一种动态优先级实时任务调度算法

夏家莉[1)] 陈 辉[1)] 杨 兵[2)]

[1)](江西财经大学软件与通信工程学院 南昌 330013)

[2)](湖北大学教育学院 武汉 430062)

摘 要 现有实时任务调度算法往往根据任务的时间属性或者价值确定任务优先级,较少同时兼顾任务的价值和执行紧迫性.文中根据任务的价值和剩余执行时间讨论任务的剩余价值密度,根据任务的截止期和空余执行时间分析任务的紧迫性;然后综合任务的剩余价值密度和执行紧迫性,提出了动态分派任务优先级的DPA策略;最后提出了基于DPA的抢占调度算法DRTP. DRTP算法分析了任务抢占调度的各种可能条件,分析了系统中可能出现的颠簸调度,并给出避免颠簸的条件.仿真实验结果显示,与其它同类算法相比,DRTP算法能够提高系统价值收益,降低任务截止期错失率,并大大减少任务抢占的次数.

关键词 实时任务;剩余价值密度;执行紧迫性;动态优先级分派

中图法分类号 TP316 **DOI号**: 10.3724/SP.J.1016.2012.02685

A Real-Time Tasks Scheduling Algorithm Based on Dynamic Priority

XIA Jia-Li[1)] CHEN Hui[1)] YANG Bing[2)]

[1)](*School of Software and Communication Engineering, Jiangxi University of Finance and Economic, Nanchang* 330013)

[2)](*School of Education, Hubei University, Wuhan* 430062)

Abstract Most of the existing real-time scheduling algorithms assign the priorities of tasks according to tasks' time and value, but few of them can synchronously consider the value and urgency of a task. In this paper, for a real-time task, we firstly discuss its dynamic value density according to its value and remainder execution time, and analyze its execution urgency by considering its deadline and spare time. Based on the dynamic value densities and execution urgencies of the real-time tasks, we propose a strategy named DPA to dynamically assign the priorities of the real-time tasks. Additionally, we also present a DPA-based scheduling algorithm named DTRP. For the algorithm, we analyze all the situations of preemptive scheduling, discuss the thrashing and the condition of avoiding the thrashing. The experimental results show that the DRTP algorithm is prior to other analogous algorithms on the accrual value of real-time system, the deadline miss ratio and the preemptive number.

Keywords real-time task; remainder value density; execution urgency; dynamic priority assignment

1 引 言

最近几年,实时任务系统广泛应用到航天控制[1-2]、工业控制[3]、柔性制造网络控制[4]、机器人智能控制[5]、无线传感器网络[6-8]、云计算[9]、多处理器下多媒体流调度[10]以及嵌入式智能设备[11]中.实时任务系统是用来处理有定时限制工作负载的任务处

收稿日期:2008-11-08.本课题得到国家自然科学基金(60763002,60863016)、江西省自然科学基金(2008GZS0021)、湖北省教育厅科学技术研究计划优秀中青年人才项目资助.夏家莉,女,1965年生,博士,教授,博士生导师,主要研究领域为实时系统、实时数据库系统、软件工程.陈 辉,男,1976年生,博士,副教授,主要研究方向为实时数据库系统、数据挖掘技术. E-mail: comdoc@126.com.杨 兵(通信作者),男,1975年生,博士后,副教授,主要研究方向为移动数据库、实时任务处理技术. E-mail: yangbing@126.com.

理系统,任务处理的主要目标是进行任务调度从而使尽可能多的任务能够在任务截止期前完成执行,并实现系统性能到达最佳[12].然而,在时间紧急的实时任务系统中做出调度决策是一项非常困难的工作,因为该决策涉及到大量的实时系统参数和实时任务属性[13].

截止期最早最优先 EDF[14-15]算法根据任务截止期来决定任务执行顺序,让截止期最早的任务优先执行.空闲时间最短最优先 LSF[16]算法则将最高优先级指派给最短空闲时间的任务,保证最紧迫的任务优先执行. Semghouni 等人[17]在 EDF 算法的基础上,同时考虑任务的重要性,提出了一种新的调度策略 GEDF 来提高 EDF 算法处理系统过载的能力.为了提高周期性实时任务系统的效率,文献[18]提出了一种更为通用的实时任务模型,允许周期性任务的截止期和周期可以在某些周期内变化.文献[19]提出了一种计算周期性任务最小截止期的算法.文献[18-19]在保证系统可调度性的基础上,通过动态调整实时任务的截止期和周期,加快任务的调度速度,减少系统的等待时间,从而在系统过载的情况下提高任务的可调度性,进而提高系统的效率.为了克服 EDF 算法和可能 LSF 算法存在的不足,文献[20]根据任务的相对截止期和空闲时间,采用综合加权的方式综合任务的 EDF 优先级和 LSF 优先级设计任务的优先级,并通过加权参数调节两种优先级策略对表优先级影响的权重.

随着实时任务系统广泛应用到各种应用领域,具体应用也对实时任务调度的效率和灵活性提出了更高要求,单纯考虑任务的时间属性来确定任务优先级的方式已经不能满足需要. Burns 等人[1]指出,提高实时系统灵活性和效率的一种流行方式就是采用基于价值的优先级调度策略.简单地讲,基于价值的调度就是从一个任务集合中选择地执行某些任务,从而保证系统具有最佳的价值收益.而价值[5,12]可以认为是任务成功执行对系统的贡献,其值是应用系统决定的,是任务的固有属性.文献[5]结合汽车自动导航系统的具体应用,运用计量理论和决策分析方法给出了计算任务价值的系统化、合理化的方法.文献[2]根据任务的价值高低提出了价值最高最优先 HVF 算法,在系统高负载情况下,让价值最高的任务优先执行.为了提高系统在单位时间内的价值收益,文献[2,21]根据任务的价值及任务的最大执行时间,确定任务的价值密度,并由此提出了价值密度最大最优先 HVDF 算法,保证对系统价值收益贡献率高的任务优先执行.很显然, HVF 及 HVDF 都是静态优先级分配算法.在基于价值的调度系统中,当且仅当实时任务完成并提交时,系统才能获得相应的价值增益.当系统采用抢占的调度策略时,不管当前执行任务正执行到哪一个阶段,都有可能被新到达的其它任务抢占,甚至于夭折.为了保护执行事务不被其它任务抢占而夭折, Aldarmi 和 Burns[13]根据任务的价值及其剩余执行时间,提出了一种动态价值密度的实时调度策略 DVD,优化系统资源的利用率.随着执行任务剩余执行时间的减少, DVD 算法赋予执行任务的优先级会快速增加,直至无穷大.因此,不管执行任务在时间紧迫程度如何,它都很难被其它任务抢占,因此可能造成后到达的紧迫任务错失截止期.

在具体的实时应用系统中,时间紧迫的任务其价值未必高,而价值高的任务其执行时间未必紧迫.因此,片面依据任务的时间属性进行任务调度虽能保证时间紧迫性高的任务优先执行,提高任务的成功执行率,但可能使一些价值高的任务错过截止期,降低系统总收益.相反,片面强调系统价值收益,则可能使得那些低价值任务得不到执行的机会或者频繁地被高价值任务抢占而错失截止期,降低任务的成功执行率[22].

