文章编号:1001 - 9081(2005)04 - 0962 - 04

战术层库存/运输优化问题的一种求解算法

闫 昱[1],袁庆达[2],郭文夷[1]
(1. 上海第二工业大学 计算机与信息学院, 上海 201209; 2. 上海对外贸易学院 国际经贸学院,上海 201620)
(qd2yuan@sina.com)

摘 要:讨论了现代物流管理中比较重要、复杂的战术层随机库存/运输联合优化问题的一种求解思路,并根据求解约束集中器选址问题(CCLP)的方法和禁忌搜索算法设计了一个启发式算法。重点介绍了编程实现此算法的要点,并用模拟算例对设计的算法进行了验证,计算结果是比较理想的。

关键词:库存—运输联合优化;约束集中器选址问题;禁忌搜索算法
中图分类号: TP311.52 **文献标识码**:A

Heuristic algorithm of the tactics inventory/transportation integrated optimization problem

YAN Yu[1], YUAN Qing-da[2], GUO Wen-yi[1]
(1. *College of Computer and Information, Shanghai Second Polytechnic University, Shanghai* 201209, *China*;
2. *International Business School, Shanghai Institute of Foreign Trade, Shanghai* 201620, *China*)

Abstract: A way for solving the important and complicated problem of the inventory/transportation integrated optimization in the modern logistics tactics plan was discussed, and a heuristic algorithm based on the method for soling the capacitated concentrator location problem and the tabu search algorithm was discussed. And then the main points of programming were emphasized. Finally, the algorithm was tested by some simulated cases and the result was perfect.

Key words: inventory/transportation integrated optimization; CCLP; tabu search algorithm

库存和运输分别是物流系统中创造时间价值和空间价值的关键要素,但它们又存在显然的"效益背反"性,因此,如何平衡这两者的关系从而寻求物流系统的整体经济性是理论和实践都需要解决的问题,也是现代物流研究的重点。在物流管理实践中,无论是采购物流中的 JIT 采购,还是销售物流中的通城配送,所涉及的都是多客户、多频次、小批量的运输,而在战术层决策时,客户的需求又多是随机的。虽然库存分配问题和运输问题都是运筹学中的经典问题,也存在一些算法,但对于规模较大的问题,并不存在多项式时间算法,比如运输问题,实际上往往是多个 TSP 问题,而 TSP 问题本身就是已知的 NP-Hard 问题。考虑到企业在不同的物流决策阶段面对的问题不同,本文以战术层的库存/运输问题为出发点,重点介绍一个基于选址问题的启发式算法的设计和实现过程。

1 问题介绍

考虑一个由一个中央仓库和 n 个客户组成的单物品配送系统。假设客户到中央仓库的距离 $d_i(i=1,2,\cdots,n)$ 已知,客户之间的距离 $d_{ij}(i,j=1,2,\cdots,n)$ 已知,客户 j 的最大存储能力 C_j 已知,单位时间的需求量是随机的,但均值和方差已知,并假设不同客户的需求是相互独立的,同一客户不同单位时间的需求也相互独立且服从相同分布。客户单周期的需求量远小于车辆载重量,每辆车必须为多个客户服务。设 p_j 表示客户 j 每个周期可以接受的最大缺货概率。单位商品单位时间保管费为 h。设采用同种类型的配送车辆,其载重量为 W,单位距离走行费用为一常数,不妨假设其等于 1。每辆车执行任务时产生的固定费用为 F。此外由于运输和仓库作业时间的限制,配送时间间隔必定要大于某个值,这里假设配送时间间隔的最小值为 T^{LB}。对于这样的一个系统,战术层物流决策需要确定在系统将来实际运作时使其单位时间的库存和运输费用最小的库存分配策略和运输策略,即需确定对每个客户的配送周期和配送量(库存策略)及所需准备的车辆数和车辆行驶路线(运输策略)。

如果不考虑库存相关因素,需要求解的问题就是车辆路线问题(Vehicle Routing Problem, VRP)[1],也就是操作层的决策问题而非战术层面的问题。本文讨论问题的侧重点是确定客户的分组,进而确定对每组客户的配送周期和这个配送系统所需车辆数,并不过分强调路线的优化,是战术层面的问题。

