基于动态优先级策略的最优软非周期任务调度算法

涂 刚 阳富民 卢炎生
(华中科技大学计算机科学与技术学院 武汉 430074)
(tugang@163.net)

摘 要 周期任务与非周期任务的混合调度是实时调度研究的一个重要方向. 通过定义"调度"和"逆调度",对实时周期任务集在使用 EDF 算法调度时的可挪用时间进行分析,求出了周期任务集在使用 EDF 调度时的最大可挪用时间. 在此基础上,提出用于缩短非周期任务响应时间和周转时间的调度算法——ISA(idle stealing algorithm). ISA 算法充分使用最大可挪用时间,在保证周期任务满足最后期限的同时能取得非周期任务的最优响应时间和周转时间. 证明了 ISA 算法的最优性,并使用仿真实验进行了性能验证.

关键词 非周期任务;调度;逆调度;可挪用时间;EDF 算法

中图法分类号 TP316.2

An Optimal Scheduling Algorithm for Soft Aperiodic Tasks

TU Gang, YANG Fu-Min, and LU Yan-Sheng
(*School of Computer Science and Technology, Huazhong University of Science & Technology, Wuhan* 430074)

Abstract The problem of hybrid scheduling of hard period task and soft aperiodic task is important part of the research of real-time scheduling. To analysis the idle and movable time of period task set scheduled by earliest deadline first scheduling algorithm, two definitions scheduling and converse scheduling are given. These two definitions are used to calculate the max movable time of period task set. Using the max movable time, an algorithm named idle stealing algorithm (ISA) is given to decrease the response time of aperiodic task. ISA makes full use of the max movable time of period task set. It can largely decrease the response time of aperiodic tasks while guaranteeing the deadline of period tasks. The ISA algorithm provides the shortest response time of aperiodic task and is proved to be optimal by experiments.

Key words aperiodic tasks; scheduling; converse scheduling; idle time; EDF algorithm

1 引 言

为了缩短硬周期任务和软非周期任务混合调度中软非周期任务的响应时间,人们提出了各种算法,这些算法可以归为 3 类:①后台执行法;②带宽保留法;③挪用法.

后台执行法是将软非周期任务作为后台任务执行,该方法保证了硬周期任务的执行,但它没有采取有效措施来减少软非周期任务的响应时间.

带宽保留法包括 PE(priority exchange)[1], DS(deferrable server)[1], SS(sporadic server)[2], MS(multiple server)[3] 等. 这些算法的共同点是为软非周期任务的执行保留了一定的 CPU 时间(带宽),软非周期任务在使用完带宽后需要等待下次带宽被充满后才能再次投入运行. 带宽保留法能大大提高软非周期任务的响应时间,但它们在保留带宽的同时限制了任务对 CPU 的使用.

挪用法的基本思想是:在保证硬实时周期任务不错过最后期限的同时,尽量使用周期任务集"挪出"的时间[4]. 若挪用不当会造成周期任务集不可调度,所以如何挪用和挪用多少时间是挪用法要考

收稿日期:2003-06-24;修回日期:2003-11-15
基金项目:国家"八六三"高技术研究发展计划基金项目(2002AA445010)

虑的主要问题. 周期任务集在使用不同调度算法时, 可挪用时间是不同的. 通常分为基于固定优先级的算法[5~7]和基于动态优先级的算法[8~12].

Lehoczky 等人在文献[5]中以及 Tia 等人在文献[6]中提出的基于固定优先级的算法获得了较好的效果, 是基于固定优先级算法的代表. 何军等人在文献[7]中提出了一种将固定优先级算法和动态优先级算法相结合的 ODD 算法. ODD 算法充分利用了动态优先级调度算法可使 CPU 使用率达到 1 的优点, 取得了较大的可挪用时间. 但由于固定优先级算法对 CPU 的使用率低, 所以这些挪用算法的效率不是很高.

基于动态优先级的算法能够获得很好的响应时间, 但使用动态优先级调度时, 周期任务的最大挪用时间的理论计算复杂度为 $O(n!)$(n 为周期任务的个数). 这使得进行直接计算将很费时间, 所以现有算法多是近似算法. 如 Buttazzo 的 TB(N)算法[8], Ripoll 的 EESS 算法[9].

Spuri 等人的 EDL 算法[10,11]基于 Chetto 等人在文献[13]中提出的理论. EDL 算法对周期任务执行过程中的空闲时间进行分析, 可以精确求得周期任务集的最大可挪用时间. 但由于其只能从任务集的角度计算可挪用时间, 忽略了单个任务的特性, 所以计算量过大, 其计算复杂度为 $O(N)$[13](N 为周期任务集在一个超周期中的到来次数之和).

本文通过定义"调度"和"逆调度", 提出一种精确求解周期任务集在使用 EDF 算法[14]调度时最大可挪用时间的方法. 在此基础上, 提出了用于缩短非周期任务响应时间的调度算法——ISA. ISA 充分使用了最大可挪用时间, 能取得 EDF 调度下非周期任务的最短响应时间和最小周转时间, 且算法复杂度降为 $O(n)$(n 为周期任务数). 本文证明了 ISA 的最优性, 并且使用仿真实验进行了验证.

2　周期任务集最大可挪用时间

2.1　实时系统模型与基本假设

我们使用以下的模型描述实时系统: 实时任务被分为硬实时周期任务和软实时非周期任务. 硬实时周期任务集记为 Π, $\Pi=\{T_1, \cdots, T_n\}$. T_i($1\leqslant i\leqslant n$)代表周期任务, $T_i=(S_i, C_i, D_i, P_i)$, 其中 S_i 为任务的到来时间, C_i 为任务的最大执行时间(WCET), D_i 为任务的最后期限, P_i 为任务的周期. 记 $\tau_{ij}=(s_{ij}, c_{ij}, d_{ij}, p_{ij})$为周期任务 T_i 的第 j 次到来, 其中 $s_{ij}=S_i+(j-1)\times P_i$, $d_{ij}=s_{ij}+D_i$, $c_{ij}=C_i$, $p_{ij}=P_i$. 软非周期任务集记为 Γ, $\Gamma=\{J_1, \cdots, J_n\}$. J_i($1\leqslant i\leqslant n$)代表非周期任务, $J_i=(S_i, C_i, D_i)$, 其中 S_i 为任务的到来时间, C_i 为任务的最大执行时间, D_i 为任务的最后期限.

周期任务集的超周期(hyperperiod)记为 H, $H=LCM(P_1, \cdots, P_n)$, LCM 是最小公倍数函数. 使用 EDF 调度时, 周期任务的执行在任何时刻 t 和时刻$(t+k\times H)(k\in N)$是相同的, 所以只需讨论任务集在一个超周期中的执行情况[9], 本文只对任务集在第 1 个超周期中的运行情况进行讨论. 周期任务的 CPU 使用率用 U_Π 表示, 这里 $U_\Pi=\sum(C_i/P_i)$. 我们知道, 在使用 EDF 算法调度周期任务时, U_Π 的上确界为 1[14].