为了兼顾公平和效率,本文综合考虑任务价值、截止期与空余时间 3 个特征参数,提出了一种综合任务动态价值密度与执行紧迫性的动态优先级分派策略(Dynamic Priority Assignment, DPA).首先,为了合理保护执行任务不易被其它任务抢占,同时又给其它高紧迫性任务执行的机会, DPA 策略根据任务的价值和剩余执行时间提出了一种不同于文献[13]的动态价值密度计算方法.该方法将任务的价值密度控制在一个确定的区间$[\overline{VD_i}, p\times\overline{VD_i})$内,其中$\overline{VD_i}=\frac{V_i}{C_i}$为一个常数,仅与任务本身的固有属性有关,$p$ 是调节任务动态价值密度对优先级影响权重的系数.其次,为了提高时间高紧迫性任务的执行机会, DPA 策略根据任务的截止期和空余时间设计了一个新的实时任务执行紧迫性的评价指标$\delta_i=q^{\varphi_i}$,其中 φ_i为任务的执行强度,δ_i为任务执行紧迫性,其取值区间为$(0,1]$,q 是调节任务时间紧迫性对优先级影响权重的系数.通过选取不同参数 p 和q, DPA 策略可以改变任务动态价值密度与紧迫性对任务优先级的影响权重,提高了对不同应用环境的适应性.第三,在 DPA 策略的基础上,提出了

一种新的实时任务调度算法 DRTP(Dynamic Real-time Transaction Scheduling)，该算法详细分析了任务调度中可能出现的情况，讨论了任务抢占与非抢占调度的条件. 最后，论文还讨论了 DRTP 算法抢占调度中可能出现的系统颠簸[16]现象，并通过推理分析给出了避免系统颠簸条件.

2　任务模型

假设实时任务系统中实时任务集合为 $T=\{T_1,T_2,\cdots,T_n\}$，其中每个任务 T_i 都具有以下属性[23]：

(1) P_i，表示 T_i 的执行周期，若 T_i 为非周期任务，则假设其周期 P_i 为无穷大；

(2) D_i，表示 T_i 的相对截止期，$D_i \leq P_i$，即系统中同时仅存在任务的一个实例，因此在上下文中不区分任务及其实例；

(3) C_i，表示 T_i 的理论执行时间，$C_i \leq D_i$；

(4) b_i，表示 T_i 放行并准备执行的时间；

(5) e_i，表示 T_i 执行完成时间；

(6) d_i，表示 T_i 的绝对截止期，且 $d_i=b_i+D_i$；

(7) V_i，表示 T_i 的预期价值，若 T_i 在截止期前完成，则 T_i 对系统累积价值的贡献为 V_i；否则为 0.

假设 τ_i 为系统的当前时刻，那么根据任务在系统中执行状态，可以将系统中实时任务划分为如下几类：

(1) 执行任务，表示在当前执行周期内已放行但尚未执行完成的任务；

(2) 活动任务，表示在当前执行周期内已开始，但目前因被抢占而处于等待状态的任务；

(3) 等待任务，表示在当前执行周期内已经放行，但因未获得系统执行权而处于等待状态的任务；

(4) 休眠任务，表示在当前执行周期内已完成执行或者已夭折，目前正等待下一个执行周期到来的任务.

此外，针对该任务模型本文还作如下假定：

(1) 系统中任务间相互独立，即除 CPU 外，任务间无冲突资源与相互依赖关系；

(2) 任务不会自动挂起；

(3) 任务切换时间为 0 或者很小以致可以忽略.

3　任务优先级分派

本节将详细讨论影响实时任务优先级的两个最主要因素：任务的动态价值密度与执行紧迫性，并提出基于任务价值密度与紧迫性的优先级分派函数.

3.1　任务即时价值

在实时任务系统中，系统各种功能总是通过执行相关的任务来完成. 然而，由于系统各个功能在整个系统中重要性的不同，系统执行各个任务的重要程度也存在着明显的差异. 为了具体量化某个任务对于系统的重要性，本文使用任务的价值(Value，简记为 V)表示任务的重要性. 显然，任务价值是任务本身的固有属性，与任务的时间特征无关；此外，任务的价值也不是在任务成功执行的那个瞬间产生，而是随着任务的执行而逐渐积累的过程.

定义 1(即时价值).　当实时任务 T_i 开始执行后，任务累积产生的价值称为任务的即时价值(Immediate Value)，记为 IV_i. 若任务 T_i 执行了 t 个单位时间，则 T_i 的即时价值记为 $IV_i(t)$.

假设曲线 $IV_i(t)=k\times t^p$ 表示任务 T_i 即时价值产生的过程，其中 $p(p\geq 1)$ 反映任务即时价值产生速度的变化. 当 $p=1$ 时，表示任务即时价值匀速产生，当 $p>1$ 时，表示任务即时价值变加速产生. 显然，当 $t=C_i$ 时，$IV_i(t)=V_i$，故有 $V_i=k\times C_i^p \Rightarrow k=\frac{V_i}{C_i^p}$. 因此，任务的即时价值可以由下式得到

$$IV_i(t)=\frac{V_i\times t^p}{C_i^p} \tag{1}$$

在任务成功提交之前，任务所累积产生的即时价值只是一种可能价值收益，并没有给系统带来实实在在的价值收益. 当且仅当成功执行并正常提交后，任务才能够带给系统大小为 V_i 的价值收益，否则丢弃已产生的即时价值.

3.2　剩余价值密度

定义 2(平均价值密度).　任务预期价值与任务理论执行时间的比率称为任务的平均价值密度，记为 $\overline{VD}$.

显然，任务 T_i 的平均价值密度 $\overline{VD_i}=\frac{V_i}{C_i}$，它仅与任务本身的属性有关，而与任务的执行过程无关. 然而，任务的平均价值密度并不能反映任务即时价值产生的动态过程. 为了反映任务价值产生的变化速度，下面给出了任务剩余价值密度的概念.

定义 3(剩余价值密度).　任务在其剩余执行时间内，单位时间产生的价值称为任务的剩余价值密度(Remainder Value Density)，记为 RVD；若任务 T_i 已执行 t 个单位时间，则其剩余价值密度表示

为 $RVD_i(t)$.

根据定义,任务 T_i 的剩余价值密度可由下式得到

$$RVD_i(t)=\frac{V_i-IV_i(t)}{C_i-t}=\frac{(C_i^p-t^p)\times V_i}{C_i^p(C_i-t)} \quad (2)$$

其中,V_i 为 T_i 的预期价值;t 为 T_i 的已执行时间;$IV_i(t)$ 为 T_i 当前已产生的即时价值;C_i 为任务的理论执行时间. 显然,当 $p=1$ 时,有 $RVD_i(t)=\overline{VD_i}$,与 T_i 的已执行时间 t 无关.

定理 1. 若 $0\leq t<C_i$ 且 t 取定值时,任务 T_i 的剩余价值密度 $RVD_i(t)$ 关于参数 p 递增.

证明. 令 $f(p)=\frac{C_i^p-t^p}{C_i^p}$,因为

$$f'(p)=\frac{t^p(\ln C_i-\ln t)}{C_i^p}>0,$$

$f(p)$ 为关于参数 p 的增函数,所以,$RVD_i(t)=\frac{f(p)\times V_i}{C_i-t}$ 也关于参数 p 递增. 故原命题得证. 证毕.

定理 2. 若 $0\leq t<C_i$,$p>1$ 且 p 取定值时,任务 T_i 的剩余价值密度 $RVD_i(t)$ 关于 t 递增.

