$$\Pr(X<f_0)=F(f_0)>1-\varepsilon \tag{13}$$

就说明任意一个最优值比 f_0 要小的概率是很大的,从而说明 f_0 确实是一个很大的数.然而,$\mathcal{W}$的元素个数是很大的,我们只能随机选取一小部分样本值来估计分布函数 $F(x)$,设选取的样本数量为 N,那么搜索初始值即为这 N 个样本中的最大值.设根据这 N 个样本得到的密度估计函数为 $\hat{f}_N(x)$,那么分布估计函数为 $F_N(x)=\int_{-\infty}^{x} p_N(t)\mathrm{d}t$.我们把分布估计函数当作 $O(\mathcal{W})$的分布函数,利用式(13)来判断是否该选取 f_0 作为搜索初始值.

设 $O(\mathcal{W})$服从分布 $F(x)$,选取的 N 个样本为 $x_1,x_2,\ldots,x_N$,$F(x)$的密度函数为 $f(x)$,我们使用核密度估计方法[40]来估计 $f(x)$,设核密度估计函数为

$$\hat{f}_N(x)=\frac{1}{Nh}\sum_{i=1}^{N}K\left(\frac{x_i-x}{h}\right),$$

其中,h 为窗宽(bandwidth),$K(\cdot)$为核函数.对于核函数,我们选择最常用的高斯核(Gaussian kernel),关于核函数的详细介绍见文献[40].在计算窗宽 h 时,先根据平均积分均方误差(mean integrated square error)确定出最小窗宽表达式,再采用 plug-in 方法[41]计算窗宽的估计值.关于核密度估计的详细算法见文献[40,41].

在实际情况中,所有的样本都不小于 $U_{\min}=\sum_{i=1}^{n}\frac{C_i^{\min}}{T_i}$ 且都不大于 1,因此在求密度估计函数时,可以规定当 $x<U_{\min}$ 或 $x>1$ 时, $\hat{f}_N(x)=0$.

已知密度估计函数 $\hat{f}_N(x)$,也就可以计算出分布估计函数 $F_N(x)$了.当公式(14)满足时,我们就认为当前样本中的最大值 f_0 在所有线性规划最优值中确实是一个足够大的值:

$$\Pr(X<f_0)=F_N(f_0)>1-\varepsilon \tag{14}$$

因此,可以将 f_0 及其对应的最优点 x_0 作为算法 1 的搜索初始值(f_0,x_0).如果 f_0 并不满足公式(14)时,就再通过随机选取并求解线性规划子问题来添加样本.添加新样本后的最大值 f_0 可能发生变化,也可能不会变化,但无论怎样,我们找到的确实比 f_0 小的数量增加了,从而 $\Pr(X<f_0)=F_N(f_0)$也会增加,最终总会超过事先设定的 $1-\varepsilon$.但是随着样本的增加,计算密度估计函数的时间也会增加,因此,采集的样本数量达到某个上限 N 时,就直接取出其中的最大值 f_0 及对应的 x_0 作为搜索初始值了.另外,还需考虑在采样过程中求解密度估计函数的时机,我们设定一个周期 N_p,每采样 N_p 个样本后立即根据所有已知的样本求解密度估计函数 $\hat{f}_N(x)$.算法 3 给出了计算搜索初始值的方法.

算法 3. 计算搜索初始值.

Input:集合 $W_1,W_2,\ldots,W_n$,正数$\varepsilon\in(0,1)$,样本最大值 N,采样周期 N_p;

Output:搜索初始值 f_0,x_0.