2 求解思路和算法中所用方法

对于前面描述的问题,正如文献[2]所指出的,求解此类问题的最优解(策略)将是非常复杂的,因此我们从对配送区域内的所有客户进行分组来求解和设计算法。但如果直接处理客户划分问题,文献[3]指出类似问题是 NP-Complete,同样不是多项式可解的。本文在对所有客户进行分组时,并不直接去求解这个划分问题,而是将之转化为约束集中器选址问题(Capacitated Concentrator Location Problem, CCLP)。在采用 CCLP 问题求解算法获得客户分组后,对之进行处理获得本文所研究问题的解,然后应用禁忌搜索算法(Tabu Search, TS)对解进行优化,这就是本文总的求解思路。

2.1 约束集中器选址问题介绍

CCLP 是通信领域中的一个问题,文献[4]将之应用到约束车辆路线问题当中(Capacitated Vehicle Routing Problem,

收稿日期:2004 - 09 - 01 **基金项目**:上海高校优秀青年教师资助项目(JS304001)
作者简介:闫昱(1973 -),女,陕西人,讲师,硕士,主要研究方向:数据库、优化算法设计、信息系统开发; 袁庆达(1972 -),男,黑龙江人,讲师,博士,主要研究方向:物流和供应链管理、物流管理信息系统; 郭文夷(1954 -),男,上海人,副教授,硕士,主要研究方向:计算机软件.

CVRP),并得到了较其他算法更优良的解。可以这样对 CCLP 描述:给定 m 个集中器(具有固定容量 $Q_j(j=1,\cdots,m)$)可能的位置和 n 个终端,终端 i 占用集中器 w_i 的容量。问题是确定需要的集中器个数($\leqslant m$)和集中器位置,以及每个终端应与哪个集中器相连,在满足每个终端仅与一个集中器连接和集中器容量约束的前提下,使得总费用最小。费用包括将集中器设置在位置 j 时的安装费 $v_j(j=1,\cdots,m)$,以及终端与集中器的连接费 $c_{ij}(i=1,\cdots,n;j=1,\cdots,m)$。

令 y_j 和 x_{ij} 为 0 - 1 变量,$y_j=1$ 表示选择位置 j,否则为 0;$x_{ij}=1$ 表示终端 i 连接到集中器 j,否则为 0。则 CCLP 可以用下面的整数线性规划模型表示:

$$\text{Minimize } P=\sum_{i=1}^{n}\sum_{j=1}^{m}c_{ij}x_{ij}+\sum_{j=1}^{m}v_jy_j$$

$$\text{s.t.}\quad \sum_{j=1}^{m}x_{ij}=1 \qquad \forall i \tag{1}$$

$$\sum_{i=1}^{n}w_ix_{ij}\leqslant Q_j \qquad \forall j \tag{2}$$

$$x_{ij}\leqslant y_j \qquad \forall i,j \tag{3}$$

但 CCLP 依旧是 NP-Hard 问题[5],并不存在求解其最优解的多项式时间算法,但由于它的约束和目标函数结构都要比本文所研究的问题简单,故更容易在可接受的时间内获得满意解。

2.2　问题转化

通过上节介绍可知,CCLP 实际上也就是获得对所有终端的一个最优分组。将之应用到本文设计的算法中主要是借助其易解的特性来简化对客户划分的处理。为了应用 CCLP,就要给出集中器设置费 v_j,终端与集中器的连接费 c_{ij},也即如何用所研究问题中的费用因素来表征它们。下面将从集中器集合的预定、集中器的设置费和终端与集中器的连接费三个方面阐述本文欲采用的转化过程。

在 CCLP 问题中,集中器的集合是给定的。为了应用它的方法,就需要确定本文研究的问题中选哪些客户、选多少客户来形成类似集中器集合的集合,不妨将之称为种子客户集合。本文考虑一种自动生成方式。自动生成种子客户集合的方式是指种子集合的产生由算法生成,不需要人工干预。比如可以先将所有客户都看作种子客户,然后通过解 n 个变形 0 - 1 背包问题确定选择哪些客户作为种子客户。这种方法的计算量明显要大些,但优点在于计算设置费和连接费都比较简单。因此本文采用这种方法,即种子客户集合由全部客户组成($m=n$)。基于这样的种子集合,可以确定相应的设置费为 $v_j=2d_j$,连接费取为 $c_{ij}=d_i+d_{ij}-d_j,i=1,\cdots,n;j=1,\cdots m$。