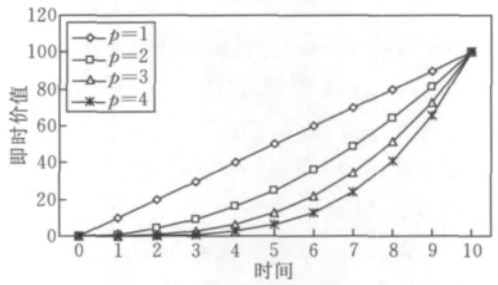
证明. 令 $f(t)=\frac{C_i^p-t^p}{C_i-t}$,故

$$f'(t)=\frac{(p-1)t^p+C_i^p-C_ipt^{p-1}}{(C_i-t)^2}.$$

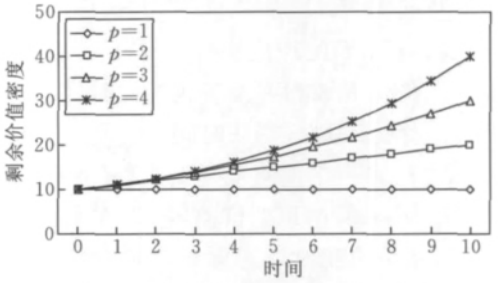
再令 $g(t)=(p-1)t^p+C_i^p-C_ipt^{p-1}$,因为 $g'(t)=p(p-1)t^{p-2}(t-C_i)$,故 $g'(t)<0$,即 $g(t)$ 为减函数,因此,有 $g(t)>g(C_i)\Rightarrow g(t)>0$,$f'(t)=\frac{(p-1)t^p+C_i^p-C_ipt^{p-1}}{(C_i-t)^2}=\frac{g(t)}{(C_i-t)^2}>0$,即 $f(t)$ 为增函数. 所以,$RVD_i(t)=\frac{f(t)\times V_i}{C_i^p}$ 也关于 t 递增. 故原命题得证. 证毕.

若称任务 T_i 在执行 $t(0\leq t<C_i)$ 个单位时间时的剩余价值密度与开始时刻剩余价值密度的比率为 T_i 剩余价值增长倍率,记为 $R_i(t)$,显然,$R_i(t)=\frac{RVD_i(t)}{RVD_i(t)_{t=0}}=\frac{C_i^p-t^p}{C_i^{p-1}(C_i-t)}=\frac{C_i-(t/C_i)^{p-1}t}{C_i-t}$,当 $p=1$ 时,$R_i(t)=1$;当 $p>1$ 时,$R_i(t)>1$,且 p 的值越大,$R_i(t)$ 越大. 当 $t\to C_i$ 时,T_i 剩余价值密度的增幅达到最大,即 $\max R_i(t)=\lim\limits_{t\to C_i}\frac{C_i^p-t^p}{C_i^{p-1}(C_i-t)}=\lim\limits_{t\to C_i}\left(\frac{(C_i-t)}{C_i^{p-1}(C_i-t)}\times\sum\limits_{\substack{j=0\\k=p-1}}^{k}C_i^jt^{k-j}\right)=p$. 也就是说,在任务 T_i 执行期间,其剩余价值密度的变化区间为 $[\overline{VD_i},p\times\overline{VD_i})$.

假设任务 T_i 的预期价值 $V_i=100$,理论执行时间 $C_i=10$,当 $p=1,2,3,4$ 时,T_i 的即时价值及剩余价值密度变化曲线如图 1 所示. 当 $p=1$ 时,在 T_i 执行期间,T_i 即时价值的产生速度稳定不变,剩余价值密度为常数. 当 p 的值大于 1 时,T_i 即时价值与剩余价值密度都随 T_i 执行时间的增加而增大,且当 p 的值越大,T_i 即时价值及剩余价值密度增速越快.



(a) 任务即时价值随时间变化曲线



(b) 任务剩余价值密度随时间变化曲线

图 1 变化曲线

任务仅在成功提交之后才能够给系统带来与其预期价值等值的价值收益,否则,其产生的即时价值无效. 若夭折一个已产生即时价值的任务,不但没有给系统带来收益,相反还浪费了任务执行时消耗的系统资源. 因此,完全有必要采取一种措施来保护那些已经开始执行的任务,使它们能够尽可能地提交. 在实时任务系统的四类实时任务中,休眠任务的执行周期尚未到达,可以不用考虑,下面我们比较其它 3 类任务. 对于实时系统中的某个任务 T_i,若 T_i 为等待任务(T_i 已执行时间 $t=0$),T_i 的剩余价值密度 $RVD_i(t)=\frac{V_i}{C_i}$ 为常数;若 T_i 为活动任务或执行任务($0<t<C_i$),T_i 的剩余价值密度 $RVD_i(t)=\frac{(C_i^p-t^p)\times V_i}{C_i^p(C_i-t)}$. 根据定理 2,显然有 $RVD_i(t)>\frac{V_i}{C_i}$. 同时,p 的取值越大,$RVD_i(t)$ 的值也越大. 因此,可以提高 T_i 的优先级,在一定程度上降低了 T_i 夭折的可能性,提高了系统的成功执行率.

3.3　紧迫性分析

在实时系统调度中，不仅要求系统价值收益最大化，同时也要求尽可能满足任务的时间约束，提高任务的成功执行率. 传统的基于任务时间属性的调度方法往往根据任务的截止期[14-15]或者空闲时间[16]来评判任务执行的紧迫性，然而，仅依据截止期或空余时间并不能准确地反映任务执行的紧迫性. 本节分析任务截止期、剩余执行时间及空闲时间的关系，提出一个新的评价指标来判定任务执行的紧迫性，即：为了保证任务时间约束，任务必须尽快开始执行的紧迫程度.

定义 4(执行强度).　完成任务所需的执行时间与任务空余时间的比称为任务的执行强度，记为 φ.

任务 T_i 的执行强度 $\varphi_i=\dfrac{C_i-t}{d_i-\tau_i}$，其中 τ_i 为当前时间，t 为 T_i 的已执行时间. 显然，任务执行强度越大，说明为了完成任务，任务空闲时间中需用于执行任务的时间份额越大.

定义 5(执行紧迫性).　为了保证任务截止期，任务需尽快执行的紧迫程度称为任务执行的紧迫性(Urgency)，记为 δ.

假设在 τ_i 时刻，任务 T_i 已执行了 t 个单位时间，这时，T_i 的执行紧迫性 $\delta_i(t)$ 可由下式(3)得到

$$\delta_i(t)=q^{\varphi_i}=q^{\frac{C_i-t}{d_i-\tau_i}} \tag{3}$$

其中 $q(q\geq 1)$ 为调节任务执行强度对任务执行紧迫性影响大小的参数. 显然，任务执行强度 φ 的取值区间为 $(0,1]$，因此任务执行紧迫性 δ 的取值区间为 $(1,q]$. 特别地，当 $q=1$ 时，表示任务执行的紧迫性为常数.

定理 3.　当 $q(q>1)$ 取定值时，任务执行的紧迫性随等待时间的增加而增高.

证明.　假设在 τ_i 时刻，任务 T_i 已执行了 t 个单位时间，其 T_i 放行后的累积等待时间(不包括 T_i 的执行时间)为 x.