1. **function** *FindInitial*(ε,N,N_p)
2. $count\leftarrow 0$ //统计样本的个数
3. $Sample\leftarrow\varnothing$
4. $(f_0,x_0)\leftarrow(0,C^{\min})$
5. **while** $count\leqslant N$ **do**
6. 随机选取元素 $w_{ih_i}\in W_i(i=1,2,\ldots,n)$
7. 求解线性规划 $wLP=wLP\begin{pmatrix}1 & 2 & \ldots & n\\ h_1 & h_2 & \ldots & h_n\end{pmatrix}$,得到最优值 $wOPT$ 和最优点 wx
8. $Sample\leftarrow Sample\cup\{wOPT\}$ //不能去掉 $Sample$ 中的相同元素
9. **if** $wOPT>f_0$ **then**
10. $(f_0,x_0)\leftarrow(wOPT,wx)$
11. **end if**
12. **if** $mod(count,N_p)=0$ **then**

13. 根据样本 *Sample* 计算密度估计函数 $\hat{f}(x)$
14. 计算分布估计函数 $\hat{F}(x)$
15. **if** $\hat{F}(f_0)>1-\varepsilon$ **then**
16. **return** (f_0,x_0)
17. **end if**
18. **end if**
19. **end while**
20. **return** (f_0,x_0)
21. **end function**

3.5 求解LPS模型的算法

通过第 3.2~第 3.4 节的叙述,我们给出求解 LPS 模型的最终算法(见算法 4),定理 3 和定理 4 保证了该算法的正确性.

算法 4. LPS 模型求解算法.

Input:

1. 实时系统的 n 个任务周期 $T_1,T_2,\ldots,T_n$;
2. n 个任务运行时间的范围 $[C_1^{\min},C_1^{\max}],[C_2^{\min},C_2^{\max}],\ldots,[C_n^{\min},C_n^{\max}]$;
3. T:连续 T 次为找到新的最优值;
4. 正数$\varepsilon\in(0,1)$;
5. 样本采集最大数量 N;
6. 样本采样周期 N_p;

Output:实时系统调度的最大 CPU 利用率 f^*以及对应的 n 个任务的执行时间 $C=(C_1^*,C_2^*,\ldots,C_n^*)$.

1. **function** LPSOptimization
2. **for** $i\leftarrow 1$ **to** n **do**
3. 计算 $P_{i-1}(T_i)$和 W_i
4. **end for**
5. $(f_0,x_0)\leftarrow FindInitial(\varepsilon,N,N_p)$
6. $(f^*,C^*)\leftarrow FindOptimal((f_0,x_0),T)$
7. **return** (f^*,C^*)
8. **end function**

4 算法的时间复杂度

下面分析算法 4 的时间复杂度.

计算 $P_i(T_i)$的方法来自文献[18]所提出的二叉树剪枝方法.由二叉树的性质可知,$|P_{i-1}(T_i)|\leqslant 2^{i-1}$;

又因为 $|P_{i-1}(T_i)|\leqslant|S_i|\leqslant\sum_{j=1}^{i}\lfloor T_i/T_j\rfloor\leqslant i\lfloor T_i/T_1\rfloor$,因此计算单个 $P_{i-1}(T_i)$的时间复杂度是线性的 $O(nT_i)$,那么计算所有的集合 $P_0(T_1),P_1(T_2),\ldots,P_{n-1}(T_n)$的时间复杂度为 $O(n^2T_n)$.

根据 W_i 的定义,计算 W_i 时最坏情况下的循环次数为$|P_{i-1}(T_i)|\times 2i\leqslant O(nT_n)\times 2n=O(n^2T_n)$,那么计算 $W_1,W_2,\ldots,W_n$ 的时间复杂度不超过 $O(n^3T_n)$.

在计算搜索初始值时,设每采样 N_p 次计算一次密度函数,一共采样 N 次.那么最坏情况为采样了 N 次均未满足 $\hat{F}(f_0)>1-\varepsilon$.计算密度函数的次数为$\lfloor N/N_p\rfloor$.文献[42]指出:在所有使用核密度估计方法求解近似密度函数的算法时间复杂度为 $O(SV)$,其中,S 为采样样本的个数,V 为样本值的个数.在这里,$S=N$,V 可以近似为 S.因此,

计算$\lfloor N/N_p \rfloor$次密度估计函数的时间复杂度为 $O(N^3/N_p)$.我们使用商用优化软件 MINOS 进行快速求解线性规划问题,文献[43]指出:用 MINOS 求解线性规划的平均时间复杂度为 $O(n+m)$,其中,n 为变量数(也即任务数),m 为不等式数.在本问题中,$m\leqslant 3n$,因此在 LPS 方法中,求解一次线性规划只需要花费线性的时间 $O(n)$.综上所述,计算搜索初始值的时间复杂度不会超过 $O(N^4n/N_p)$.一般情况下,N/N_p 不会取得太大,因此可以把 N/N_p 看做一个常数,实际时间复杂度为 $O(N^3n)$.