为了集中讨论所关心的问题, 还作了下面的一些假设:

(1) 任务切换和进程调度所使用的 CPU 时间忽略不计.

(2) 任务在到来后即可投入运行且可被抢占.

(3) 除非被抢占, 任务不会挂起.

(4) 任务之间相互独立.

(5) 非周期任务的调度遵循先到先服务原则.

(6) 周期任务的最后期限与周期相等, $D_i=P_i$. 周期任务到来时间 $S_i=0$.

(7) $U_\Pi\leqslant 1$.

2.2　调度和逆调度

为了更好地描述周期任务集的调度情况, 我们定义了"调度"和"逆调度". 它们是时间段的集合, 用于纪录任务集在其超周期中的执行情况. "调度"和"逆调度"是相对于超周期 H 对称的概念, 下面是相关的定义及其性质.

定义 1. 任务执行区间: 任务 T_i 在时刻 s_i 得到 CPU 进入执行状态, 在时刻 e_i 让出 CPU 退出执行状态, 若 T_i 在从 s_i 到 e_i 的时间段内从未被抢占 CPU, 则称从 s_i 到 e_i 的时间段为 T_i 的任务执行区间, 记为$[s_i, e_i)$.

定义 2. 调度 ξ: 调度 ξ 记载了使用 EDF 算法进行调度时, 周期任务集 $\Pi=\{T_1, \cdots, T_n\}$在一个超周期中的执行过程. 定义 $\xi=\{\xi_1, \cdots, \xi_n\}$, 其中 ξ_i($1\leqslant i\leqslant n$)记载了任务 T_i 的执行过程, $\xi_i=\{\xi_{i1}, \cdots, \xi_{ik}\}$, $k=H/P_i$. ξ_{ij}($1\leqslant j\leqslant k$)记载了任务到来 τ_{ij}的执行过程, 它是任务执行区间的集合. $\xi_{ij}=\{[s_1, e_1), \cdots, [s_p, e_p)\}$, 这里 $s_q<e_q<s_{q+1}$($1\leqslant q\leqslant p$).

例**1**. 图 1(a)是周期任务集 $\Pi=\{T_1=(0,2,6,6),T_2=(0,5,10,10)\}$在使用 EDF 进行调度时的执行情况,对应的调度为 $\xi=\{\{\xi_{11},\xi_{12},\xi_{13},\xi_{14},\xi_{15}\},\{\xi_{21},\xi_{22},\xi_{23}\}\}$,其中 $\xi_{11}=\{[0,2)\}$, $\xi_{12}=\{[7,9)\}$, $\xi_{13}=\{[12,14)\}$, $\xi_{14}=\{[18,20)\}$, $\xi_{15}=\{[25,27)\}$, $\xi_{21}=\{[2,7)\}$, $\xi_{22}=\{[10,12),[14,17)\}$, $\xi_{23}=\{[20,25)\}$.

性质**1**. 属于调度 ξ 的任何两个不同的任务执行区间$[s_i,e_i)$和$[s_j,e_j)$不相交

证明 该性质的证明分两种情况:同一任务的任务执行区间;不同任务的任务执行区间

当$[s_i,e_i)$和$[s_j,e_j)$属于同一任务时,由定义 2 得:$((s_i<e_i<s_j<e_j)\vee(s_j<e_j<s_i<e_i))$

$\Rightarrow([s_i,e_i)\cap[s_j,e_j)=\varnothing)$.

当$[s_i,e_i)$和$[s_j,e_j)$属于不同任务时:由任何时刻系统中不可能有两个任务同时处于执行状态

$\Rightarrow([s_i,e_i)\cap[s_j,e_j)=\varnothing)$. 证毕

由于 ξ 中的不同的任务执行区间是不会相交的,所以使用符号"<"(早于)和">"(晚于)来标识$[s_i,e_i)$和$[s_j,e_j)$在时间上的先后的关系

性质**2**. ξ_{ij} $(1\leqslant i\leqslant n,1\leqslant j\leqslant k)$的相邻任务执行区间$[s_i,e_i)$与$[s_{i+1},e_{i+1})$之间的时间段$[e_i,s_{i+1})$属于一个或多个任务执行区间.

证明 根据性质 1 的结论:

$([s_i,e_i)\cap[s_{i+1},e_{i+1})=\varnothing)\wedge(e_i<s_{i+1})$

$\Rightarrow[e_i,s_{i+1})\neq\varnothing$.

根据 EDF 调度的性质,$[e_i,s_{i+1})$是任务到来 τ_{ij} 被抢占 CPU 的时间段$\Rightarrow[e_i,s_{i+1})$时间段 CPU 非空闲.

$[e_i,s_{i+1})$时间段 CPU 非空闲$\wedge([e_i,s_{i+1})\neq\varnothing)\Rightarrow[e_i,s_{i+1})$属于一个或多个任务执行区间

证毕

定义**3**. 调度执行区间:属于周期任务集 Π 的任何任务在时刻 S 得到 CPU 进入执行状态,属于 Π 的所有任务在时刻 E 让出 CPU 退出执行状态,若从 S 到 E 的时间段内 Π 中任务从未让出 CPU,则称从 S 到 E 的时间段为 Π 的调度执行区间,记为 $ER=[S,E)$.

定义**4**. 调度空闲区间:属于周期任务集 Π 的所有任务在时刻 S 让出 CPU 退出执行状态,属于 Π 的任何任务在时刻 E 得到 CPU 进入执行状态,若从 S 到 E 的时间段内 Π 中任务从未占用 CPU,则称从 S 到 E 的时间段为 Π 的调度空闲区间,记为 $IR=[S,E)$.

周期任务集单独运行时,调度执行区间 ER 是 CPU 忙的时间段,调度空闲区间 IR 是 CPU 空闲的时间段. 图 1(a)中 ξ 的 ER 和 IR 分别为 $ER=\{[0,9),[10,17),[18,27)\}$, $IR=\{[9,10),[17,18),[27,30)\}$. 调度执行区间与调度空闲区间是交替分布的,且 $ER\cup IR=H$,即调度执行区间与调度空闲区间的并集是超周期.

性质**3**. ξ 的任何任务执行区间属于且只属于一个调度执行区间;ξ 中同属于一个任务到来 τ_{ij} 的任务执行区间只分布在一个调度执行区间中.

证明. 任务执行区间$[s,e)$是执行时间段$\Rightarrow([s,e)\cap IR=\varnothing)$.

$([s,e)\cap IR=\varnothing)\wedge([s,e)$为连续时间段$)\wedge([s,e)\neq\varnothing)\Rightarrow[s,e)$属于且只属于一个调度执行区间

使用反证法证明 ξ 中同属于一个任务到来 τ_{ij} 的任务执行区间只分布在一个调度执行区间中.

假设任务到来 τ_{ij} 的任务执行区间分布在多个调度执行区间中$\Rightarrow\tau_{ij}$ 的起止时间中包含了一段 CPU 空闲时间$\Rightarrow$在 τ_{ij} 还未执行完毕时,CPU 进入了空闲状态. 这与 EDF 调度的性质相矛盾,所以假设不成立. 证毕.