因为 $d_i-\tau_i=D_i-x-t(0\leq x<D_i-t,0\leq t<C)$，其中，$d_i-\tau_i$ 表示 T_i 的空余时间. 所以上式(3)可改写为 $\delta_i(t)=q^{\frac{C_i-t}{D_i-t-x}}$.

令 $f(x)=\dfrac{C_i-t}{D_i-t-x}$，因为 $f'(x)=\dfrac{C_i-t}{(D_i-t-x)^2}>0$，故 $f(x)$ 关于 x 递增.

所以，$\delta_i(t)=q^{f(x)}$ 也关于 x 递增. 故原命题得证.　证毕.

当实时任务 T_i 开始放行时，其已执行时间 $t=0$，空闲时间为 D_i，这时 T_i 执行的紧迫性最小，为 $q^{\frac{C_i}{D_i}}$ $(C_i\leq D_i)$. 若任务 T_i 一直处于等待状态或者执行一段时间后被抢占而处于等待状态，随着等待时间增加，其执行紧迫性不断增加，到某个时刻，T_i 的空闲时间恰好仅够用于完成余下的执行任务时，也就是说，T_i 必须立即转为执行状态，否则将因为没有足够的执行时间而夭折. 这时 T_i 的执行紧迫性达到最大，等于 q. 因此，在 T_i 的执行期间内，其执行紧迫性的变化区间为 $[q^{\frac{C_i}{D_i}},q]$.

对于实时任务系统各种任务，其执行紧迫性的处理方式也是不同的：(1) 仅考虑能满足截止期任务的紧迫性，对于无法满足截止期的任务则直接进夭折处理；(2) 执行任务的紧迫性等于其最近一次开始执行时的紧迫性，并在其执行期间保持不变；(3) 活动任务与等待任务的紧迫性随其等待时间的增加而逐渐增大；(4) 休眠任务的执行周期尚未到达，不考虑其紧迫性.

3.4　动态优先级分派 DPA

前面，我们根据实时任务的价值与执行时间分析了任务的剩余价值密度，根据任务的截止期与空余时间分析了任务的执行紧迫性. 接下来，我们将综合考虑任务的剩余价值密度与执行紧迫性这两个方面的因素，提出一种实时任务优先级动态分派策略 DPA(Dynamic Priority Assignment)，该策略将致力于实现系统价值收益与任务成功执行率综合性能最优.

假设在 τ_i 时刻，非休眠任务 T_i 已执行 t 个单位时间，DPA 策略定义任务 T_i 的动态优先级为 $DyPri(T_i)$，其理论执行时间为 C_i，预期价值为 V_i，截止期为 d_i，则称 T_i 在 τ_i 时刻的动态优先级为 $DyPri(T_i)$，且 $DyPri(T_i)$ 可以由下式得到.

$$DyPri(T_i)=RVD(t)\times\delta(t)=\frac{(C_i^p-t^p)\times V_i}{C_i^p(C_i-t)}\times q^{\frac{C_i-t}{d_i-\tau_i}} \tag{4}$$

定义 6(基本优先级).　根据 DPA 策略，实时任务在其执行周期内优先级的最小值称为该任务的基本优先级.

根据 3.2 及 3.3 节的分析，当任务 T_i 放行时，其剩余价值密度及执行紧迫性均为最小值，分别为 $\dfrac{V_i}{C_i}$ 和 $q^{\frac{C_i}{D_i}}$. 因此，T_i 的基本优先级即最小优先级 $\min DyPri(T_i)=\min(RVD_i(t))\times\min(\delta_i(t))=q^{\frac{C_i}{D_i}}\times\dfrac{V_i}{C_i}$. 显然，$T_i$ 的基本优先级仅与任务本身属性

及参数 q 有关，而与 T_i 运行时的时间参数无关. 随着 T_i 等待时间或者已执行时间的增加，T_i 的剩余价值密度或执行紧迫性不断增加，或者二者均增加. 当某个时刻，T_i 的剩余价值密度和执行紧迫性均到达最大时，T_i 的优先级到达最大，$\max DyPri(T_i)=\max(RVD_i(t))\times\max(\delta_i(t))=pq\times\frac{V_i}{C_i}$.

假设当实时系统开始运行时，所有任务同时放行，这时，基本优先级最高的任务首先获得系统执行权. 然后，随着系统时间的推移，系统中各个任务的优先级动态变化. 对于执行任务，其执行紧迫性保持不变(3.3 节)，但其剩余价值密度随着已执行时间的增加而不断增加，这在一定程度上保护了执行任务不被其它任务抢占；对于等待任务与活动任务，其剩余价值密度保持不变，但随着其等待时间的增加，其执行紧迫性不断增加，这给它们提供了抢占系统执行权的机会. 此外，通过调节参数 p 与 q 的取值，还可以调节任务剩余价值密度及执行紧迫性对任务动态优先级的影响力. 若 p 值较大时，剩余价值密度大的任务将优先获得系统的执行权，故能提高系统累积价值收益. 此外，执行任务的剩余价值密度随已执行时间增加而增大，能够减少执行任务被抢占而夭折的概率，进而减少了任务抢占的次数，提高任务的成功执行率. 若 q 的值较大时，执行紧迫性高的任务将优先获得系统的执行权，可以增加价值密度低的任务参与系统执行的机会. 但是，等待任务或活动任务的紧迫性随等待时间增加而增高，使任务抢占的概率大大增加，可能会造成部分价值密度大的任务夭折而降低系统累积价值收益.

4 动态抢占调度策略 DRTP

前面，我们已经讨论了基于任务剩余价值密度及执行紧迫性的动态优先级分派策略 DPA. 本节中，我们将基于该策略提出一种新的实时任务动态抢占的调度算法.

4.1 系统颠簸及避免

在基于动态优先级的实时系统中，两个或多个任务优先级交替上升而导致任务之间反复抢占的现象，称为系统颠簸现象[16]. 实时任务之间的抢占需要进行上下文切换，消耗系统资源. 而系统颠簸则是一种频繁的任务抢占，将导致系统的额外开销大大增加. 根据 3.4 节分析，在基于任务剩余价值密度及执行紧迫性的动态优先级分派策略中，由于参数 p、q 取值不同时，可能造成执行任务与非执行任务(不包括休眠任务)优先级交替增大的现象. 为了避免可能出现的系统颠簸现象，本文设定一个任务抢占门限，避免优先级差别很小任务间的抢占.

假设给定一个颠簸避免系数 $\beta(\beta\geq1)$，对于任意两个任务 T_2 与 T_1，T_1 为执行任务. 某一时刻，它们的优先级分别为 $DyPri(T_1)$ 与 $DyPri(T_2)$. 如果仅当 $DyPri(T_2)\geq\beta\times DyPri(T_1)$ 时，方许可 T_2 抢占 T_1，那么可以避免 T_2 与 T_1 之间抢占. 考虑任务 T_2 与 T_1 的任意性，根据下面的定理 4，总能找到一个能够避免系统颠簸现象的颠簸避免系数 β.

定理 4. 给定一个任务集，总存在一个颠簸避免系数 β 能使得系统避免出现系统颠簸.

