第35卷 第19期 计 算 机 工 程 2009年10月
Vol.35 No.19 Computer Engineering October 2009
·网络与通信· 文章编号：1000—3428(2009)19—0114—03 文献标识码：A 中图分类号：N945.15

WSN软实时系统的DVS控制优化算法

陈 坚[1]，邹 涛[2]，梁根池[1]
(1. 武警工程学院研究生大队，西安 710086；2. 武警工程学院通信工程系，西安 710086)

摘 要：为降低无线传感器网络(WSN)软实时系统中的能量消耗，建立能量消耗数学模型，引入离散事件系统框架中的优化控制问题使目标函数最小化，采用线性规划方法求解，得到一种可扩展的低复杂度算法，并对该算法进行优化。数值结果表明，应用优化算法对WSN节点进行动态电压调节，能在满足时限要求的基础上更大程度地节省节点能量。

关键词：无线传感器网络；软实时系统；动态电压调节；线性规划

Optimal Algorithm for DVS Control in Soft Real-time System of WSN

CHEN Jian[1], ZOU Tao[2], LIANG Gen-chi[1]
(1. Graduate Student Team, Engineering College of Armed Police Force, Xi'an 710086;
2. Communication Engineering Department, Engineering College of Armed Police Force, Xi'an 710086)

【Abstract】In order to reduce energy consumption of soft real-time system in WSN, this paper establishes mathematical model of energy consumption, introduces an optimal control problem in the Discrete Event System(DES) framework to minimize the objective function, resorts to linear programming method for solving the problem, obtains a scalable algorithm of low complexity, and optimizes the algorithm. Numerical results show that the performance of optimal algorithm is improved greatly, Dynamic Voltage Scaling(DVS) controls with the optimal algorithm in WSN nodes can save much power while meeting the requirements of deadline.

【Key words】Wireless Sensor Network(WSN); soft real-time system; Dynamic Voltage Scaling(DVS); linear programming

无线传感器网络(WSN)在诸多领域，如工业控制、家庭自动化、消费电子产品、安全和军事侦察、资产跟踪、智能农业、生态环境研究与健康监测等方面有广泛的应用。很多对实时性要求并不高但是对能量消耗要求苛刻的应用随之产生，对WSN节点中软实时系统的能耗节省研究成为一个新课题。一种备受关注的方法是动态电压调节(Dynamic Voltage Scaling, DVS)技术，其主要思想是：在满足任务实限要求的前提下，动态调整处理器的运行频率，使处理器的工作电压不必总是处于最大值从而节省电能[1]。

1 优化控制的数学模型

1.1 离散时间系统框架中的优化控制

离散事件系统(Discrete Event System, DES)框架中的优化控制问题原理来自于著名的极大代数方程[2]：

$$x_i = \max\{d_i,\ x_{i-1}\} + s_i(u_i),\ i = 1,\ 2,\cdots,\ K \tag{1}$$

其中，x_i是任务i的完成时间；s_i是任务i的服务时间；u_i是任务i的处理时间。系统包括一个确定的单队列服务，它要处理给定顺序的K个任务。

在软实时系统中，引入一个函数$\varphi_i(x_i)$，称为任务i的罚函数，它根据任务的时限要求对任务完成时间进行处理，则目标函数J定义如下：

$$J = \sum_{i=1}^{K}\left(\theta_i(u_i) + \beta\varphi_i(x_i)\right) \tag{2}$$

其中，u_i是任务i的处理时间；$\theta_i(u_i)$为任务i的代价函数，是正的、连续可微的，且在u_i上单调递增的严格凸函数；β为正实数。此时优化控制问题就成为给定约束条件使函数J最小化的问题。单队列综合系统模型[3]如图1所示。

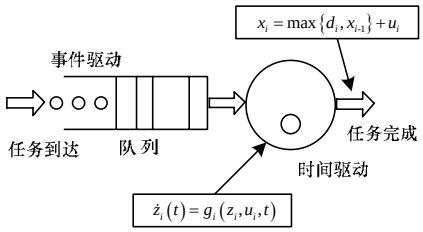


图1 单队列综合系统模型

其中，每个任务都有自己所处的状态，由定义式$\dot{z}_i(t) = g_i(z_i,\ u_i,\ t)$知其状态随时间而不断变化，$x_0$时刻是初始状态，由$\xi = z_i(x_0)$定义。在该模型中，$u_i$的取值将同时影响$z_i$和$x_i$，优化控制的目标就是寻找一个控制序列$\{u_i,\ u_{i+1},\cdots,u_k\}$来使式(2)中的目标函数$J$最小化。这个问题中没有状态变量和基于排队系统的确定性DES模型。