考虑剪枝搜索算法的时间复杂度,$\mathcal{W}$中的线性规划问题的总数为

$$K'=|W_1||W_2|...|W_n|\leqslant\prod_{i=1}^{n}iT_i/T_1=\frac{n!T_1T_2...T_n}{T_1^n}.$$

那么搜索树的叶子结点数量为 K',所有的结点数为 $K''\leqslant\prod_{j=1}^{n}k_j$, 其中, $k_j=\prod_{i=1}^{j}\frac{n!T_{n-i+1}...T_{n-1}T_n}{(n-i+1)!T_1^{n-i+1}}$. 虽然这是一个非常庞大的数,由于初始的搜索最优值的大小已经超过了大约$(1-\varepsilon)|\mathcal{W}|$线性规划最优值(在后面的实验中,我们将取$\varepsilon$=0.02),因此在实际的搜索中,大量的子树会被剪枝掉.此外,即使在搜索某棵子树时访问了过多的结点,一旦连续 T 次的结点访问中都没有找到新的最优值,我们就会额外花费 t 的时间来寻找可以剪枝的结点,这样又减少了大量搜索所消耗的时间.时间 t 的计算如下:设当前访问的结点 *wNode* 的深度为 d,那么访问结点的总次数的最坏情况为

$$\sum_{i=d}^{n}\sum_{j=1}^{i-1}|W_j|\leqslant(n-d+1)\sum_{i=1}^{n-1}|W_i|\leqslant(n-d+1)\sum_{i=1}^{n-1}iT_i/T_1\leqslant\frac{(n-d+1)n(n-1)T_n}{T_1}.$$

因此,$t=O(n^3T_n)$.这是一个伪多项式的时间.用 MINOS 计算线性规划的平均时间为 $O(n)$,因此算法 1 的时间复杂度为 $O(k_vn+n^4T_nK_v/T)$,其中,K_v 在实际情况中运行算法 2 的次数是一个较小的数,因此我们可以把 K_v/T 看做常数.因此,实际复杂度为 $O(k_vn+n^4T_n)$.

综上,LPS 算法的理论时间复杂度为 $O(n^2T_n+n^3T_n+N^3n+k_vn+n^4T_n)=O(N^3n+k_vn+n^4T_n)$.事实上,在实时系统中,$T_n$ 是一个有界的变量,因此可以把 T_n 看做一个常数.而对于采样次数 N,我们不可能采集关于 n 的指数次样本,否则也就失去了采样的意义,因此,采样次数必须是一个关于 n 的多项式,设 $N=O(n^\alpha)$,其中,$\alpha>0$.我们把α称为采样因子.当$\alpha\leqslant 1$ 时,$O(N^3n+n^4T_n)=O(n^4)$;当$\alpha>1$ 时,$O(N^3n+n^4T_n)=O(n^{3\alpha+1})\geqslant O(n^4)$.因此对任意$\alpha>0$,有 $O(N^3n+n^4T_n)=O(n^{3\alpha+1})$.从而在实际情况中,LPS 算法的时间复杂度为

$$O(n^{3\alpha+1}+K_vn),\alpha>0 \tag{15}$$

从公式(15)中可以看出:对 LPS 算法的时间开销起决定性作用的因素在于选取搜索初始最优值以及深度优先搜索,同时,前者的结果又影响后者的运行时间.α越大,选取搜索初始最优值的时间开销就越大,但相对地访问的结点总数 K_v 就越少.