证明. 给定一个实时任务集，记其中平均价值密度最大的任务为 T_1，平均价值密度最小的任务为 T_2. 假设在 τ_i 时刻，T_1 为执行任务，且已执行了 $t_1(t_1>0)$ 个单位时间，T_2 为非执行任务，已执行了 $t_2(t_2\geq0)$ 个单位时间，这时 T_1 与 T_2 的优先级分别记为 $DyPri(T_1)$ 与 $DyPri(T_2)$. 这时，若允许任务 T_2 能够抢占 T_1，则有下式(5)中的条件成立：

$$DyPri(T_2)\geq\beta\times DyPri(T_1) \tag{5}$$

假设 T_2 抢占 T_1 后，执行了 x 个单位时间，即到了 τ_i+x 时刻，T_1 与 T_2 的优先级分别为 $DyPri'(T_1)$ 与 $DyPri'(T_2)$. 为了避免系统出现颠簸现象，我们强制要求，在 T_2 抢占 T_1 后，T_1 不能再去抢占 T_2. 因此，必须满足下式(6)中的条件：

$$DyPri'(T_1)<\beta\times DyPri'(T_2) \tag{6}$$

根据式(6)，显然有

$$\beta>\frac{DyPri'(T_1)}{DyPri'(T_2)}\Rightarrow\beta\geq\max\left(\frac{DyPri'(T_1)}{DyPri'(T_2)}\right).$$

下面考虑一种极端情况：假设在 τ_i 时刻，任务 T_1 为执行任务，其剩余价值密度最大，为 $p\times\overline{VD_1}$，$\overline{VD_1}=\frac{V_1}{C_1}$，执行紧迫性最小，为 $q^{\frac{C_1}{D_1}}$；任务 T_2 为等待任务，其剩余价值密度最小，为 $\overline{VD_2}$，$\overline{VD_2}=\frac{V_2}{C_2}$，执行紧迫性也最小，为 $q^{\frac{C_2}{D_2}}$. 此后，T_2 抢占 T_1，T_1 从执行状态转变为等待状态，其剩余价值密度保持不变，执行紧迫性逐渐增大；T_2 从等待状态转变为执行状态，其执行紧迫性保持不变，剩余价值密度逐渐增大. 到 τ_i+x 时刻，T_1 执行紧迫性从最小值增加到最大值 q. 假设 x 足够小，T_2 剩余价值密度增幅足够小，以致可以忽略，因此有 $\beta\geq\max\left(\frac{DyPri'(T_1)}{DyPri'(T_2)}\right)\Rightarrow\beta\geq$

$\frac{\max(DyPri'(T_1))}{\min(DyPri'(T_2))}=\frac{\overline{VD_1}\times p\times q}{\overline{VD_2}\times q^{\frac{C_2}{D_2}}}$.

因为在 τ_i 时刻，T_1 为执行任务，T_2 为等待任务，根据 3.4 节中的分析，显然有 $\overline{VD_1}>\overline{VD_2}$. 又因为 $q\geq 1$ 且 $C_2\leq D_2$，显然有 $q\geq q^{\frac{C_2}{D_2}}$，因此有

$$\beta\geq\frac{\overline{VD_1}\times p\times q}{\overline{VD_2}\times q^{\frac{C_2}{D_2}}}>1 \tag{7}$$

因此，当式(7)成立时，系统一定能够避免颠簸现象. 故原命题得证.　　证毕.

4.2　基于动态优先级的抢占调度

在实时任务系统中，休眠任务的执行周期尚未到达. 因此，在基于动态优先级的实时任务调度时，仅需考虑执行任务、活动任务以及等待任务. 在任务调度时，通过计算各个任务的动态优先级，并将活动任务及等待任务中优先级最高的任务与执行任务的优先级进行比较，并根据相应的策略调度，从而使系统获得最佳性能.

假设在 τ_i 时刻，T_1 为执行任务，且已执行了 $t_1(0\leq t_1<C_1)$ 个单位时间. 此时，记 T_1 的优先级为 $DyPri(T_1)$，它保证截止期的基本条件为 $d_1-\tau_i\geq C_1-t_1$. T_2 是所有活动任务及等待任务中优先级最高的任务，且已执行了 $t_2(0\leq t_2<C_2)$ 个单位时间. 此时，T_2 的优先级为 $DyPri(T_2)$，它保证截止期的基本条件为 $d_2-\tau_i\geq C_2-t_2$. 那么，基于动态优先级的抢占调度策略说明如下：

(1) 如果 $DyPri(T_2)<\beta\times DyPri(T_1)$，任务 T_2 未能满足抢占任务 T_1 的条件. T_1 继续执行，T_2 保持原状态；

(2) 如果 $DyPri(T_2)\geq\beta\times DyPri(T_1)$，任务 T_2 满足抢占任务 T_1 的条件. 此时，根据 T_2 的截止期可有如下两种不同的处理策略：

(a) 如果条件 $d_2-\tau_i-(C_1-t_1)\geq C_2-t_2$ 与 $d_1-\tau_i-(C_2-t_2)\geq C_1-t_1$ 同时成立，说明如果 T_2 等待 T_1 提交后才开始执行，T_2 仍能保证其截止期；而若 T_2 抢占 T_1，T_1 等待 T_2 提交后再继续执行，T_1 也能满足截止期，这时可按调度策略 I(参见 4.2.1 节)处理；

(b) 如果条件 $d_2-\tau_i-(C_1-t_1)<C_2-t_2$ 成立，说明若 T_2 不抢占 T_1，T_2 将无法保证截止期，因此必须实施抢占. 这时，根据 T_1 被抢占后是否夭折，可分如下两种不同处理策略：

(i) 如果条件 $d_1-\tau_i-(C_2-t_2)\geq C_1-t_1$ 成立，则说明 T_2 抢占 T_1，T_1 等待 T_2 提交后继续执行仍能满足截止期，这时可按调度策略 II(参见 4.2.2 节)处理；

(ii) 如果条件 $d_1-\tau_i-(C_2-t_2)<C_1-t_1$ 成立，则说明 T_2 抢占 T_1 后，T_1 将无法满足截止期而夭折，这时则按照调度策略 III(参见 4.2.3 节)处理.

4.2.1　调度策略 I

实施调度策略 I 的前提条件是：任务 T_2 满足抢占任务 T_1 的基本条件，且在无其它影响条件下，无论 T_2 抢占 T_1 与否，T_1 与 T_2 都能满足截止期. 这时可有两种不同的处理策略：乐观处理策略与悲观处理策略.

(1) 乐观处理策略

乐观处理策略是任务 T_2 不抢占任务 T_1，T_2 等待 T_1 提交后才开始执行. 乐观处理策略的处理步骤为：任务 T_1 继续执行，等到 T_1 成功提交后，将 T_2 的优先级设定为 $\max(DyPri'(T_1),DyPri'(T_2))$，其中，$DyPri'(T_1)$ 为 T_1 等待 T_2 执行(即等待 C_2-t_2 个单位时间)，直到 T_2 提交时的优先级；$DyPri'(T_2)$ 为 T_2 等待 T_1 执行(即等待 C_1-t_1 个单位时间)，直到 T_1 提交时的优先级.

(2) 悲观处理策略

悲观处理策略是 T_2 抢占 T_1. 悲观处理策略的处理步骤为：暂停任务 T_1 的执行进程，执行任务 T_2，T_1 等待到 T_2 提交后再继续执行.

显然，乐观处理策略可以减少任务间不必要的上下文切换，提高系统的效率，而悲观处理可以避免因其它因素影响导致高优先级任务 T_2 无法满足截止期. 在实际的实时任务系统中，实时任务的数量很多. 如果 T_2 具有较高的基本优先级，即使等待 T_1 提交后，T_2 仍然有较大可能性获得系统执行权，那么，则采用乐观处理策略，否则，采用悲观处理策略.