定义**5**. $\xi(t)$:$\xi(t)$是调度 ξ 在时刻 t 以前执行完的任务执行区间的集合.

根据周期任务集的特性有 $\xi(S)=\varnothing$, $\xi(t+H)=\xi(t)$. 例 1 中, $t=14$, $\xi(14)=\{\{\xi_{11}(14),\xi_{12}(14),\xi_{13}(14),\xi_{14}(14),\xi_{15}(14)\},\{\xi_{21}(14),\xi_{22}(14)\}\}=\{\{\{[0,2)\},\{[7,9)\},\{[12,14)\},\varnothing,\varnothing\},\{\{[2,7)\},\{[10,12]\}\}\}$,见图 1(b).

定义**6**. 逆调度 ξ^{-1}:逆调度 ξ^{-1}用于记载周期任务集在一个超周期中的执行情况,ξ^{-1}是相对于 ξ 的,即对于任何属于 ξ^{-1}的任务执行区间$[s,e]$,存在且仅存在一个任务执行区间$[H-e,H-s]$属于 ξ. 定义 $\xi^{-1}=\{\xi_1^{-1},\cdots,\xi_n^{-1}\}$, $\xi_i^{-1}=\{\xi_{i1}^{-1},\cdots,\xi_{ik}^{-1}\}$, $(1\leqslant i\leqslant n,k=H/P_i)$,这里 $\xi_{ij}^{-1}=\{[H-e_p,H-s_p),\cdots,[H-e_1,H-s_1)\}$, $(1\leqslant j\leqslant k)$,其中 $\{[s_1,e_1),\cdots,[s_p,e_p)\}=\xi_{i(k-j+1)}$, $\xi_{i(k-j+1)}$ 属于 ξ. 称 $\xi_{i(k-j+1)}$为 ξ_{ij}^{-1}对应的任务执行区间集.

例 1 中, $\xi^{-1}=\{\{\xi_{11}^{-1},\xi_{12}^{-1},\xi_{13}^{-1},\xi_{14}^{-1},\xi_{15}^{-1}\},\{\xi_{21}^{-1},\xi_{22}^{-1},\xi_{23}^{-1}\}\}$, $\xi_{11}^{-1}=\{[3,5)\}$, $\xi_{12}^{-1}=\{[10,12)\}$, $\xi_{13}^{-1}=\{[16,18)\}$, $\xi_{14}^{-1}=\{[21,23)\}$, $\xi_{15}^{-1}=\{[28,30)\}$, $\xi_{21}^{-1}=\{[5,10)\}$, $\xi_{22}^{-1}=\{[13,16),[18,$

20)}, $\xi_{23}^{-1}=\{[23, 28)\}$, 见图 1(c). 显然, 若 ξ 是可调度的, ξ^{-1}同样是可调度的, 记为 *schedulable*(ξ) ⇒*schedulable*(ξ^{-1}).

我们称 $\xi_{i(k-j+1)}$ 和 ξ_{ij}^{-1} 所表示的任务到来 $\tau_{i(k-j+1)}$和 τ_{ij}^{-1} 为对应的任务到来, 例 1 中 $\xi_{13}=\{[25, 27)\}$和 $\xi_{11}^{-1}=\{[3, 5)\}$表示了两个对应的任务到来 τ_{13} 和 τ_{11}^{-1}. 设 s_1, d_1 和 s_2, d_2 分别为 $\tau_{i(k-j+1)}$和 τ_{ij}^{-1}的到来时间和最后期限, 它们之间的关系为 $s_1=H-d_2$ 和 $s_2=H-d_1$. 将图 1(a)和图 1(c)比较可以看出, ξ^{-1}实际上是将 ξ 中所有任务执行区间相对于其所在超周期的中间点作一个水平旋转得到的. 在 ξ 中任务是尽可能提前执行, 而在 ξ^{-1}中任务是尽可能推迟执行, 依据 ξ^{-1}进行的调度符合 EDL 的性质: $\Omega^{EDL}(0, t)\geqslant\Omega^{Y}(0, t)$[13]. ξ^{-1}的 *ER* 和 *IR* 分别为 $ER=\{[3, 12), [13, 20), [21, 30)\}$, $IR=\{[0, 3), [12, 13), [20, 21)\}$, ξ 和 ξ^{-1}的 *ER* 和 *IR* 是呈对称分布的, 且 $\xi_{i(k-j+1)}$和 ξ_{ij}^{-1}处在对称于超周期中点的调度执行区间中. 属于 ξ的性质 1, 2, 3 同样适用于 ξ^{-1}.

性质 **4**. 属于调度 ξ^{-1}的任何两个不同任务执行区间$[s_i, e_i)$和$[s_j, e_j)$不相交.

性质 **5**. ξ_{ij}^{-1}($1\leqslant i\leqslant n$, $1\leqslant j\leqslant k$)的相邻任务执行区间$[s_i, e_i)$与$[s_{i+1}, e_{i+1})$之间的时间段$[e_i, s_{i+1})$属于一个或多个任务执行区间.

性质 **6**. ξ^{-1}的任何任务执行区间属于且只属于一个调度执行区间; ξ^{-1}中同属于一个任务到来 τ_{ij}的任务执行区间分布在一个调度执行区间中.

定义 **7**. $\xi_{ij}^{-1}-\xi_{ij}(t)$:

记 $\xi_{ij}^{-1}=\{[s_1, e_1), \cdots, [s_k, e_k), [s_{k+1}, e_{k+1}), \cdots, [s_q, e_q)\}$, $\xi_{ij}(t)=\{[s_1', e_1'), \cdots, [s_p', e_p')\}$, $e_p'\leqslant t$;

这里 $e_k\leqslant s_{k+1}$, $len(\xi_{ij}(t))=(e_1'-s_1')+\cdots+(e_p'-s_p')$, 则

$$\xi_{ij}^{-1}-\xi_{ij}(t)=\begin{cases}\varnothing, \text{ 当 } (e_1'-s_1')+\cdots+(e_p'-s_p')=C_i,\\ \{[s_{k+1}, e_{k+1}), \cdots, [s_q, e_q)\},\\ \quad \text{当 } e_1'-s_1')+\cdots+(e_p'-s_p')<C_i\\ \quad \text{且}(e_1-s_1)+\cdots+(e_k-s_k)=len(\xi_{ij}(t)).\end{cases}$$

这里 C_i 为任务的最大执行时间. $\xi_{ij}^{-1}-\xi_{ij}(t)$是从 ξ_{ij}^{-1}中按照时间先后移走等量于任务到来ξ_{ij}在时刻 t 以前已经执行完的时间段. 如例 1 中 $\xi_{11}^{-1}=\{[3, 5)\}$, $\xi_{11}(1)=\{[0, 1)\}$, $\xi_{11}^{-1}-\xi_{11}(1)=\{[4, 5)\}$.