5 实验结果与分析

本文利用 Matlab 优化工具包及 Matlab 的优化工具接口 Tomlab[44]进行实验,比较 LPS、基于 MBP 与基于 NLP 的 3 种方法求解实时系统 RM 优化设计问题所得到的最优值(CPU 最大利用率)和时间开销.其中:在 LPS 方法中,求解线性规划时使用 Tomlab 中的 MINOS 包;在基于 MBP 的方法中,求解混合整数线性规划时使用 Tomlab 中的 CPLEX 包.算法实验环境为 Windows 7,Intel i7-3770,8GM RAM.

实验条件描述如下:算法的精度为 10^{-4},任务周期 T_i 从[50,5000]中均匀、随机地选取.这样选取有两个原因:(1) 区间跨度大,不等式的变量系数范围可以在 1~100 之间;(2) 各个系数的值会尽量保证不同.任务执行时间 C_i 的区间为 $\left[\frac{1}{10n}T_1,\lambda T_i\right]$, 其中,$\lambda$在[0.4,0.6]之间随机选取.这是因为 3 种方法均为优化算法,必须保证每个问题均有可行解,变量下界 $T_1/10n$ 能够保证每个问题都有可行解;变量上界的设定是为了保证当所有运行时间均达到上界时系统一定不可调度,从而才能够发挥各方法的作用.在基于 NLP 的方法中,为了在保证实验精度范围内使得运行时间较短,根据公式(4)我们设定Δv 的值也为 10^{-4}.在基于 MBP 的方法中,M 值的数量级至少应是所有在

算法中出现的常数值数量级的两倍,注意到,任务周期最大是 5 000,从而将 M 定为 10^8.在 LPS 方法中,设置 ε=0.02.T=max(1000,40n)为未更新最优值情况下的连续搜索结点最大数,n 越大,T 越大.为了在采样过程中不耗费大量时间,同时又能获得一个较大的结果,我们设定采样周期为 N_p=ln|w|,采样最大数量 N=10N_p.对相同的任务数 n,随机生成 10 组任务集进行实验,将得到的 10 个结果的平均值作为实验结果.实验结果见表 1.

从表 1 中可以看出:LPS 方法与基于 MBP 的方法所得到的最优值相同;而基于 NLP 的方法得到的最优值就小于 LPS 与基于 MBP 的方法,这是因为使用基于 NLP 的方法很容易掉入局部最优值.当 n 较小时,基于 MBP 的方法运行时间最少;但随着 n 的增大,运行时间陡然增加;当 n>30 时,几乎不能算出结果来.而此时,LPS 的优势逐渐显现出来.基于 MBP 的方法失效的原因主要有 3 点:(1) 问题搜索空间大,算法运行时间随规模呈指数增长;(2) 当中间迭代点出现多个非整数分量时,目前没有理论告诉我们最好的选择策略,只能是进行随机选择,不能够提升算法性能;(3) 该方法引入了相当于原问题 1 倍数量的变量,这也增大了该方法的运行时间.当 n>50 时,基于 NLP 的方法的运行时间大幅增加,而 LPS 仍然能在较短的时间内得到最优解(如图 2 所示).基于 NLP 的方法在这种情况下失效的原因有两点:一是该方法把原来的线性约束优化变成了一个非线性约束优化问题,而非线性约束优化算法相比于线性约束优化算法有本质的区别,其时间复杂度远高于线性约束优化算法;另一方面,除变量自身的区间约束条件外,该方法将其余每个线性不等式的长度被扩大了 1 倍,这也增加了运行处理的时间.因此:在 n 比较小时,选择基于 MBP 的方法;当 n 稍大时,可以选择 LPS 或基于 NLP 的方法;当 n 很大时,就应该选择 LPS 方法.

实验结果还表明,LPS 方法所产生的线性规划搜索树的结点个数是随 n 增长而呈指数级增长的.但实际的搜索结点数与任务数并不是指数的关系,事实上,利用对线性规划搜索树进行剪枝,实际访问的结点数只占总结点数的很少一部分.