4.2.2　调度策略 II

调度策略 II 也称为不夭折抢占调度策略，具体处理步骤为：暂停任务 T_1 的执行进程，执行任务 T_2，T_1 等待到 T_2 提交后再继续执行.

4.2.3　调度策略 III

若任务 T_1 被任务 T_2 抢占后会错过截止期而夭折，那么，夭折 T_1 将使 T_1 已产生的即时价值无效. 因此，若 T_2 夭折 T_1，T_2 需要弥补因 T_1 夭折而损失的即时价值. 假设在 τ_i 时刻，T_1 已执行的时间为 t_1，T_2 已执行的时间为 t_2，则若 T_2 能抢占 T_1，则还应满足如下条件：

$$\frac{(C_2^p-t_2^p)}{C_2^p(C_2-t_2)}\times\left(V_2-\frac{V_1\times t_1^p}{C_1^p}\right)\times q^{\frac{C_2-t_2}{d_2-\tau_i}}\geq$$

$$\beta\times\frac{(C_1^p-t_1^p)}{C_1^p(C_1-t_1)}\times V_1\times q^{\frac{C_1-t_1}{d_1-\tau_i}} \quad (8)$$

其中$\left(V_2-\frac{V_1\times t_1^p}{C_1^p}\right)$表示 T_2 补偿 T_1 夭折损失后的预期价值，$\frac{V_1\times t_1^p}{C_1^p}$表示 T_1 夭折时已产生的即时价值.

综上分析，基于动态优先级的实时任务调度算法具体如下：

Notations:

- $C, V, d, \tau_i, t, p, q, \beta$ have been depicted in the paper
- T_c the executing transaction
- Q the queue of transactions including waiting transactions and active transactions
- T_i the transaction in Q which has the maximum priority

```
Procedure: PrioritAssignment(T_j)   //DPA strategy
Begin
  if (T_j is T_c or T_j ∈ Q)
                          //T_j is a non-sleeping transaction
    DyPri(T_j)=RVD_j(t)×δ_j(t);
  else                    //T_j is a sleeping transaction
    DyPri(T_j)=0;
End

Procedure: TransSchudling(T_c, Q) //DRTP algorithm
Begin
  Set the value of β based on formula (7);
  For each (transaction T_j in the Q)
    Call PrioritAssignment (T_j);
  DyPri(T_i)=max(DyPri(T_j)), here T_j ∈ Q;
  if DyPri(T_i)≥β×DyPri(T_c) then
    if d_i−τ_i<C_i−t_i+C_c−t_c then
      if d_c−τ_i<C_c−t+C_i−t_i then
        if formula (8) holds then
          Schedule the transactions according to strategy
          III;
        else
          T_c continues running and T_i keeps its status;
      else
        Schedule the transactions according to strategy II;
    else
      Schedule the transactions according to strategy I;
  else
    T_c continues running and T_i keeps its status;
End
```

5 仿真实验

所有实验均在 CPU 为 AMD Athlon 2.0 GHz，内存为 512 MB 的 PC 机上运行，实验程序采用 C 语言实现. 实验中所有任务的时间参数 C、D 与 P 均随机产生，且满足 $C<D\leq P$.

实验首先分析算法 DRTP 的基本性能，然后将其同 GEDF[17]、HVF[2] 及 DVD[13] 算法进行对比分析. 实验中采用 3 项性能指标来评价算法优劣：系统累积价值收益[24](Total Value，TV)、任务抢占次数(Preemption Number，PN)以及任务截止期错失率(Miss Deadline Ratio，MDR).

5.1 基本性能分析

在基本性能实验中，我们假定实时系统包括 4 个实时任务(如表 1 所示)，且系统负载为 2.0. 实验中，通过 3 组实验分别变换参数 p、q 以及 β 的值来观察 DRTP 算法的性能指标.

表 1 基本性能实验所使用任务集

任务	C	D	S	V	VD
T_1	9	11	16	60	6.667
T_2	7	8	11	80	11.429
T_3	2	7	8	40	20.0
T_4	5	6	9	30	6.0

实验 1. 固定参数 $q=2$，改变参数 p 的值，分析 p 对算法性能的影响，实验根据式(7)选取参数 β 的值. 实验结果如图 2 所示，随着 p 值增大，任务剩余价值密度对优先级的影响增大，任务执行紧迫性对优先级的影响相对减小. 因此在任务调度时，剩余价值密度高的任务优先获得执行权，且不易被抢占，大大提高了价值密度大的任务的成功执行率及系统的累积价值收益(如图 2(a)所示). 此外，q 值一定时，颠簸避免系数 β 的值随 p 值增大而增大，使任务间抢占的困难加大. 因此，如图 2(b)所示，当 p 值增大时，任务之间的抢占次数逐渐减少. 根据 3.2 节中的分析，当 p 值增大时，执行任务的动态价值密度增幅越大，因而不易被抢占，保证执行任务能够正常完成. 因此，如图 2(c)所示，任务截止期错失率随 p 的增加而减小.

此外，图 2 中所示的 3 项性能指标都说明：当 p 增大到一定值(等于 2.5)后，系统性能基本不变. 这是因为，当 p 值增大到一定值时，任务的剩余价值密度对优先级的影响足够大，以致于可以忽略任务执行紧迫性对优先级的影响. 这时，即使继续增大 p 值，任务优先级也不会有明显的变化，因此对任务执行的结果也没有明显的影响.

实验 2. 固定参数 $p=2$，改变参数 q 的值，分析 p 对系统性能的影响，同样地，根据式(7)选取参数 β 的值. 实验结果如图 3 所示，随着 q 值增大，任务执行紧迫性对优先级的影响增大，任务剩余价值密度对优先级的影响相对减小. 因此在任务调度时，执行紧迫性高的任务优先获得系统执行权. 因此，如

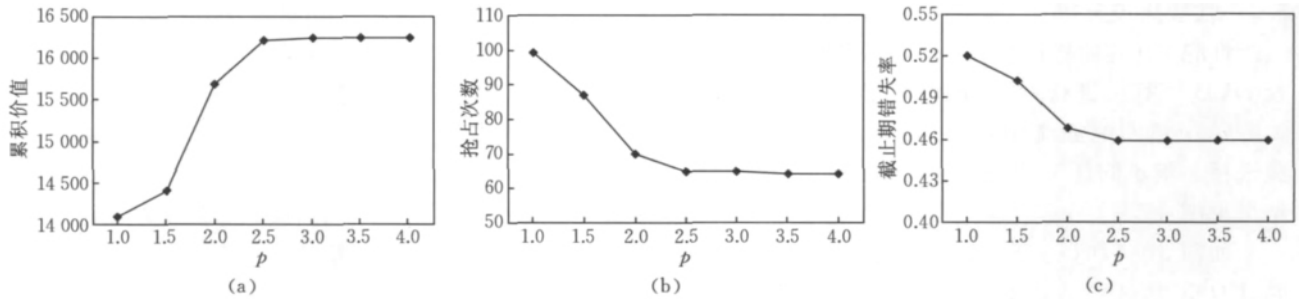


图 2　负载为 2.0，变化参数 p 时的系统性能分析

图 3(a)所示，系统的价值总收益随着 q 值增大而逐步减小. 此外，由于执行任务的紧迫性在执行期间保持不变，而非执行任务（活动任务与等待任务）的紧迫性随着其截止期的临近而不断增加. 在任务调度过程中，执行任务与非执行任务的优先级可能交替上升，且 q 值越大，这种优先级交替上升的速度越快. 因此，如图 3(b)所示，任务间抢占的概率大大增加. 此外，由于执行任务与非执行任务频繁相互抢占，很容易造成任务因无法获得足够的 CPU 时间而错过截止期，导致任务截止期错失率增大，结果如图 3(c)所示.