定义 **8**. $\xi^{-1}-\xi(t)$: $\xi^{-1}-\xi(t)=\{\{\cdots, \xi_{1p}^{-1}-\xi_{1p}(t), \cdots\}, \cdots, \{\cdots, \xi_{nq}^{-1}-\xi_{nq}(t), \cdots\}\}$, ($1\leqslant p, q\leqslant k$, $k=H/P$).

例 1 中 $\xi^{-1}-\xi(14)=\{\{\varnothing, \varnothing, \varnothing, \{[21, 23)\}, \{[28, 30)\}\}, \{\varnothing, \{[15, 16), [18, 20)\}, \{[23, 28)\}\}\}$, 如图 1(d). 我们将图 1(a), (c), (d)结合起来可以看出, 图 1(d)是将周期任务集在时刻 14 之前按图 1(a)的排列安排任务执行, 时刻 4 之后按图 1(c)的排列安排任务执行. $\xi^{-1}-\xi(t)$实际上给出了一种调度形式, 这种调度在时刻 t 以前按 EDF 算法进行调度, 时刻 t 之后推迟任务的执行.

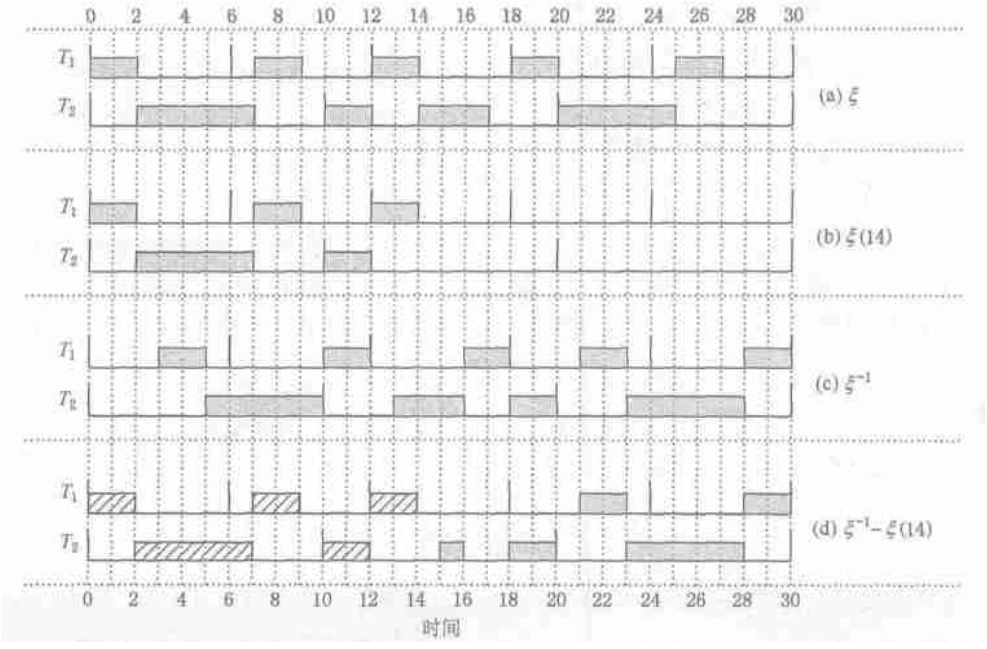

图 1　ξ, $\xi(t)$, ξ^{-1}, $\xi^{-1}-\xi(t)$举例

2.3 周期任务集的最大可挪用时间

定理 **1**. ξ 的任何调度执行区间不能前移，对应 ξ^{-1} 的任何调度执行区间不能后移.

证明 先证明 ξ 的任何调度执行区间 ER（开始时间为 S）不能前移，记为 $F\text{-}unmovable(ER)$.

由于调度执行区间由任务执行区间组成，所以 ER 不能前移体现为任务执行区间不能前移，只需证明属于 ER 的任务执行区间不早于 S 开始即可. 设 s 为任意属于 ER 的任务到来 τ_{ij} 的到来时间，若 $s \geqslant S$，即所有 τ_{ij} 不早于 S 到来，当然属于 ER 的任务执行区间不早于 S 开始，则 ER 不能前移. 用反证法，设 $s < S \Rightarrow$ 任务到来 τ_{ij} 在调度空闲区间到来：

由于使用 EDF 进行调度 $\Rightarrow$ 除非任务被其他任务抢占，任务到来后即投入运行.

$(s < S) \wedge$ 任务在调度空闲区间到来 $\Rightarrow$ 存在 $[s_i, e_i) \in \tau_{ij}$，使得 $s \leqslant s_i < S \Rightarrow [s_i, e_i)$ 部分属于一个调度空闲区间. 从而与性质 3 矛盾，所以有 $s \geqslant S$，即任意属于 ER 的任务到来 τ_{ij} 的到来时间晚于 S，ER 不能前移，记为 $F\text{-}unmovable(ER)$.

接下来证明 ξ^{-1} 的任何调度执行区间 ER^{-1}（结束时间为 D）不能后移，记为 $B\text{-}unmovable(ER^{-1})$.

由于 ξ 的任务到来 τ_{ij} 的到来时间 $s \geqslant S$，根据 ξ^{-1} 的定义，对应 ξ^{-1} 中 $\tau^{-1}_{i(k-j+1)}$ 的最后期限 $d = H - s$ 不会晚于其所在的调度执行区间 ER 的结束时间 $D = H - S \Rightarrow d \leqslant D$.

τ_{ij} 与 $\tau^{-1}_{i(k-j+1)}$ 一一对应 $\Rightarrow$ 任意属于 ER^{-1}（结束时间为 D）的 τ^{-1}_{ij}（最后期限为 d），有 $d \leqslant D \Rightarrow$ 任意属于 ER^{-1} 的 τ^{-1}_{ij} 将在时刻 D 前结束执行 $\Rightarrow ER^{-1}$ 不能后移，即 $F\text{-}unmovable(ER)$. 证毕.

ξ^{-1} 的任何调度执行区间不能后移，即 ξ^{-1} 的调度空闲区间的位置是所有调度 X 中最靠前的，这符合 EDL 调度的性质：$\Omega^{EDL}(0, t) \geqslant \Omega^{X}(0, t)$[13].

定理 **2**. τ_{ij} 和 τ_{pq} 是 ξ 的两个任务到来，τ_{ij} 的最后期限早于 τ_{pq} 的最后期限，则移走 τ_{pq} 的任务执行区间后，τ_{ij} 的任务执行区间在 ξ 中的位置不会因此而可以前移（左移）；同样，τ^{-1}_{ij} 和 τ^{-1}_{pq} 是 ξ^{-1} 的两个任务到来，τ^{-1}_{ij} 的到来时间早于 τ^{-1}_{pq} 的到来时间，则移走 τ^{-1}_{ij} 的任务执行区间后，τ^{-1}_{pq} 的任务执行区间在 ξ^{-1} 中的位置不会因此而可以后移（右移）.

