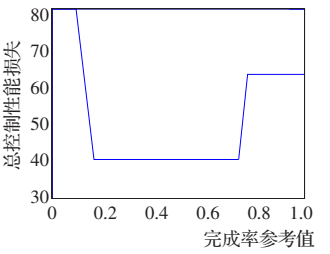


(d)系统 2 的 MDR

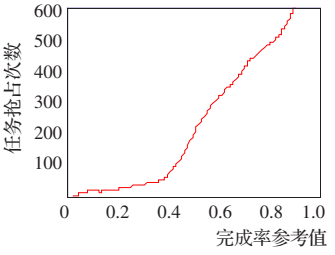
图 3　两个子系统的 CPC 和 MDR 比较



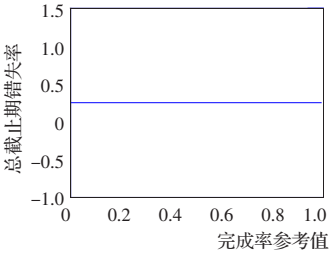
(a)EUR



(b)CPC



(c)CSN



(d)MDR

图 4　EUR、CSN、总 CPC 和 MDR 比较

(4)EUR 是随 α 单调递减的,对 $\alpha\leqslant$0.1 达到最大值(0.785 0),对 $\alpha\geqslant$0.9 达到最小值(0.750 0)。

(5)CSN 是随 α 单调递增的,对 $\alpha\leqslant$0.1 时为零,对 $\alpha\geqslant$0.9 时达到最大(590 次)。

(6)对于任何 α,总 MDR 一直是 0.227 3,不管子系统的 MDR 如何变化。可见通过 TLDF,任务的切换次数大大减少,频繁的抢占由有条件抢占调度得到控制;而基于最晚截止期的 EDF 得到的两个子系统的控制性能损失分别是 17.139 6 和 57.108 0,截止期错失率分别是 0.169 0 和 0.277 1,有效利用率则是 0.773 0。采用最晚截止时间作为截止时间的 EDF 固然计算开销比较小,但其不足主要有:任务的实际截止期要早于最

晚截止期,这样对于硬截止期任务来说,可能会有更多的任务错失实际截止期, 从而影响系统的控制性能及其稳定性;EDF不能优先调度比较重要或关键的任务,而TLDF可通过适当地调整抢占阈值参数,确保重要的控制任务优先调度,相应的控制系统性能损失比较小,而非重要任务可以有较高的控制性能损失。

6 结束语

具有模糊截止期的控制任务是计算机控制中常常出现的问题,而对于具有不确定任务特征的控制任务的调度一般不能按实时系统中的处理方式来简单处理,在计算资源有限的情况下有必要研究这类控制任务的调度问题,在确保一定控制性能的前提下尽量提高计算资源的利用率。通过提出奉献度的概念,该文创新点是以管道流量泄漏为应用对象,将模糊调度设计应用于管道流量泄漏监测中, 提出最大奉献优先调度策略,奉献大的控制任务优先得到调度;为了控制在最大奉献优先调度策略中可能出现的任务间频繁切换,减少系统开销,又提出基于抢占阈值的最大奉献优先调度策略,当控制任务完成到一定额度时,拒绝让其他控制任务抢占。仿真表明通过约束任务间的抢占条件,既能减少由于任务间频繁切换所造成的系统开销,又能适量均衡各个子系统的控制性能损失。随着我国西部大开发及西气东输工程的实施,将有大量的油气管道投入建设和运行,为将泄漏事故造成的各种危害减少到最小,需要研究泄漏在线监测技术的算法,而在线监测系统的模糊调度设计对提高的泄漏监测灵敏度和定位精度具有直接影响和决定作用。笔者设计的在线监测系统已经投入生产,每年带来经济效益达1 500万元,完全符合工业现场要求,具有一定的实际意义和应用价值。

参考文献:

[1] Marti P,Fuertes J M,Fohler G.Improving quality of control using flexible timing constraints:Metric and scheduling[C]//Proceedings of the 23rd IEEE Real-Time Systems Symposium,Texas,USA,2002:91-100.