同样地，实验 2 的 3 项性能指标都说明：当参数 q 增大到一定值(3.5)以后，任务紧迫性对任务优先级的影响足够大，从而可以忽略任务价值密度对任务优先级的影响. 这时，即使再增大 q 的值，也不会造成任务优先有明显的变化，因此对任务执行的结果也不会有明显的影响.

实验 3.　固定参数 $p=2$，$q=2$，改变颠簸避免系数 β 的值，分析 β 对系统性能的影响. 根据 4.1 节分析，若保证系统无颠簸，β 理论取值应不小于 7.5. 如图 4 所示，当 β 取值很小时，任务间可能出现相互抢占现象，造成任务错失截止期的概率增大，系统累积价值收益减少. 随着 β 值增大，任务间抢占的难度增大，系统抢占次数以及任务截止期错失率大大减少，系统累积价值收益增大.

从图 4 还可以看出，当 β 大于 3 时，系统的抢占次数、任务截止期错失率及系统价值收益基本保持不变. 这说明在实验中，当 β 大于 3 时，系统中出现的抢占都是正常的，基本上不存在任务间相互交替抢占现象. 这时，继续增大 β 的值，不会明显减少任务之间抢占次数，对系统累积价值收益以及任务截止期错失率影响都很小.

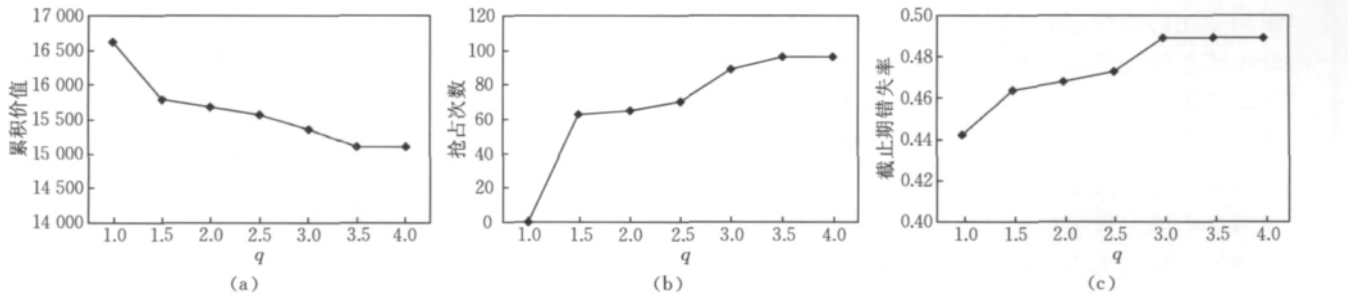


图 3　负载为 2.0，变化参数 q 时的系统性能分析

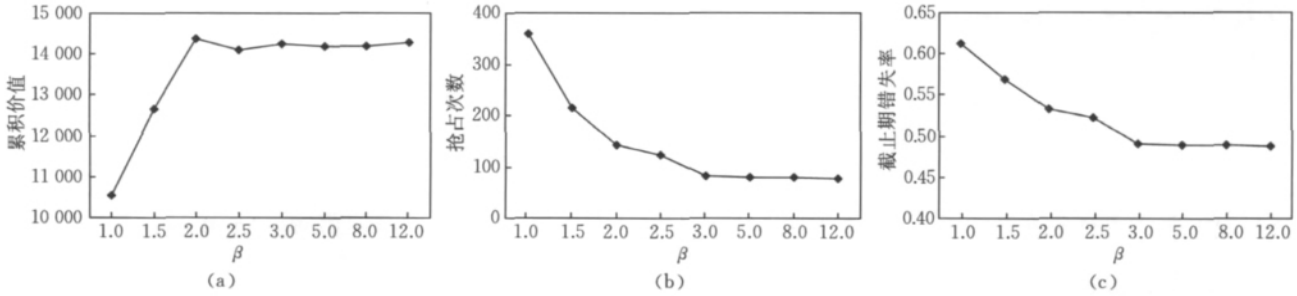


图 4　负载为 2.0，变化参数 β 时的系统性能分析

5.2 性能比较分析

性能对比实验将DRTP算法与GEDF[17]、HVF[2]及DVD[13]算法进行对比分析，实验分别在负载为0.5～3.0的实时系统中进行.每种负载条件下，实验选择3组不同任务集分别进行并取平均值，实验结果见图5.

如图5(a)和(b)所示，当系统负载较小(≤ 1)时，DRTP算法并没有任何优势，系统的价值收益TV及截止期错失率MDR均比GEDF算法差.这是因为，当系统负载较低时，只要能够合理安排截止期早的任务优先执行就能保证绝大部分任务正常完成，而GEDF算法在这方面做得最好.而基于价值(密度)的调度算法则优先选择价值(密度)大的任务优先执行，可能造成截止期早的任务错过截止期而夭折，导致系统累积价值收益减少和任务截止期错失率增大.当系统负载较高时，由于系统资源限制，只能够保证部分任务能够正常完成，这时只有通过合理安排任务执行的顺序，才可获得最佳的系统性能.当系统负载大于1.0时，DRTP算法在系统累积价值收益及任务截止期错失率这两个方面的指标明显优于GEDF算法，尽管HVF算法在系统累积价值收益这一指标上与DRTP算法相差不大，但是，相对HVF算法，DRTP算法大大降低了任务截止期错失率.此外，如图5(c)所示，无论系统负载如何，在任务抢占次数这这一指标上，DRTP算法总是最优于其它3种算法.

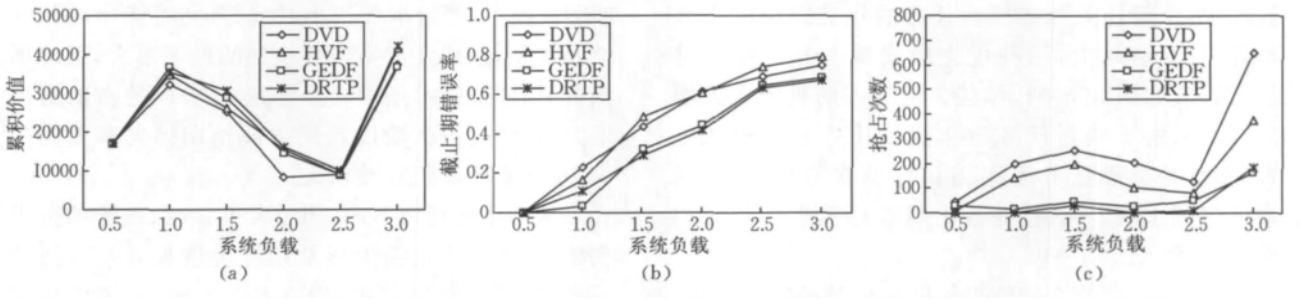


图5 4种算法在不同系统负载下的性能对比分析

6 总 结

论文研究了任务剩余价值密度及执行紧迫性随时间变化的特性，并综合这两方面因素，提出了动态分配任务优先级的DPA策略，由此提出了基于任务动态优先级的实时任务调度算法DRTP.该算法可通过改变参数p与q来调节任务剩余价值密度与执行紧迫性对任务优先级影响的权重，提高了算法对满足不同应用需要的灵活性.此外，算法还讨论了任务抢占与非抢占调度的条件，讨论了系统可能出现的颠簸现象及避免颠簸的条件.最后，实验仿真结果显示，当系统负载较高时，DRTP算法能够大大减少任务的抢占次数，降低任务截止期错失率，并提高系统累积价值收益.

