第35卷 第18期 计 算 机 工 程 2009年9月
Vol.35 No.18 Computer Engineering September 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)18—0254—03 文献标识码：A 中图分类号：TP183

微小卫星星务计算机任务调度算法

张小林[1,2]，杨根庆[1]，张宇宁[1,2]
(1. 中国科学院上海微系统与信息技术研究所，上海 200050；2. 中国科学院研究生院，北京 100039)

摘 要：为了在有限的时间内尽可能多地安排具有时间约束的卫星任务，提出一种针对具有独占性、优先级相同的任务的先完成先调度算法EFFFS，对算法性能进行分析，将其与同类算法进行比较，结果表明，该算法具有较小的时间复杂度和较好的调度性能，适用于计算资源受限的环境。

关键词：微小卫星；任务调度；先完成先调度算法

Task Schedule Algorithm for Micro-satellite On-board Computer

ZHANG Xiao-lin[1,2], YANG Gen-qing[1], ZHANG Yu-ning[1,2]
(1. Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050;
2. Graduate University of Chinese Academy of Sciences, Beijing 100039)

【Abstract】This paper proposes a First-Finished First-Scheduled(FFFS) algorithm of single-resource task schedule for the micro-satellite named EFFFS to resolve the conflicts among tasks which have an exclusion relation and the same priority, analyzes the performance of the algorithm, and compare it with other algorithms. Test results show that the algorithm has less time complexity and better performance, and it is suitable for the environment with limited computing resources.

【Key words】micro-satellite; task schedule; First-Finished First-Scheduled(FFFS) algorithm

微小卫星技术的发展对星务管理系统提出了新的要求。星务计算机需要完成多种有非常严格的时间和资源限制的任务，因此，研究微小卫星星务计算机的任务分配和调度问题非常必要和迫切。本文对单个星务管理计算机在指定时间窗内的任务调度问题进行了研究。

1 相关研究

很多学者对资源受限的规划与调度问题进行了研究，虽然这些算法主要针对通用的调度问题，但可以移植到卫星应用中[1-4]。文献[5]对几种主要的针对单资源调度问题的算法进行了性能比较。这些算法都是基于启发式局部搜索算法或者基于约束的方法。

2 单资源任务调度模型的建立

在单个星务管理计算机的情况下，对于有 N 个请求的任务，需要最小化不能被调度的任务数。首先进行如下假设：(1)任务的时间窗限制在[1,1 440] min内。(2)任务开始执行后，就不能被抢占执行。(3)任务不能并行处理。

任务 $T[j]$ 由下面5个参数组成：

$$T[j]=<S\_w,\ F\_w,\ D,\ S,\ F> \tag{1}$$

其中，$T[j].S\_w$ 和 $T[j].F\_w$ 表示任务 $T[j]$ 时间窗的下边界和上边界。$T[j]$ 的持续时间为 $T[j].D$，$T[j]$ 的开始和结束时间为 $T[j].S$ 和 $T[j].F$，$T[j].F=T[j].S+T[j].D$。$T[j]$ 的时间窗余量是 $T[j].F\_w-T[j].S\_w-T[j].D$。$T[j]$ 可以在其时间窗内任意调度。$T[j]$ 可能的最早开始时间表示为 $T[j].S\_e$，其值等于 $[T[j].S\_w$，$T[j].F\_e$ 表示 $T[j]$ 可能的最早完成时间，其值等于 $[T[j].S\_w+T[j].D$。

如果任务 $T[j]$ 可以在其时间窗内被处理，$U[j]=0$，如果不能被按时处理，$U[j]=1$，目标就是最小化不能被按时调度的任务数：

$$\min \quad \sum_{j=1}^{n} U[j] \tag{2}$$

2.1 先完成先调度算法

先完成先调度(First-Finished First-Scheduled, FFFS)算法从空的调度表开始，首先安排最早能完成的任务。调度算法从参数 T 开始，T 是请求服务的队列：

$$T=[T1,T2,T3,\cdots,Tj] \tag{3}$$

FFFS算法的伪代码如下：

```
req_gen(T, finish_time)
 {req_list=[]
 for (i =0;i<=len(T);i++)
     if    finish_time <= T(i).S_w:
                    T(i).S=T(i).S_w
                    T(i).F=T(i).S_w+T(i).D
          把任务 T(i)加入到  req_list 中
           elseif    finish_time+T(i).D<=T(i).F_w:
                     T(i).S=finish_time
                     T(i).F=finish_time+t.D
               把任务 T(i)加入到 req_list 中
                     endif;
          return req_list }
 fffs(T)
 {     finish_time=0;
              sched=[];/*初始调度表为空*/
               while true
  T=req_gen(T, finish_time)
  if len(T)==0
       break;
```