Table 1 Optimal values and elapsed time of the three methods, and the number of total nodes and actually visited nodes by LPS method

表 1 3 种方法的最优值和运行时间,LPS 方法的总结点与实际访问结点数

任务数n	不等式数	基于NLP的方法		基于MBP的方法		LPS			
		U	运行时间(s)	U	运行时间(s)	U	运行时间(s)	总结点数	实际访问的结点数
5	22	0.980 0	0.532	0.980 9	0.177	0.980 9	0.640	482	14
10	121	0.969 4	1.07	0.977 2	0.554	0.977 2	1.45	3.37×10^{8}	137
15	293	0.971 6	2.48	0.978 8	3.23	0.978 8	3.60	2.03×10^{16}	298
20	510	0.975 1	5.37	0.977 6	10.2	0.977 6	6.34	1.47×10^{25}	223
25	1 090	0.968 4	10.9	0.975 2	17.5	0.975 2	7.99	2.07×10^{35}	3 457
30	2 175	0.977 3	25.8	0.979 1	115	0.979 1	14.6	5.44×10^{47}	7 494
35	2 369	0.976 7	39.5	0.978 2	546	0.978 2	43.6	9.84×10^{55}	10 423
40	3 439	0.975 0	63.1	–	–	0.976 4	25.3	7.60×10^{72}	6 285
45	3 804	0.975 1	133	–	–	0.977 8	40.2	4.02×10^{77}	14 576
50	4 498	0.974 3	234	–	–	0.976 1	57.7	3.27×10^{92}	17 092
60	7 035	0.974 8	624	–	–	0.976 4	123	5.86×10^{118}	21 649
70	7 491	0.958 2	862	–	–	0.976 1	404	3.34×10^{130}	33 342
80	9 400	0.971 3	1 673	–	–	0.976 3	358	4.52×10^{159}	23 950
90	12 797	–	–	–	–	0.973 7	545	1.83×10^{181}	33 071
100	17 424	–	–	–	–	0.974 4	784	1.30×10^{205}	41 027

6 结论及未来工作

本文提出了一种新的实时系统 RM 优化设计算法——基于树状的线性规划搜索(LPS)方法.首先将 RM 优化设计模型分拆成若干个线性规划子问题,再构造线性规划问题的搜索树,利用深度优先搜索及其剪枝算法,找到线性规划子问题中最大的最优值,从而得到了 CPU 利用率最大值.与已有的两种方法相比,LPS 方法的运行速度更快;尤其当任务数量较大时,LPS 仍然能在较短的时间内运行,但基于 NLP 和 MBP 的方法已经不能适用.

本文的工作可以与计算机可满足性模定理(satisfiability modulo theories,简称 SMT)领域[45]的多个研究热点问题联系起来.提出的 LPS 方法可以求解 SMT 中线性运算(linear arithmetic)部分的可满足性及优化问

题[46−48].我们接下来将比较研究 SMT 求解器 Z3[49]、Yices[50]以及基于 SMT 的优化求解器 OPT-MathSAT[47]、Symba[48]的线性运算部分,进一步改进 LPS 方法,提高 SMT 中线性运算部分的可满足性和优化问题的求解效率.

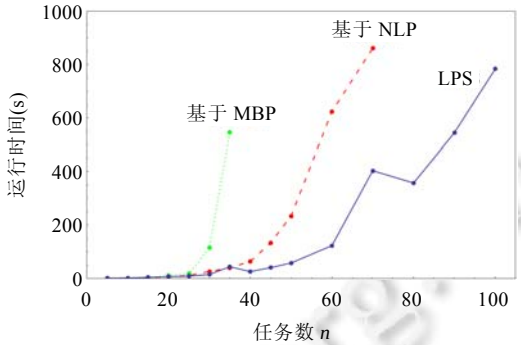

Fig.2 Relationship between task number and elapsed time by the three methods

图 2 任务数与 3 种方法的运行时间的关系

References:

[1] Mall R. Real-Time Systems: Theory and Practice. New Delhi: Pearson Education India, 2007.