1.2 优化模型

代价函数$\theta_i(u_i)$表征能量的消耗量，是任务处理时间u_i的函数，处理时间越长，代价越大。为了达到更高的正确性，为目标函数式添加了罚函数$\varphi_i(x_i)$作为补充项，$\varphi_i(x_i)$是任务完成时间x_i的函数，对每个任务i来说$\varphi_i(x_i)$都不同。目标函数J还含有一个正实数β，它是平衡能量消耗和时延的权重因子。另外，若任务i有N_i个操作，则处理时间u_i有一个下

作者简介：陈 坚(1983－)，男，硕士研究生，主研方向：无线数据通信；邹 涛，副教授；梁根池，硕士研究生

收稿日期：2009-01-22 **E-mail：**chen-jian.net@163.com

界 γN_i [4]，其中 γ >0，是个与设备相关的参数。

优化控制问题 P 的数学模型为

$$P: \quad \min_{u_1,u_2,\cdots,u_K} \sum_{i=1}^{K}\left[\theta_i(u_i)+\beta\varphi_i(x_i)\right]$$

$$\text{s.t.} \quad x_i=\max\{x_{i-1},a_i\}+u_i$$

$$u_i \geq \gamma N_i, \quad i=1,\ 2,\cdots,K$$

这个优化控制问题的数学模型函数一般为非凸函数，如果设计变量 K 取得非常大，其求解将非常困难。为克服这个难题不少学者采取分解的方法，尝试构造不同的代价函数来解决问题。本文将在已有结论的基础上进行扩展讨论。

1.3 代价函数与罚函数

代价函数满足如下假设：

假设 1 对于所有的 $i=1,\ 2,\cdots,K$，$\theta_i(u_i)$ 是正的、连续可微的，且在 u_i 上单调递减的凸函数。

在 WSN 节点中，代价函数的形式如下[4]：

$$\theta_i(u_i)=C_1N_i\left(\frac{V_t u_i}{u_i-N_iC_2}\right)^2$$

其中，C_1, C_2 为系统参数；V_t 为处理器阈值电压；N_i 表示任务 i 有 N_i 个操作。

引入罚函数 $\varphi_i(x_i)$ 的目的是为满足软实时系统的 QoS 要求。可以选择已知函数来模拟罚函数 $\varphi_i(x_i)$ 对超时限的处理。一般选择满足如下假设的函数来充当罚函数：

假设 2 $\varphi_i(\cdot)$ 连续可微，严格凸且其最小值可以在有限点处取得[5]。

满足这个假设的一个例子是二次函数 $\varphi(x)=(x-d)^2$，引入二次函数：

$$\varphi_i(x_i)=(x_i-d_i)^2 \tag{3}$$

作为罚函数，它可以严格地“惩罚”任务违背时限要求，如图 2 所示，当任务在其最后期限点处完成时，$x_i=d_i$，则 $\varphi_i(x_i)$ 得到最小值。但图 3 所示曲线只是一种理想情况，实际应用中应当根据具体情况作适当的修正。

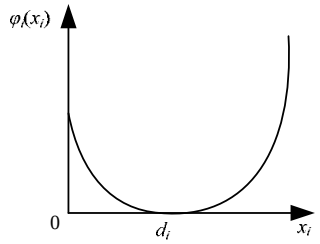


图 2 二次罚函数

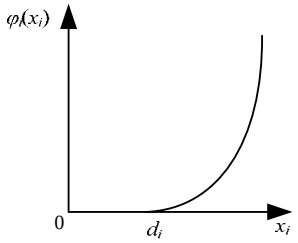


图 3 贝塞尔函数

在许多实际应用(如制造系统)中，人们通常只关心任务完成的延迟量，而不关心提前量。针对该问题，可选择其他合适的函数来充当罚函数得到延迟量，同时应避免函数在时间点 d_i 处不可微，图 3 所示的贝塞尔函数就是个很好的例子。其表达式为

$$\varphi_i(x_i)=\begin{cases}0 & x_i<d_i\\ \xi\left[a+b(x_i-d_i-\xi)\right]^2 & x_i\geq d_i\end{cases}$$