基金项目：上海市科委国际合作项目基金资助项目(052207046)
作者简介：张小林(1981－)，男，博士研究生，主研方向：高可信星载计算机；杨根庆，研究员、博士生导师；张宇宁，博士研究生
收稿日期：2009-01-20 **E-mail：**zxlin1981@163.com

```
        end
    对 T 中的任务按照其最早完成时间排序;
                finish_time=T[0].F
                把 T[0]加入到调度表 sched 中
                把 T[0]从 T 中删除
            return sched    }
```

任务列表 *T* 中每个任务的开始时间 *T*[*j*].*S* 通过计算确定。*finish_time* 表示最近调度的一个任务的完成时间，初始值为 0，*sched* 表示已经安排的调度表，初始时为空的调度表。之后，随着循环的进行，函数 req_gen 被执行，它使每个 *T*[*j*] 在其时间窗[*T*[*j*].*S_w*, *T*[*j*].*F_w*]有最早的开始时间 *T*[*j*].*S_e*，如果能够被安排，就把这个任务放入 *req_list* 中，如果一个任务不能在其时间窗内被调度，就会被丢弃，然后通过 req_gen 返回 *req_list*。如果 *req_list* 为空，算法结束，返回 *sched*；否则，对每个 *req_list* 中任务 *T*[*j*]的完成时间进行排序，最早完成的任务被排为 *T*[0]，其完成时间保存在变量 *finish_time* 中，最后把 *T*[0]加入 *sched*，同时从 *T* 中移除该任务。

2.2 改进的 FFFS 算法 EFFFS

FFFS 算法简单、计算量小，但效率不高。下面针对其存在的问题进行分析。

(1)可能丢弃存在共存可能性的任务

如图 1 所示，*T*[0]可能被安排为 *T*[0]-*a* 和 *T*[0]-*b*。先调度 *T*[0]-*a*，*T*[1]将会被丢弃，只能完成一个任务，但实际上 *T*[0]-*b* 和 *T*[1]-a 可以共存。如果先调度 *T*[1]-*a*，再调度 *T*[0]-b，则 2 个任务都能被调度。

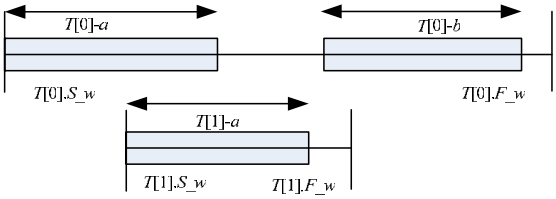
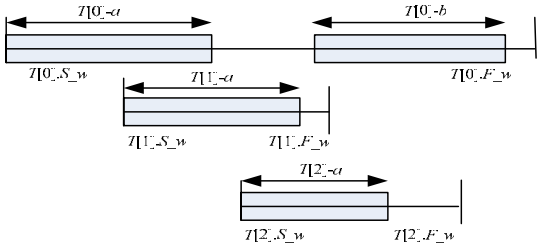


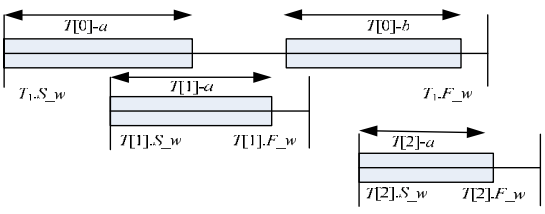
图 1　FFFS 算法存在的问题

(2)替换后可能出现的问题

如图 2(a)所示，根据(1)的改进应该调度 *T*[0]-*b* 和 *T*[1]-*a*，而采用 FFFS 算法则会调度 *T*[0]-*a* 和 *T*[2]-*a*。2 种任务都可以被调度，但 FFFS 的调度时间更短，因此，希望选择 *T*[0]-*a* 和 *T*[2]-*a*。而在图 2(b)的情况下，希望选择 *T*[0]-*b* 和 *T*[1]-*a*。