[2] Cervin A,Henriksson D,Lincoln B.How does control timing affect performance?Analysis and simulation of timing using Jitterbug and True Time[J].IEEE Control Systems Magazine,2003,23(3):16-30.

[3] 王守觉. 仿生模式识别———一种模式识别新模型的理论与应用[J].电子学报,2002,30(10):1417-1420.

[4] 安冬,王库,王守觉.高维空间点覆盖方法在物种计算机自动分类中的应用[J].电子学报,2006,34(2):277-281.

[5] Wang Shou jue,Lai Jiang liang.Geometrical learning,descriptive geometry,and bio-mimetic pattern recognition[J].Neuro-computing,2005,67:9-28.

(上接173页)

建立纹理特征向量与年龄之间对应关系的年龄估计函数,得到了最好的实验结果,年龄误差为3.89。

5 结论

提出了一种基于LBP、PCA与SVM回归相结合的年龄估计方法,能够有效地提取人脸年龄的纹理特征,并且通过SVM回归建立了纹理特征向量与年龄相对应的年龄估计函数。实验结果表明,通过此方法可以实现对人脸图像的快速准确的年龄估计,年龄误差控制在4岁左右。

参考文献:

[1] Lanitis A,Taylor C J,Cootes T F.Toward automatic simulation of aging effects on face images[J].IEEE Transaction on Pattern Analysis and Machine Intelligence,2002,24(4):442-445.

[2] Kwon Y H,da Vitoria Lobo N.Age classification from facial images[J].Computer Vision and Image Undenstanding,1999,74(1):1-21.

[3] Geng Xin,Zhou Zhi-hua,Smith-Miles K.Automatic age estimation based on facial aging patterns[J].IEEE Transaction on Pattern Analysis and Machine Intelligence,2007,29:2234-2240.

[4] Ojala T,Pietikainen M,Maenpaa T.Mulitiresolution gray-scale and rotation invariant texture classication with local binary patterns[J].IEEE Transaction on Pattern Analysis and Machine Intelligence,2007,24:971-978.

[5] 孙德山.支持向量机分类与回归方法研究[D].长沙:中南大学,2004.

[6] 王鹏,朱小燕.基于RBF核的SVM的模型选择及其应用[J].计算机工程与应用,2003,39(24):72-73.

(上接190页)

参考文献:

[1] Tur M,Pentland A.Eigenfaces for recognition[J].Journal of Cognitive Neuroscience,1991:71-86.

[2] Etemad K,Chellappa R.Discriminant analysis for recognition of human face images[J].Journal of the Optical Society of America,1997,14:1724-1733.

[3] Wiskott L,Fellous J M,Kuiger N,et al.Face recogniton by elastic bunch graph matching[J].IEEE Trans Pattern Analysis and Machine Intelligence,1997,19:775-779.

[4] Ahonen T,Hadid A,Pietikainen M.Face recognition with local binary patterns[C]//LNCS 3021:Computer Vision Proceedings,ECCV 2004.[S.l.]:Springer,2004:469-481.

[5] Ojala T,Pietikainen M,Maeopaa T.Multiresolution gray-scale and rotation invariant texture classification with local binary patterns[J].IEEE trans Pattern Analysis and Machine Intelligence,2002,24:971-987.

[6] Ojala T,Pietikainen M,Harwood D.A comparative study of texture measures with classification based on feature distributions[J].Pattern Recognition,1996,29:51-59.

[7] Philips P,Flynn P,Scruggs T,et al.Overview of face recognition grand challenge[C]//IEEE Computer Society Conference on Computer Vision and Pattern Recognition,2005,1:947-954.

[8] Mc Eliece R J.The Theory of Information and Coding[M].2nd ed.CUP,2002.

[9] Lahdenoja O,Laiho M,Paasio A.Reducing the feature vector length in local binary patten based face recognition[C]//IEEE International Conference on Image Processing,2005,2:11-14.

[10] Huang X S,Li S Z,Wang Y S.Jensen-Shannon boosting learning for object recognition[C]//IEEE Computer Society Conference on Computer Vision and Pattern Recognition,2005,2:144-149.