其中，ξ 为正数；$a,\ b$ 是适当的实数。不过它不满足假设 2，因此，须另外寻找方法求解。

2 优化控制算法

2.1 二次罚函数的前向算法

函数 $\varphi_i(x)=(x-d_i)^2$ 满足假设 2，则优化问题 Q 的数学模型为

$$Q(k,n): \quad \min_{u_k,\cdots,u_n} \sum_{i=k}^{n}\left[\theta_i(u_i)+\beta\varphi_i(x_i)\right]$$

$$\text{s.t.} \quad u_i\geq\gamma N_i, \quad i=k,\cdots,n$$

$$x_i=a_k+\sum_{j=k}^{i}u_j, i=k,\cdots,n-1$$

给定了任务到达时间，优化问题 P 可通过前向算法(FA)来求解，该算法如下：

Step 1 k:=1, $a_{K+1}=\infty$; //初始化
While n≤K

Step 2 Do
{

Step 3 //确定独立忙期
Solve problem Q(k, n)
If $x_n^*(k,n)<a_{n+1}$ then
$u_j^*=u_j^*(k,n)$ for j=k,⋯,n;
k:=n+1;
end if

Step 4 n:=n+1 //将 n 加 1
}
end while

前向，是指最优样本轨迹的构建由任务 1 开始，并及时向前进行而无需多重的向前向后扫描。令 $k=1$，$n=1$ 首先解决线性约束优化问题 $Q(k,n)$，同时得到控制序列 $u_j^*(k,n)$ 和任务完成时间 $x_n^*(k,n)$，$j=k,\cdots,n$。可以由式 $x_n^*(k,n)<a_{n+1}$ 是否成立来判断是否为忙期。如果确定任务 $k,\cdots,n$ 处于独立的忙期，对于 $k,\cdots,n$，最优样本轨迹的控制由 $u_j^*(k,n)$ 决定。然后设定新忙期第 1 个任务 $k=n+1$。所有任务重复相同的过程。此算法要执行 K 步，解决 K 个子问题而获得最优解。

2.2 改进的前向算法

引入以如下形式表征的 $\varphi_i(x_i)$ 函数:

$$\varphi_i(x_i)=\begin{cases}0 & x_i<d_i\\ \xi\left[a+b(x_i-d_i-\xi)\right]^2 & x_i\geq d_i\end{cases}$$

其中，ξ 为正数；$a,\ b$ 是适当的实数。它不满足假设 2，因此，需要另外寻找方法求解。若引入另一个假设：

假设 3 $\varphi_i(\cdot)$ 是连续可微的严格凸函数，且在 i=1, 2,⋯, K-1 时非递减，其他情况下单调递增。

证明以下定理[4]：

定理 如果对于问题 $Q(1,\ k)$，任务 i 不是关键任务，即 $x_i^*(1,K)>a_{i+1}$，那么对于问题 P，任务 i 不可能是任何忙期的最后一个任务。

改进的前向算法(IFA)如下：

Step 1 //初始化
Solve sub-problem Q(1, K)
For j=1, 2,⋯, K
If $x_j^*(1,K)=a_{j+1,}$ add j to set {Critical}
Endfor
//控制循环跳出
add K+1 to set {Critical}
k:=1, i:=1; n:=Critical(i), $a_{K+1}=\infty$;

```
While   n≤K
 Do
 {
Step 2    Solve problem Q(k,n)
Step 3    //确定独立忙期
 If  x*n (k, n)<a_{n+1} then
    u*j=u*j(k,n)   for j=k,…,n;
     k:=n+1;
 end if
Step 4     i:=i+1        //将 n 加 1
     n:=Critical(i)
 }
end while
```

其中，集合{*Critical*}是所有关键任务的集合，对比原始算法有以下几个不同：

(1)在初始化阶段，IFA 必需解决问题 $Q(1,\ K)$ 并且取得所有关键任务的信息。

(2)在 Step4 中，IFA 将下一个关键任务前移 1 后，赋值给 n。

(3)IFA 最多进行 $K+1$ 步(通常少于 K 步)，而 FA 总要求进行 K 步。如果所有 K 个任务处于独立的忙期且没有关键任务，那么解 $Q(1,\ K)$ 即可得优化解。

(4)当 K 增大时，$Q(1,\ K)$ 计算量增大。但可通过进一步的变换算法取得 $Q(1,\ K)$ 的初始解[5]。

2.3 权重因子 β 的选取

在软时限要求的实时系统中，希望任务只是在偶然情况下才超过时限。β 值有不同的计算方法，其取值对结果有很大影响，一般可用如式(1)所示的经验公式，各参数的含义同上文。

$$\beta=\max_{all\ i}\left\{\frac{\theta(\gamma N_i)}{(\gamma N_i)^2}\right\} \tag{4}$$