[2] Burchard A, Liebeherr J, Oh Y, Son SH. New strategies for assigning real-time tasks to multiprocessor systems. IEEE Trans. on Computers, 1995,44(12):1429−1442. [doi: 10.1109/12.477248]

[3] Wang YJ, Chen QP. On schedulability test of rate monotonic and its extendible algorithms. Ruan Jian Xue Bao/Journal of Software, 2004,15(6):799−814 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/799.htm

[4] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973, 20(1):46−61. [doi: 10.1145/321738.321743]

[5] Davis RI, Zabos A, Burns A. Efficient exact schedulability tests for fixed priority real-time systems. IEEE Trans. on Computers, 2008,57(9):1261−1276. [doi: 10.1109/TC.2008.66]

[6] Stankovic J, Spuri M, Ramamritham K, Buttazzo G. Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms. Dordrecht: Kluwer Academic, 1998.

[7] Bini E, Buttazzo G. A hyperbolic bound for the rate monotonic algorithm. In: Proc. of the 13th Euromicro Conf. on Real-Time Systems. Delft: IEEE Computer Society Press, 2001. 59−66. [doi: 10.1109/EMRTS.2001.934000]

[8] Kuo TW, Mok AK. Load adjustment inadaptivereal-time systems. In: Proc. of the 12th Real-Time Systems Symp. IEEE Computer Society Press, 1991. 160−170. [doi: 10.1109/REAL.1991.160369]

[9] Kuo TW, Liu YH, Lin KJ. Efficient on-line schedulability tests for priority driven real-time systems. In: Proc. of the 6th Real-Time Technology and Applications Symp. IEEE Computer Society Press, 2000. 4−13. [doi: 10.1109/RTTAS.2000.852446]

[10] Han CC, Lin KJ, Hou CJ. Distance-Constrained scheduling and its applications to real-time systems. IEEE Trans. on Computers, 1996,45(7):814−826. [doi: 10.1109/12.508320]

[11] Lauzac S, Melhem R, Mossé D. An efficient RMS admission control and its application to multiprocessor scheduling. In: Proc. of the 1st Merged Int'l Parallel Processing Symp. and Symp. on Parallel and Distributed Processing. IEEE Computer Society Press, 1998. 511−518. [doi: 10.1109/IPPS.1998.669964]

[12] Lauzac S, Melhem R, Mossé D. An improved rate-monotonic admission control and its applications. IEEE Trans. on Computers, 2003,52(3):337−350. [doi: 10.1109/TC.2003.1183948]

[13] Lu WC, Lin KJ, Wei HW, Shih WK. Rate monotonic schedulability tests using period-dependent conditions. Real-Time Systems, 2007,37(2):123−138. [doi: 10.1007/s11241-007-9034-1]

[14] Park DW, Natarajan S, Kanevsky A. Fixed-Priority scheduling of real-time systems using utilization bounds. Journal of Systems and Software, 1996,33(1):57−63. [doi: 10.1016/0164-1212(95)00105-0]

[15] Lee CG, Sha L, Peddi A. Enhanced utilization bounds for QoS management. IEEE Trans. on Computers, 2004,53(2):187−200. [doi: 10.1109/TC.2004.1261828]

[16] Lehoczky J, Sha L, Ding Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Proc. of the 10th IEEE Real Time Systems Symp. Santa Monica: IEEE Computer Society Press, 1989. 166−171. [doi: 10.1109/REAL.1989.63567]

[17] Bini E, Buttazzo G. The space of rate monotonic schedulability. In: Proc. of the 23th IEEE Real-Time Systems Symp. IEEE Computer Society Press, 2002. 169−178. [doi: 10.1109/REAL.2002.1181572]

[18] Liu JX, Wang YJ, Cartmell M. An improved ratemonotonic schedulabilitytest algorithm. Ruan Jian Xue Bao/Journal of Software, 2005,16(1):89−100 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/89.htm

[19] Min-Allah N, Khan SU, Wang X, Zomaya AY. Lowest priority first based feasibility analysis of real-time systems. Journal of Parallel and Distributed Computing, 2013,73(8):1066−1075. [doi: 10.1016/j.jpdc.2013.03.016]

[20] Audsley N, Burns A, Richardson M, Tindell K, Wellings AJ. Applying new scheduling theory to static priority preemptive scheduling. Software Engineering Journal, 1993,8(5):284−292. [doi: 10.1049/sej.1993.0034]