证明 先证明移走 τ_{pq} 的任务执行区间后，τ_{ij} 的任务执行区间不会因此而可以前移. 记 τ_{ij} 和 τ_{pq} 的最后期限分别为 d_i，d_p.

由于 $d_i < d_p \Rightarrow$ 使用 EDF 调度时 τ_{ij} 的优先级比 τ_{pq} 高 $\Rightarrow \tau_{ij}$ 的执行不会被 τ_{pq} 推迟 $\Rightarrow$ 移走 τ_{pq} 的任务执行区间不影响 τ_{ij} 的任务执行区间的位置 $\Rightarrow \tau_{ij}$ 的任务执行区间不会因此而可以前移.

接下来证明移走 τ^{-1}_{ij} 的任务执行区间后，τ^{-1}_{pq} 的任务执行区间不会因此而可以后移.

记 τ^{-1}_{ij} 和 τ^{-1}_{pq} 的到来时间分别为 s_i，s_p；$\tau_{i(k-j+1)}$ 和 $\tau_{p(k-q+1)}$ 为 τ^{-1}_{ij} 和 τ^{-1}_{pq} 在 ξ^{-1} 中对应任务到来，它们的最后期限分别为 d_i，d_p.

由于 $s_i < s_p \Rightarrow d_i = H - s_i > d_p = H - s_p \Rightarrow$ 移走 $\tau_{i(k-j+1)}$ 的任务执行区间后，$\tau_{p(k-q+1)}$ 的任务执行区间不会因此而可以前移 $\Rightarrow$ 移走 τ^{-1}_{ij} 的任务执行区间后，τ^{-1}_{pq} 的任务执行区间不会因此而可以后移.

证毕.

根据定理 1，ξ^{-1} 中所有任务执行区间是不可后移的，但将任务执行区间从 ξ^{-1} 移到 $\xi(t)$ 得到 $\xi^{-1} - \xi(t)$ 后，$\xi^{-1} - \xi(t)$ 的所有任务执行区间却不再是不可后移的，即 $\xi^{-1} - \xi(t)$ 不再具有 EDL 的性质. 考察将 ξ^{-1} 中属于某任务到来 τ_{ij} 的一段任务执行区间 $[a, b)$ 从 ξ^{-1} 移到 $\xi(t)$ 的过程，$\tau_{ij} = (s_1, c_1, d_1, p_1)$ 与其他还未从 ξ^{-1} 完全移到 $\xi(t)$ 的任务到来 $\tau_{pq} = (s_2, c_2, d_2, p_2)$ 之间的关系可以有下面 4 种情况：① $s_1 < s_2$，$d_1 < d_2$；② $s_1 > s_2$，$d_1 < d_2$；③ $s_1 < s_2$，$d_1 > d_2$；④ $s_1 > s_2$，$d_1 > d_2$. 这里不考虑到来时间或最后期限相等的情况，它们可归为这 4 种情况之一. 图 2 是这 4 种情况的直观表示，其中 $[a, b)$ 段是要从 ξ^{-1} 中移走的部分.

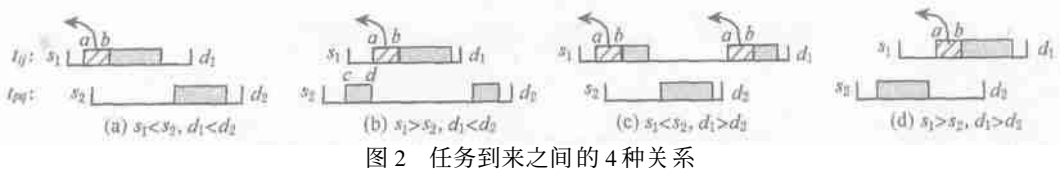

图 2 任务到来之间的 4 种关系

我们逐个分析移走 $[a, b)$ 对 τ_{pq} 剩余任务执行区间的影响：

上面第①，③两种情况是 τ_{ij} 的到来时间早于 τ_{pq} 的到来时间. 根据定理 2，移走 $[a, b)$ 后 τ_{pq} 的任务执行区间不会因此而可以后移. 第④种情况不需考虑，这是因为 τ_{pq} 的到来时间和最后期限都早于

τ_{ij}, τ_{pq}将先于 τ_{ij} 执行, 所以待到 τ_{ij}开始执行时, τ_{pq} 的任务执行区间已全部移到了$\xi(t)$中. 所以情况①③④中移走$[a, b)$后, τ_{pq}的剩余任务执行区间不会因此而可以后移. 第②种情况较复杂, 在 ξ^{-1}中 τ_{ij} 和τ_{pq}的执行过程有可能出现如图 2(b)的情况, 移走 $[a, b)$部分后, ξ^{-1}中 τ_{pq}的剩余部分$[c, d)$可以向后移动. 其解决办法是将$[c, d)$整体向后移动$(b-a)$的距离来填充原有的$[a, b)$段, 这样 τ_{pq}的剩余部分同样是不能后移的.

定义 **9**. $\xi^{-1}(t)$: $\xi^{-1}(t)$与 $\xi^{-1}-\xi(t)$类似, 是从 ξ^{-1}中移走等量于在时刻 t 以前已经执行完的任务执行区间, 其不同点在于从 ξ^{-1}中移出时间段的顺序. $\xi^{-1}(t)$遵循下面的规则:

设$[t, t+\varepsilon)(\varepsilon>0, \varepsilon\to 0)$为任务到来 τ_{ij}将要执行的时间段, $[t^{-1}, t^{-1}+\varepsilon)$为按照 $\xi^{-1}-\xi(t)$定义的方法将从 ξ^{-1}移出的时间段, $[t^{-1}, t^{-1}+\varepsilon)\in\tau_{ij}^{-1}$, $[t^{-1}, t^{-1}+\varepsilon)$从 ξ^{-1}移到$\xi(t)$中后成为$[t, t+\varepsilon)$. 由于 ε 趋于 0, 我们可以假设$[t, t+\varepsilon)$时间段内任务不会发生切换. τ_{pq}^{-1}代表任何还未完全从 ξ^{-1}移到$\xi(t)$的任务到来, $[t^{-1}, t^{-1}+\varepsilon)$按照下面的方法从 ξ^{-1}移出:

① 若 τ_{ij}^{-1}的到来时间早于所有的 τ_{pq}^{-1}, 则直接从 ξ^{-1}中移出$[t^{-1}, t^{-1}+\varepsilon)$即可;

② 若存在到来时间早于 τ_{ij}^{-1}到来时间的 τ_{pq}^{-1}, 则在移走 τ_{ij}^{-1}的时间段$[t^{-1}, t^{-1}+\varepsilon)$后, 对 ξ^{-1}的剩余部分进行调整: 将 τ_{pq}^{-1}的任务执行区间中结束时间早于时刻 t^{-1}的任务执行区间向后移动 ε 的距离.

定理 **3**. $(\xi^{-1}(t), (0\leqslant t\leqslant H))$的所有调度执行区间不能后移.