3 数值结果

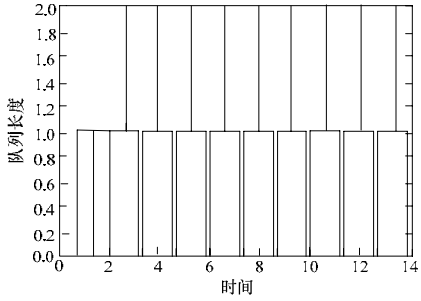
计算以下目标函数来验证算法性能，部分参数表 1。

表 1 部分计算参数

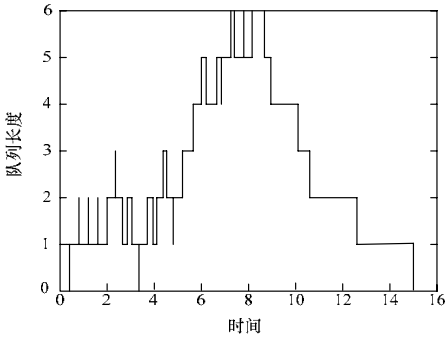
参数名	参数值	说明
V_t	5	阈值电压
N_i	1	任务所含的操作数
β	1, 0.1, 10, 100	权重因子
K	20	任务总量

计算结果表明，当加权因子 β 不变时，任务的不同到达速率对应不同的忙期结构。β=1 时忙期的结构如图 4 所示。其中，图 4(a)为低到达速率，有多重忙期，这表明处理器工作在基本恒定的速率下，在运行速率满足时限要求的基础上没有消耗更多的能量；但当任务到达速率增大时，通常忙期也延长，这说明也会有更长的队列，如图 4(b)所示，系统处理时间也会相应延长，处理器低速运行将会降低到时间的准确性。

表 2 列出了不同 β 值的情况下采用优化的 DVS 控制和未采用 DVS 控制的系统开销，其前后变化非常显著，显然，β 值固定时能量开销也是恒定的，而随着 β 的增大，任务完成时间和任务时限的偏差会加大，导致能耗增加。可见，当 $\beta=10$ 时效果最好。



(a)低到达速率



(b)高到达速率

图 4 β=1 时忙期的结构

表 2 不同 β 值的系统开销

β	代价函数	计算结果 (加入优化控制)	计算结果 (不加入控制)
	$\sum_{i=1}^{K}\theta_i(u_i)$	27.259 500	500.000 000
0.1	$\sum_{i=1}^{K}\varphi_i(u_i)$	71.633 000	1.574 000
	$\sum_{i=1}^{K}(\theta_i(u_i)+\beta\varphi_i(x_i))$	34.422 800	500.157 400
	$\sum_{i=1}^{K}\theta_i(u_i)$	28.881 400	500.000 000
1.0	$\sum_{i=1}^{K}\varphi_i(u_i)$	12.915 200	1.574 800
	$\sum_{i=1}^{K}(\theta_i(u_i)+\beta\varphi_i(x_i))$	41.796 500	501.574 800
	$\sum_{i=1}^{K}\theta_i(u_i)$	47.725 600	500.000 000
10.0	$\sum_{i=1}^{K}\varphi_i(u_i)$	5.419 120	1.574 810
	$\sum_{i=1}^{K}(\theta_i(u_i)+\beta\varphi_i(x_i))$	101.916 800	515.748 100
	$\sum_{i=1}^{K}\theta_i(u_i)$	94.706 000	500.000 000
100.0	$\sum_{i=1}^{K}\varphi_i(u_i)$	2.288 014	1.574 806
	$\sum_{i=1}^{K}(\theta_i(u_i)+\beta\varphi_i(x_i))$	323.507 400	657.486 000

4 结束语

引入恰当的罚函数来对超时限进行处理是算法的关键，也是本文算法取得成功的一个重要原因。改进的前向算法还在于它在初始阶段就取得了所有关键任务的信息，提高了整个算法性能。数值结果表明，WSN 节点采用优化的 DVS 控制能在满足时限要求的基础上更大程度节省节点能量，从而延长网络寿命。

参考文献

[1] 王华勇, 陈 渝, 戴一奇. 适用于不确定环境中的 DVS 软实时调度算法[J]. 计算机工程, 2006, 32(11): 4-6.

[2] Cassandras C, Gokbayrak K. Optimal Control for Discrete Event and Hybrid Systems[J]. Modeling, Control, and Optimization of Complex Systems, 2002, (20): 285-304.

（下转第 119 页）