[21] Sjödin M, Hansson H. Improved response-time analysis calculations. In: Proc. of the 19th Real-Time Systems Symp. IEEE Computer Society Press, 1998. 399−408. [doi: 10.1109/REAL.1998.739773]

[22] Min-Allah N, Khan SU, Ghani N, Li J, Wang L, Bouvry P. A comparative study of rate monotonic schedulability tests. Journal of Supercomputing, 2012,59(3):1419−1430. [doi: 10.1007/s11227-011-0554-z]

[23] Díaz-Ramírez A, Mejía-Alvarez P, Leyva-del-Foyo LE. Comprehensive comparison of schedulability tests for uniprocessor rate-monotonic scheduling. Journal of Applied Research and Technology, 2013,11(3):408−436. [doi: 10.1016/S1665-6423(13)71551-7]

[24] Liu JX, Wang YJ, Wang Y, Xing JS, Zeng HT. Real-Time system design based on logic OR constrained optimization. Ruan Jian Xue Bao/Journal of Software, 2006,17(7):1641−1649 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1641.htm

[25] Chung JY, Liu JWS, Lin KJ. Scheduling periodic jobs that allow imprecise results. IEEE Trans. on Computers, 1990,39(9):1156−1174. [doi: 10.1109/12.57057]

[26] Liu JWS, Lin KJ, Shih WK, Yu ACS, Chung JY, Zhao W. Algorithms for scheduling imprecise computations. Computer, 1991, 24(5):58−68. [doi: 10.1109/2.76287]

[27] Dey JK, Kurose J, Towsley D. On-Line scheduling policies for a class of IRIS (increasing reward with increasing service) real-time tasks. IEEE Trans. on Computers, 1996,45(7):802−813. [doi: 10.1109/12.508319]

[28] Rajkumar R, Lee C, Lehoczky J, Siewiorek D. A resource allocation model for QoS management. In: Proc. of the 18th Real-Time Systems Symp. IEEE Computer Society Press, 1997. 298−307. [doi: 10.1109/REAL.1997.641291]

[29] Millan-Lopez V, Feng W, Liu JWS. Using the imprecise-computation technique for congestion control on a real-time traffic switching element. In: Proc. of the Int'l Conf. on Parallel and Distributed Systems. IEEE Computer Society Press, 1994. 202−208. [doi: 10.1109/ICPADS.1994.590126]

[30] Ramanathan P. Graceful degradation in real-time control applications using (*m*,*k*)-firm guarantee. In: Proc. of the 27th Annual Int'l Symp. on Fault-Tolerant Computing. IEEE Computer Society Press, 1997. 132−141. [doi: 10.1109/FTCS.1997.614086]

[31] Chen X, Cheng AMK. An imprecise algorithm for real-time compressed image and video transmission. In: Proc. of the 6th Int'l Conf. on Computer Communications and Networks. IEEE Computer Society Press, 1997. 390−397. [doi: 10.1109/ICCCN.1997.623341]

[32] Feng WC, Liu JWS. Algorithms for scheduling real-time tasks with input error and end-to-end deadlines. IEEE Trans. on Software Engineering, 1997,23(2):93−106. [doi: 10.1109/32.585499]

[33] Hansson J, Son SH, Stankovic JA, Andler S. Dynamic transaction scheduling and real location in overloaded real-time database systems. In: Proc. of the 5th Int'l Conf. on Real-Time Computing Systems and Applications. IEEE Computer Society Press, 1998. 293−302. [doi: 10.1109/RTCSA.1998.726430]

[34] Min-Allah N, Khan SU, Wang Y. Optimal task execution times for periodic tasks using nonlinear constrained optimization. Journal of Supercomputing, 2012,59(3):1120−1138. [doi: 10.1007/s11227-010-0506-z]

[35] Wang Y, Lane DM. Solving a generalized constrained optimization problem with both logic AND and OR relationships by a mathematical transformation and its application to robot motion planning. IEEE Trans. on Systems, Man, and Cybernetics (Part C: Applications and Reviews), 2000,30(4):525−536. [doi: 10.1109/5326.897079]

[36] Boggs PT, Tolle JW. Sequential quadratic programming. Acta Numerica, 1995,4(1):1−51.