证明. 使用归纳法: 先证时刻 $t=0$, $\xi^{-1}(t)$的所有调度执行区间不能后移; 然后假设时刻 t, $\xi^{-1}(t)$的所有调度执行区间不能后移, 证明时刻 $t+\varepsilon(\varepsilon>0, \varepsilon\to 0)$, $\xi^{-1}(t+\varepsilon)$的所有调度执行区间不能后移; 由于超周期的长度有限, 所以定理得证.

当 $t=0$ 时, $\xi^{-1}(0)=\xi^{-1}$, 由定理 1 推出 ξ^{-1}的所有调度执行区间不能后移.

假设时刻 t, $\xi^{-1}(t)$的所有调度执行区间不能后移, 下面证明 $\xi^{-1}(t+\varepsilon)$, $(\varepsilon>0, \varepsilon\to 0)$的所有调度执行区间不能后移.

首先考察任务在 t 到 $t+\varepsilon$ 时间段内没有发生切换的情况.

若在 t 到 $t+\varepsilon$ 时间段内 CPU 空闲

$\Rightarrow(\xi^{-1}(t+\varepsilon)=\xi^{-1}(t))$

$\Rightarrow\xi^{-1}(t+\varepsilon)$的所有调度执行区间不能后移.

若在 t 到$t+\varepsilon$ 时间段内 CPU 处于执行状态, 根据 $\xi^{-1}(t)$的定义和前面分析的 4 种情况 $\Rightarrow$在将 ε 长度的时间段挪出 ξ^{-1}之后, $\xi^{-1}(t)$的任何调度执行区间是不能后移的.

然后考察任务在 t 到$t+\varepsilon$ 时间段内发生了切换的情况: 设任务切换的时刻为 $t_1, \cdots, t_i, \cdots$, 则可以将这个过程按这些时间点分为多个时间段. 根据上面的分析, 在每个时间段结尾 $\xi^{-1}(t_i)$的所有调度执行区间不能后移, 因此 $\xi^{-1}(t)$的所有调度执行区间在 t 到$t+\varepsilon$ 时间段内不能后移.

由 $t=0$ 时 $\xi^{-1}(t)$满足性质, 并且若时刻 t, $\xi^{-1}(t)$的所有调度执行区间不能后移, 则时刻 $t+\varepsilon$ $(\varepsilon>0)$, $\xi^{-1}(t+\varepsilon)$的所有调度执行区间不能后移, 且超周期的长度是有限的, 推出 $\xi^{-1}(t)(0\leqslant t\leqslant H)$的所有调度执行区间不能后移. 证毕.

定理 **4**. 记 T 为一个超周期中周期任务集被挪用的时间与 CPU 的空闲时间和, S 为$\xi^{-1}(t)$的第 1 个调度执行区间的开始时间, 则$(S-t-T)$为周期任务集在时刻 t 的最大可挪用时间.

证明. 由定理 3 $\Rightarrow\xi^{-1}(t)$中的任何调度执行区间不能后移.

(S 为$\xi^{-1}(t)$的第 1 个调度执行区间的开始时间)$\wedge$($\xi^{-1}(t)$的调度执行区间不能后移)

$\Rightarrow$在时刻 t, S 的值是不能再增大的

$\Rightarrow(S-t-T)$为任务集在 t 可挪用最大时间.

证毕.

需要注意的是, $(S-t-T)$的值可以为 0, 周期任务集的执行时间不能被挪用. 在这种情况下, 需要等到对应的调度执行区间结束再重新计算$(S-t-T)$的值.

根据定理 4 的结论可以设计相应的算法, 但直接使用该结论会造成大量的计算, 所以需要进行一些简化. 将一个超周期中所有任务到来的到来时间点用向量 $\alpha=\{s_1, s_2, \cdots, s_i, s_{i+1}, \cdots, s_N\}$ 表示, 其中 $s_i<s_{i+1}$且 $s_1=0$, $s_N=H$[13], N 为周期任务在一个超周期中的到来次数之和. 根据定义 9 计算得到每个 s_i 对应的$\xi^{-1}(s_i)$, 记 S_i 为$\xi^{-1}(s_i)$的第 1 个调度执行区间的开始时间, 则将相应 S_i 组成向量 $\beta=\{S_1, \cdots, S_N\}$, 周期任务的完成量 $len(\xi^{-1}(s_i))$(见定义 7)组成向量 $\gamma=\{len(\xi^{-1}(s_1)), \cdots, len(\xi^{-1}(s_N))\}$. 得到 β 和 γ 之后, 我们可以计算任意时刻 t

对应的 $\xi^{-1}(t)$的 S:假设 $len(\xi^{-1}(s_i))\leqslant len(\xi^{-1}(t))<len(\xi^{-1}(s_{i+1}))$, 这里只需将 $\xi^{-1}(s_i)$的 $len(\xi^{-1}(t))-len(\xi^{-1}(s_i))$长度的任务执行区间按照 $\xi^{-1}(t)$的定义从 $\xi^{-1}(s_i)$中移出即可得到 S.

3 ISA 算法

3.1 算法描述

ISA 算法的思路是: 让周期任务集先运行一个 H 的时间, 得出 ξ, ξ^{-1}, β, γ 的值, $\xi^{-1}(t)$的值在运行过程中计算. ISA 调度算法描述如下:

(1) 在第 1 个 H 中使用 EDF 调度周期任务集运行, 纪录 ξ 的值. 若有非周期任务到来, 将其放在后台运行, 使用周期任务集的空闲时间执行非周期任务.

(2) 在记录 ξ 的同时, 根据定义 6 计算 ξ^{-1} 的值.

(3) 利用第 1 个 H 结束前的 CPU 空闲时间计算 β, 由此得到 γ. 若 $U_{\Pi}<1$ 且无非周期任务占用的情况下, 这个空闲区间总存在[14], 否则只能占用其他空闲时间.

(4) 在第 2 个 H 中使用 EDF 进行调度, 累计 CPU 空闲时间到 T 中, 累计周期任务执行时间到 F 中.

(5) 若非周期任务在时刻 t 到来, 根据 F, β, γ 计算 $\xi^{-1}(t)$, 得到 S.

(6) 计算$(S-t-T)$, 若$(S-t-T)>0$, 则挪用$(S-t-T)$长度的时间段, 若$(S-t-T)=0$, 则等到对应的调度执行区间结束, 转步(5), 重新计算$(S-t-T)$的值.

(7) 挪用$(S-t-T)$时间段后, 将$(S-t-T)$累加到 T 中, 继续使用 EDF 进行调度, 累计 CPU 空闲时间到 T 中, 周期任务执行时间到 F 中.

(8) 新的超周期到来, 置 $T=0$, $F=0$, 转第(4)步.