参 考 文 献

[1] Burns A, Prasad D, Bondavalli A, Di Giandomenico F, Ramamritham K, Stankovic J, Strigini L. The meaning and role of value in scheduling flexible real-time systems. Journal of Systems Architecture, 2000, 46(4): 305-325

[2] Jensen E D, Locke C D, Toduda H. A time-driven scheduling model for real-time operating systems//Proceedings of the IEEE Real-Time Systems Symposium. San Diego, CA, USA, 1985: 112-122

[3] Kim Dong-Sung, Choi Dong-Hyuk, Mohapatra Prasant. Real-time scheduling method for networked discrete control systems. Control Engineering Practice, 2009, 17(5): 564-570

[4] Savkin A V, Somlo J. Optimal distributed real-time scheduling of flexible manufacturing networks modeled as hybrid dynamical systems. Robotics and Computer-Integrated Manufacturing, 2009, 25(3): 597-609

[5] Prasad D, Burns A. A value-based scheduling approach for real-time autonomous vehicle control. Robotica, 2000, 18(3): 273-279

[6] Saifullah A, Xu You, Lu Chenyang, Chen Yixin. Real-time scheduling for wireless hart networks//Proceedings of the 31st IEEE Real-Time Systems Symposium. San Diego, CA, USA, 2010: 150-159

[7] Virmani Deepali, Jain Satbir. Real time scheduling with virtual nodes for self stabilization in wireless sensor networks. International Journal of Information Technology and Knowledge Management, 2011, 4(2): 477-483

[8] Dong Wei, Chen Chun, Liu Xue, Zheng Kougen, Chu Rui, Bu Jiajun. Fit: A flexible, lightweight, and real-time scheduling system for wireless sensor platforms. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(1): 126-138

[9] Liu Shuo, Quan Gang, Ren Shangping. On-line scheduling of real-time services for cloud computing//Proceedings of the

6th World Congress on Services. Miami, FL, USA, 2010: 459-464

[10] Rhu J-H, Sun J-H, Kim K, Cho H, Park J K. Utility accrual real-time scheduling for (m, k)-firm deadline-constrained streams on multiprocessors. Electronics Letters, 2011, 47(3): 316-317

[11] Yang Chuanyue, Chen Jianjia, Thiele Lothar, Kuo Teiwei. Energy-efficient real-time task scheduling with temperature-dependent leakage//Proceedings of the 2010 ACM Symposium on Applied Computing. Sierre, Switzerland, 2010: 9-14

[12] Haritsa Jayant R, Carey Michael J, Livny Miron. Value-based scheduling in real-time database systems. VLDB Journal, 1993, 2(2): 117-152

[13] Aldarmi S A, Burns A. Dynamic value-density for scheduling real-time systems//Proceedings of the 11th Euromicro Conference on Real-Time Systems. York, England, UK, 1999: 270-277

[14] Liu C L, Layland James W. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 1973, 20(1): 46-61

[15] Haritsa J R, Livny M, Carey M J. Earliest deadline scheduling for real-time database systems//Proceedings of the 12th IEEE Real-Time Systems Symposium. San Antonio TX, USA, 1991: 232-243

[16] Jin Hong, Wang Hong-An, Wang Qiang, Dai Guo-Zhong. An improved least-slack-first scheduling algorithm. Journal of Software, 2004, 15(8): 1116-1123(in Chinese)
(**金宏，王宏安，王强，戴国忠. 改进的最小空闲时间优先调度算法. 软件学报**, 2004, 15(8): 1116-1123)

[17] Semghouni Samy, Amanton Laurent, Sade Bruno, Berred Alexandre. On new scheduling policy for the improvement of firm RTDBSs performances. Data & Knowledge Engineering, 2007, 63(2): 414-432

[18] Chantem T, Wang Xiaofeng, Lemmon M D, Hu X S. Period and deadline selection for schedulability in real-time systems//Proceedings of the Euromicro Conference on Real-Time Systems. Prague, 2008: 168-177

[19] Balbastre P, Ripoll I, Crespo A. Minimum deadline calculation for periodic real-time tasks in dynamic priority systems. IEEE Transactions on Computers, 2008, 57(1): 96-109

[20] Jin Hong, Wang Hong-An, Wang Qiang, Dai Guo-Zhong. An integrated design method of task priority. Journal of Software, 2003, 14(3): 376-382(in Chinese)
(**金宏，王宏安，王强，戴国忠. 一种任务优先级的综合设计方法. 软件学报**, 2003, 14(3): 376-382)

[21] Buttazzo G, Spuri M, Sensini F. Value vs. deadline scheduling in overload conditions//Proceedings of the 19th IEEE Real-Time Systems Symposium. Pisa, Italy, 1995: 90-99

[22] Abbott R, Garcia-Molina H. Scheduling real-time transactions. ACM SIGMOD Record, 1988, 17(1): 71-81

[23] Fohler G, Lennvall T, Buttazzo G. Improved handling of soft a periodic tasks in offline scheduled real-time systems using total bandwidth server//Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation. Antibes-Juan les Pins, France, 2001: 151-157

[24] Li Peng, Ravindran B. Fast, best-effort real-time scheduling algorithms. IEEE Transactions on Computers, 2004, 53(9): 1159-1175

XIA Jia-Li, born in 1965, Ph. D., professor. His research interests include real-time system, real-time database system, and software engineering.

CHEN Hui, born in 1976, Ph. D., associate professor. His research interests include real-time database system, data mining.

YANG Bing, born in 1975, post-doctoral, associate professor. His research interests include mobile database, real-time task processing.

Background

Real-time task scheduling, one of the most important research topics in real-time system, can efficiently manage the tasks with the limitations of time and load, and its goal is to properly schedule the tasks and make sure as more tasks as possible to be accomplished before their deadlines. At present, most of the real-time tasks scheduling algorithms assign the priorities of tasks according to three attributes, deadlines, slack times and value. However, a transaction with the nearest deadline might not always have the biggest value, nor the least slack time, and vice versa.

In order to improve the holistic performance of a real-time system, this paper analyzes the dynamic value density and execution urgency of a real-time task according to its value, deadline and execution time, and proposes a dynamic priority assignment strategy (named DPA), in which two parameter p and q are used to make a tradeoff between the weights of the dynamic value density and urgency of the task on its priority. Additionally, we also present a DPA-based scheduling algorithm named DTRP. Compared with the analogous strategies, the proposed solution could improve the gained-value of the system, reduce the deadline miss ratio.

This work is partly supported by the National Natural Science Foundation of China (Nos. 60763002, 60863016), the Natural Science Foundation of Jiangxi Province of China (No. 2008GZS0021), the Educational Commission Research Project for the Excellent Youth Scholars of Hubei Province of China.