[37] Byrd RH, Hribar ME, Nocedal J. An interior point algorithm for large-scale nonlinear programming. SIAM Journal on Optimization, 1999,9(4):877−900. [doi: 10.1137/S1052623497325107]

[38] Hillier F, Lieberman G. Introduction to Operations Research. McGraw-Hill, 2001.

[39] Jenkyns T, Stephenson B. Fundamentals of Discrete Math for Computer Science. London: Springer-Verlag, 2012. [doi: 10.1007/978- 1-4471-4069-6]

[40] Wasserman L. All of Nonparametric Statistics, Vol.4. New York: Springer-Verlag, 2006. [doi: 10.1007/0-387-30623-4]

[41] Sheather SJ, Jones MC. A reliable data-based bandwidth selection method for kernel density estimation. Journal of the Royal Statistical Society (Series B: Methodological), 1991,53(3):683−690.

[42] Raykar VC, Duraiswami R, Zhao LH. Fast computation of kernel estimators. Journal of Computational and Graphical Statistics, 2010,19(1):205−220. [doi: 10.1198/jcgs.2010.09046]

[43] Andrei N. On the complexity of MINOS package for linear programming. Studies in Informatics and Control, 2004,13(1):35−46.

[44] TOMLAB. The TOMLAB optimization environment for fast and robust large-scale optimization in MATLAB. http://tomopt.com/tomlab/

[45] De Moura L, Bjørner N. Satisfiability modulo theories: Introduction and applications. Communications of the ACM, 2011,54(9): 69−77. [doi: 10.1145/1995376.1995394]

[46] Dutertre B, De Moura L. A fast linear-arithmetic solver for DPLL (T). In: Proc. of the 18th Int'l Conf. on Computer Aided Verification. Springer-Verlag, 2006. 81−94. [doi: 10.1007/11817963_11]

[47] Sebastiani R, Tomasi S. Optimization in SMT with LA(Q) cost functions. In: Proc. of the 6th Int'l Joint Conf. on Automated Reasoning. Springer-Verlag, 2012. 484−498. [doi: 10.1007/978-3-642-31365-3_38]

[48] Li Y, Albarghouthi A, Kincaid Z, Gurfinkel A, Chechik M. Symbolic optimization with SMT solvers. In: Proc. of the 41st Annual ACM SIGPLAN-SIGACT Symp. on Principles of programming languages. ACM Press, 2014. 607−618. [doi: 10.1145/2535838.2535857]

[49] De Moura L, Bjørner N. Z3: An efficient SMT solver. In: Proc. of the Theory and Practice of Software, 14th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Springer-Verlag, 2008. 337−340. [doi: 10.1007/978-3-540-78800-3_24]

[50] Dutertre B. Yices 2.2. In: Proc. of the 8th Int'l Joint Conf. on Automated Reasoning. Springer-Verlag, 2014. 737−744. [doi: 10.1007/978-3-319-08867-9_49]

附中文参考文献:

[3] 王永吉,陈秋萍.单调速率及其扩展算法的可调度性判定.软件学报,2004,15(6):799−814. http://www.jos.org.cn/1000-9825/15/799.htm

[18] 刘军祥,王永吉,Matthew Cartmell.一种改进的 RM 可调度性判定算法.软件学报,2005,16(1):89−100. http://www.jos.org.cn/1000-9825/16/89.htm

[24] 刘军祥,王永吉,王源,邢建生,曾海涛.基于逻辑"或"约束优化的实时系统设计.软件学报,2006,17(7):1641−1649. http://www.jos.org.cn/1000-9825/17/1641.htm

陈力(1989－),男,重庆人,博士生,主要研究领域为可满足性模理论,实时系统,优化算法.

吴敬征(1982－),男,博士,副研究员,主要研究领域为隐蔽信道分析,网络信息安全,安全操作系统.

王永吉(1962－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为虚拟化技术,隐蔽信道,实时系统,人工智能,数据挖掘,软件工程.

吕荫润(1991－),男,硕士生,主要研究领域为可满足性模理论,优化算法.