ISA 算法中, ξ 在运行过程中记录, 在记录 ξ 的过程中计算 ξ^{-1}, 只需 2 次减法计算, 所以相应的计算量很小, 占用 CPU 资源也很小, 只是需较多的存储空间. β, γ 的计算复杂度为$O(N)$, 但 β, γ 只需计算一次, 在以后的超周期中都可使用. 根据 β 计算 $\xi^{-1}(t)$的复杂度为 $O(n)$(n 为周期任务数), 即周期任务在$[s_i, t]$ 时间段内切换的次数. 比较 EDL 在每次非周期任务到来时需进行复杂度为 $O(N)$ 的计算[11], ISA 算法大大简化了计算复杂度.

3.2 ISA 算法的可行性和最优性

首先证明周期任务集在挪出$(S-t-T)$时间段后仍是 EDF 可调度的, 记为 $schedulable_{EDF}()$.

定理 5. 逆调度 ξ^{-1} 在其调度执行区间集 ER 中的调度过程可转化为使用 EDF 算法进行调度的过程, 记为 ξ^{-1}_{EDF}.

证明. 在文献[14]中已经证明: 任何可调度的任务集都是 EDF 可调度的.

由 $schedulable(\xi^{-1})$

$\Rightarrow\xi^{-1}$在其调度执行区间集 ER 中的调度过程可转化为使用 EDF 进行调度的过程. 证毕.

定理 6. $\xi^{-1}(t)$, $(0\leqslant t\leqslant H)$在其调度执行区间集 ER 中的调度过程可转化为使用 EDF 算法进行调度的过程, 记为$(\xi^{-1}(t))_{EDF}$.

证明 同上, $schedulable(\xi^{-1}(t))$

$\Rightarrow\xi^{-1}(t)$在其调度执行区间集 ER 中的调度过程可转化为使用 EDF 进行调度的过程. 证毕.

综上, 在 ISA 算法的第(7)步仍然可以使用 EDF 进行调度.

定理 7. ISA 算法能够取得非周期任务的最短响应时间和最短周转时间.

证明. 根据定理 4, $(S-t)$是使用 EDF 调度时的最大可挪用时间, T 是同一超周期中已经挪用的时间与 CPU 空闲时间之和, 所以$(S-t-T)$是时刻 t 能分配给非周期任务的最大时间. ISA 算法的第(4)步中将$(S-t-T)$分配给非周期任务, 所以取得了最短响应时间, 并且非周期任务周转时间也是最短的. 证毕.

4 仿真实验

文献[7, 9, 10]中已经验证了挪用法优于带宽保留法和后台执行法, 且文献[10, 11]的实验结果表明 EDL 算法优于 IPE 等基于动态优先级和带宽保留的算法, 所以, 这里我们将仿真实验中 ISA 算法和挪用法中的 EDL 算法以及后台执行法比较的结果列出. 实验中, 我们对系统处于不同负载时非周期任务集的运行情况进行比较. 系统负载 $U=U_{\Pi}+U_{\Gamma}$, 其中 U_{Π} 为周期任务集的负载, U_{Γ}为非周期任务集的负载, U_{Π} 必须小于等于 1, 否则会引起任务集的不可调度性. 我们选择周期任务集负载$U_{\Pi}=40\%, 70\%, 90\%$的 3 种情况进行比较, 这样可以看出周期任务集负载变化对各种算法调度性能的影响. 周期任务集由周期随机分布在 100~500 μs 范围

内的 10 个周期任务组成, 它们的周期和执行时间如表 1 所示:

表 1　周期任务集

序号	周期	执行时间/μs		
		40%	70%	90%
1	110	50	80	90
2	160	50	80	120
3	230	100	180	180
4	270	100	180	230
5	310	150	240	280
6	380	150	240	340
7	420	180	300	370
8	460	190	300	450
9	510	200	400	490
10	590	200	400	550

非周期任务到来时间的取值是随机的, 由一个随机生成器生成, 这样符合非周期任务的特性. 非周期任务在一段时间内的平均到来时间和执行时间由 U_{Γ} 决定, 这里通过 U_{Γ} 控制非周期任务的到来. 根据周期任务集周期的分布, 我们取非周期任务平均到来时间分别为 50, 150, 200 三种情况进行比较. 这样的取法可以比较出算法在非周期任务平均到来时间不同时的效率. 实验结果取连续 100 个超周期中非周期任务响应时间的平均值.

图 3, 4, 5 为周期任务集负载分别等于 40%, 70%, 90%的比较结果. 其中每幅图包括 3 幅小图, 分别为非周期任务平均到来时间等于 50, 150, 200 的比较结果. 从比较结果可以看到 ISA 算法在系统负载较小的情况下可以获得很好的响应时间. 尤其是在周期任务负载较小的情况下, 影响 ISA 算法响应时间的主要因素是非周期任务的负载. 在系统负载较大的情况下, ISA 算法的响应时间相对较大, 这主要是由于 ISA 算法需要一定的计算时间, 但只要系统有空闲 CPU 时间, ISA 算法总是能够取得很好的响应时间, 这一点可以从图 5 看出. 在系统负载较小的情况下, ISA 与 EDL 的效率是相似的; 系统负载较大时, ISA 体现了它的优越性, 这是由于 ISA 计算最大可挪用时间的方法较 EDL 的简单, 使用 CPU 资源较少.

相对来说, ISA 使用的空间较多, ISA 使用的内存空间最大为 n^2 个整型(n 为周期任务数), 计算可知, 在具有 100 个周期任务的系统中, 这一空间大小不超过 40KB. 在现代的硬件环境下, 这一空间消耗相对算法实现的效率来说是可取的.