(a)期望调度 T[0]-a 和 T[2]-a



(b)期望调度 T[0]-b 和 T[1]-a

图 2　FFFS 改进方案

下面提出改进的 FFFS 算法 EFFFS。它采用与 FFFS 算法一样的 req_gen 函数。算法的伪代码如下：

```
efffs(req):
  finish_time=0
  sched=[];/*初始调度表为空*/
  T=req_gen(T, finish_time)
   if len(T)==0
        break;
        end
对 T 中的任务按最早完成时间 T[i].F_e 排序;
                finish_time=T[0].F_e
                  for (i =1;i<=len(T);i++)
 if T[i].F_w-T[i].D<=finish_time and T[i].F_e<T[0].F_w-t[0].D
        Ti&0_F=T[i].F_e+T[0].D
              T2=req_gen(T[1:], finish_time)
               删除 T2 中的原任务 T[i];
              if len(T2)==0
            finish_time=T[i].F
             把任务 T[i]加入到调度表中
                删除 T 中的任务 T[i]
               break
        elseif T2[0].F > Ti&0_F: finish_time=T[i].F
        把任务 T[i]加入到调度表中
                删除 T 中的任务 T[i]
                   break
            else      把任务 T[0]加入到调度表中
                删除 T 中的任务 T[0]
                    break
         return sched
```

一个任务将被丢弃时，先对这 2 个任务共存的可能性进行评估，然后评估安排被丢弃任务而放弃本来要安排调度的任务得到的收益。如果选择原本要丢弃的任务，则期望完成该任务后继续完成下一个任务的结束时间比采用 FFFS 完成相同任务数的结束时间早；否则，选择方法与 FFFS 一致。

3　EFFFS 性能分析

下面以表 1 的参数对方法进行仿真。

表 1　实验参数

参数名	参数值
最大时间窗/min	[1,1 440]
时间窗余量/min	[0,50],[0,100],[0,150],[0,200]
持续时间窗/min	[20,60]
任务数	30,40,50,60

任务的时间窗余量和持续时间窗在相应的区间内均匀分布。*T*[*j*].*S_w* 在区间 [0, 1 440-*T*[*j*].*D*]的时间余量-*T*[*j*].*D*]内均匀分布。在每种任务请求数下重复仿真 100 次，取平均值进行分析。

(1)请求任务数和任务持续时间的影响

当请求的任务增加，持续时间变长，2 种算法的效率都降低，但是 EFFFS 的性能比 FFFS 好。

(2)任务时间窗余量的分析

2 种算法都随余量的增加而改进。但是时间窗余量的影响没有请求的任务数和持续时间的影响大。

(3)任务平均持续时间和调度的空闲率分析

用空闲槽表示没有安排任务的空闲时间段，*ESR* 表示总的空闲时间占最大时间窗的比例。

EFFFS 的 *ESR* 比 FFFS 平均有 7%的改进。随着请求任务数的增加，2 种算法的 *ESR* 都减少，但 EFFFS 减少得更快。

在持续时间窗是[0, 100]时，每个任务的预期时间是 50。但仿真结果的平均持续时间 *AD* 在多数情况下低于 50。这是因为 2 种算法总是优先调度最早结束的任务，任务时间短的任务容易被调度，所以 FFFS 的 *AD* 比 EFFFS 平均低了 3%。

被调度的任务数 *NTS* 通过式(4)计算：

$$NTS = \frac{1-ESR}{AD} | MaxTimeWindow | \quad (4)$$

EFFFS 的 *AD* 略大于 FFFS 的 *AD*，但是其 *ESR* 比 FFFS 小，经计算，EFFFS 调度的任务数比 FFFS 平均有 4%的提高。EFFFS 算法对持续时间短的任务更敏感。因此，越短的任务，被调度的可能性越大。

4 与其他算法的比较

本文把 FFFS, EFFFS 与文献[5]中的任务调度算法进行了比较。采用与表 1 同样的参数，重复进行了 40 次实验。表 2 和表 3 分别表示在不同请求任务数的情况下，各种算法与近最优解的平均性能差异。这里近最优解是通过分支限界法得到的，文献[5]也采用它进行评价。从表中可以看出，当任务请求数较小时，局部搜索算法的性能较好。当任务数较大时，启发式算法 EFFFS 和贪心算法得到改进，其中，EFFFS 的性能最好。当请求任务数是 60 时，EFFFS 得到的调度结果比近似最优解好。但近似最优解是通过几分钟，甚至超过 1 h 的计算得出的结果，而 EFFFS 只需几十毫秒。

表 2 请求任务数为 30 时的算法性能比较

算法	不同的时间窗余量下，与近似最优解调度任务数的平均差异/个			
	50 min	100 min	150 min	200 min
FFFS	0.32	1.22	1.79	2.25
EFFFS	0.03	0.18	0.39	0.67
GreedyBN	0.36	1.15	1.85	2.18
GreedyDP	0.20	0.45	0.75	0.75
Hill-clibing	0.00	0.05	0.09	0.20
Genitor	0.00	0.10	0.15	0.22

表 3 请求任务数为 60 时的算法性能比较

算法	不同的时间窗余量下，与近似最优解调度任务数的平均差异/个			
	50 min	100 min	150 min	200 min
FFFS	0.60	2.05	2.30	2.70
EFFFS	-0.25	-0.20	0.15	0.55
GreedyBN	0.63	2.21	2.08	2.58
GreedyDP	0.19	0.48	0.65	0.85
Hill-clibing	0.20	1.04	1.65	1.92
Genitor	0.22	0.67	1.09	1.32

5 结束语

本文提出的 EFFFS 算法与同类算法相比，运算时间大为降低，特别适合计算资源受限的环境。目前，FFFS 和 EFFFS 由于简单高效而成功应用到星载计算机的星务管理任务调度中。

参考文献

[1] 姚 敏, 赵 敏. 基于模糊神经网络的小卫星任务自主调度设计[J]. 宇航学报, 2007, 28(2): 385-388.

[2] 陈 锋, 刘宗田, 石振国, 等. 基于禁忌搜索算法的网格任务调度[J]. 计算机工程, 2007, 33(21): 76-78.

[3] 陈 晶, 潘全科. 求解独立任务调度的离散粒子群优化算法[J]. 计算机工程, 2008, 34(6): 214-215.

[4] Kramer L A, Smith S F. Task Swapping for Schedule Improvement: A Broader Analysis[C]//Proc. of the 14th International Conference on Automated Planning and Scheduling. Whistler, British Columbia, Canada: [s. n.], 2004: 235-243.

[5] Barbulescu L, Watson J P, Whitley L D, et al. Scheduling Space-ground Communications for the Air Force Satellite Control Network[J]. Journal of Scheduling, 2004, 7(1): 7-34.

编辑 张 帆

(上接第 253 页)

平均，消除单个心搏 ST 测量带来的误差，同时发生心律失常的心搏不在统计范围内，避免单个异常心搏给 ST 测量带来的误差，如图 3 所示，其中，走速为 25 mm/s；增益为 5 mm/mV。当连续 1 min ST 段偏移量的均值都在预设的偏差允许范围之外，记 1 次 ST 段改变，开始时间为 1 min 之前。当连续 1 min ST 段偏移量的均值都在预设的偏差允许范围之内，则此次 ST 段改变结束，结束时间为 1 min 之前。对于每一段 ST 段改变记录，给出开始时间、结束时间、平均偏移量、最大偏移量、最大偏移量发生的时间等重要信息。

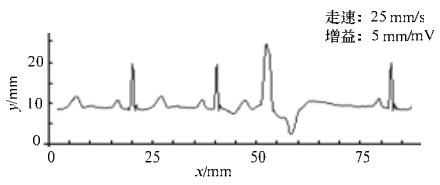


图 3 单个心率失常导致的 ST 段测量误差

(2)当前 ST 段改变状况、出现 ST 段偏移次数、ST 异常占总监护时间的百分比等实时信息。

5 结束语

本文针对动态心电图噪声大的特点，改进了提取 ST 参考等电位线的方法，运用小波变换与局部变换方法对 *J* 点进行检出，改善了 ST 特征值的测量方法，使 ST 段偏移测量的准确度有较大的提高。同时，根据临床实时监控需要设计并实现了实用的 ST 段报警记录信息。该方法使用 C++实现，有较好的实时性，以 M IT/B IH 标准心电数据库的数据进行检验，并应用于清华大学深圳研究生院嵌入式系统与技术实验室开发的远程多生理参数监护系统中[5-6]，通过临床试验中 20 例正常人与 20 例 ST 病变患者连续 4 h 的检测，证明其能得到良好的效果。

参考文献

[1] 郭继鸿. 心电图学[M]. 北京: 人民卫生出版社, 2002.

[2] Martinez J P, Almeida R, Olmos S, et al. A Wavelet-based ECG Delineator: Evaluation on Standard Databases[J]. IEEE Transactions on Biomedical Engineering, 2004, 51(4): 570-577.

[3] 张和君, 张 跃. 基于小波变换的心电信号综合检测算法研究[J]. 计算机工程与设计, 2006, 27(20): 3831-3834.

[4] 王林泓, 杨 浩. 心电信号处理中滤波器设计的研究[J]. 北京生物医学工程, 2002, 21(3): 218-221.

[5] Dai Shaosheng, Zhang Yue. A Wireless Physiological Multi-parameter Monitoring System Based on Mobile Communication Networks[C]//Proc. of IEEE International Symposium on Computer-based Medical Systems. [S. l.]: IEEE Press, 2006.

[6] 成转鹏, 张 跃. 远程心电实时监护终端的设计与实现[J]. 计算机工程, 2007, 33(11): 264-266.

编辑 张 帆