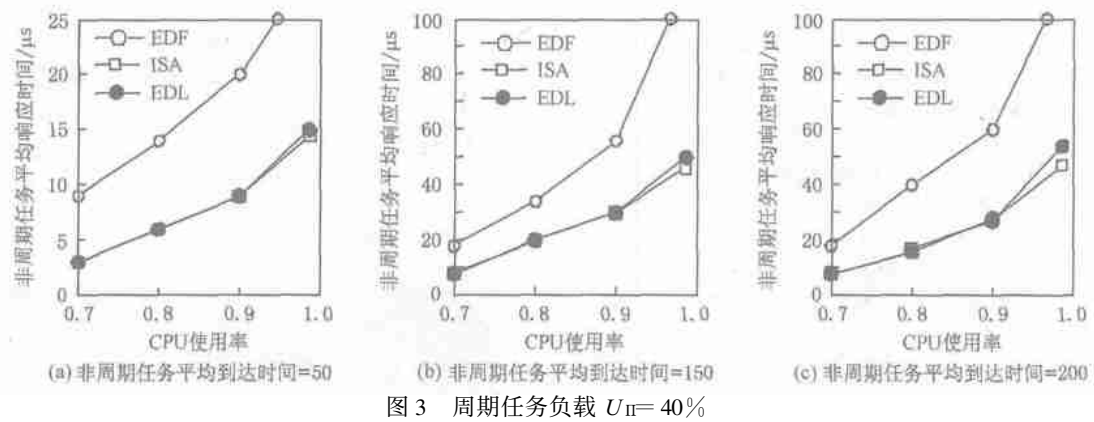

图 3　周期任务负载 U_{Π}= 40%

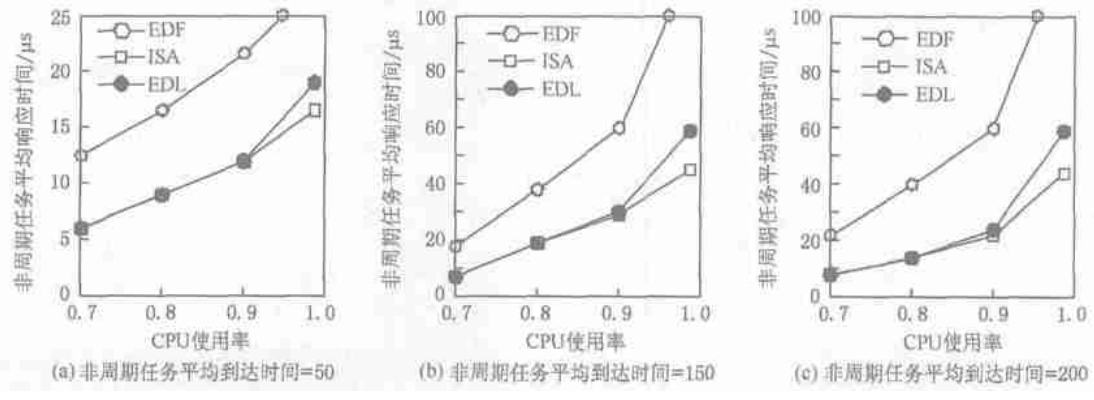

图 4　周期任务负载 U_{Π}= 70%

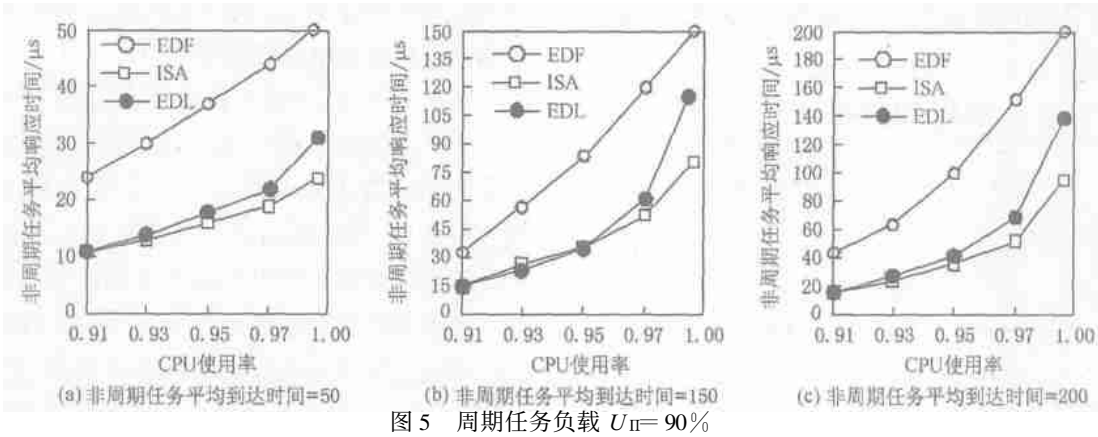

图 5 周期任务负载 $U_{\Pi}=90\%$

5 结 论

本文定义了调度"与"逆调度",然后分析了周期任务集在使用 EDF 调度时的最大可挪用时间和"调度"、"逆调度"之间的关系,在此基础上得出了在保证硬周期任务最后期限的情况下求解周期任务集最大可挪用时间的精确算法,简化了原有算法的计算复杂度 证明 ISA 算法求解的周期任务集的可挪用时间为最大可挪用时间并且使用仿真实验对软非周期任务的响应时间进行了验证.

参 考 文 献

1 J P Lehoczky, L Sha, J K Strosnider. Enhanced aperiodic responsiveness in hard real-time environments. In: Proc of the Real-Time Systems Symposium. San Jose, Califomia, USA: IEEE Computer Society, 1987. 261~270

2 B Sprunt, L Sha, J Lehocsky. Aperiodic task scheduling for hard-real-time systems. Journal of Real-Time Systems, 1989, 1(1): 27~60

3 Guillem Bernat, Alan Burns. Multiple servers and capacity sharing for implementing flexible scheduling. Real-Time Systems, 2002, 22(1/2): 49~75

4 B Sprunt, J Lehoczky, L Sha. Exploiting unused periodic time for aperiodic service using the extended priority exchange algorithm. In: Proc of the 9th Real-Time Systems Symposium. Huntsville, Alabama, USA: IEEE Computer Society, 1988. 251~258

5 J Lehoczky, L Sha, Y Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Proc of the 10th Real Time Systems Symposium. Santa Monica, California, USA: IEEE Computer Society, 1989. 166~171

6 T S Tia, J W S Liu, M Shankar. Algorithms and optimality of scheduling aperiodic requests in fixed-priority preemptive systems. Journal of Real-Time Systems, 1995, 10(1): 23~43

7 何军, 孙玉方. 提高软非周期任务相应性能的调度算法. 软件学报, 1998, 9(10): 721~727
(He Jun, Sun Yufang. An algorithm to improve the responsive performance for scheduling soft-aperiodic tasks. Journal of Software(in Chinese), 1998, 3(10): 721~727)

8 G C Buttazzo, F Sensini. Optimal deadline assignment for scheduling soft aperiodic tasks in hard real-time environments. IEEE Trans on Computers, 1999, 48(10): 1035~1052

9 I Ripoll, A Crespo, A Garcia-Fornes. An optimal algorithm for scheduling soft aperiodic tasks in dynamic-priority preemptive systems. IEEE Trans on Software Engineering(in Chinese), 1997, 23(6): 388~400

10 M Spuri, G C Buttazzo. Efficient aperiodic service under earliest deadline scheduling. In: Proc of the 15th Real-Time Systems Symposium. San Juan, Puerto Rico: IEEE Computer Society, 1994. 2~11

11 M Spuri, G C Buttazzo. Scheduling aperiodic tasks in dynamic priority systems. Journal of Real-Time Systems, 1996, 10(2): 179~210

12 陈劲林, 杨士元, 胡东成. 基于确定周期性任务的进程管理及可调度性分析. 计算机研究与发展, 2000, 37(3): 307~312
(Chen Jinlin, Yang Shiyuan, Hu Dongcheng. Study of determinate periodic task based process management and it's schedulability. Journal of Computer Research and Development (in Chinese), 2000, 37(3): 307~312)

13 H Chetto, M Chetto. Some results of the earliest deadline scheduling algorithm. IEEE Trans on Software Engineering, 1989, 15(10): 1261~1269

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

涂 刚 男,1976 年生,博士研究生,主要研究方向为实时调度算法、实时嵌入式操作系统

阳富民 男,1966 年生,教授,主要研究方向为数据库、实时嵌入式操作系统

卢炎生 男,1949 年生,教授,博士生导师,主要研究方向为数据库与信息系统、实时数据库