2.3　TS 算法介绍

TS 算法是一种智能型启发式算法,是局部搜索算法的扩展。其主要过程是在合理选择的邻域结构内寻优,并标记已得到的局部最优解,在下次迭代过程中避开这些局部最优解,逐步向全局最优解靠近,直到达到停止规则。在设计应用 TS 算法时,该算法的一些主要参数,如邻域结构、禁忌对象和特赦规则等都是必须要认真考虑的,下面分别介绍。

2.3.1　*初始解生成和邻域结构构造*

对于本文研究的问题,通过对变形 CCLP 问题求解所得到的解直观上说是形成对客户的分组,每组由一辆车来负责配送,在应用 TS 算法时,将这个解作为初始解。

TS 算法的邻域一般采用 2-opt*,3-opt,CROSS-opt[6] 来构造。这里采用序列插入方法,即在计算开始时,令 s 为当前解,q 为初始解的客户分组数。在当前解 s 当中随机选择 $\min(n,5q)$ 个顶点,并依次将它们插入到包含这些随机点当中五个最邻近顶点之一的路径中。这里的路径也就是一辆车所服务的客户序列。然后,要对新构造的解 s' 进行判断,看它是否是 s 邻域内的最好解。为此,令 $f_1(\bar{s}^*)$ 为 s 邻域中的最好目标值,$\bar{s}^*$ 为相应的解,$f_1(s_i')$ 为在邻域中搜索到第 i 步时解 s_i' 的相应目标值。当迭代到第 i 步时($i\leqslant\min(n,5q)$),如果 $f_1(s_i')<f_1(\bar{s}^*)$,令 $f_1(\bar{s}^*)=f_1(s_i'),\bar{s}^*=s_i'$。在每次迭代过程中,未被禁忌的点将参与计算。

2.3.2　*禁忌对象、禁忌长度、特赦规则和停止规则*

本算法中的禁忌对象是点的移动,当点 v 从路径 r 中删除并被插入到 r' 当中后,在禁忌长度内,将禁止它被再次移回 r 中。禁忌长度在[5,10]间随机取一整数。特赦规则为当解禁 v 后,会得到一个改进较多的目标值。停止规则为当算法迭代到 40 次时,停止运算。

3　算法实现

3.1　算法中一些关键组件介绍

首先介绍对随机需求的处理。根据概率分布,可以确定在满足缺货概率条件下对每个客户的最大配送周期 $T_i^{\max}$,进而可以令 $e_i=C_i/T_i^{\max}$ 表示对零部件 i 的确定需求,并将之应用到下面的算法实现中。虽然这种变换不能完全反映问题特性,但可以避免设计和求解复杂的随机数学模型。

其次,求解 CCLP 时涉及到拉格朗日乘子的变化,因为它是求解 CCLP 算法的主要构件。当将 CCLP 原问题松弛变化到 m 个变形 0 - 1 背包问题时,要用次梯度方法来迭代改进解的质量[7],这个改进过程主要就是根据约束条件和目标值来控制拉格朗日乘子的变化。比如,当乘子向量已经迭代到 $\lambda^{(k)}$($k\geqslant 1$),下一次再解 m 个背包问题时,乘子按下式取值:

$$\lambda^{(k+1)}=\lambda^{(k)}+t_k\left(\sum_{j=1}^{m}\bar{x}_{ij}^{(k)}-1\right) \qquad \forall i \tag{4}$$

式中 t_k 为步长,其取值为:$t_k=\alpha\dfrac{(\bar{Z}-Z_{\lambda(k)})}{\sum_{i=1}^{n}\left(\sum_{j=1}^{m}\bar{x}_{ij}^{(k)}-1\right)^2}$

式中 α 为规模系数,$\bar{Z}$ 为目标值上界。

最后,由于在求解 CCLP 时要将原问题松弛变化为多个背包问题,松弛掉了每个终端只能与一个集中器连接这个约束条件,因此在乘子取值确定后所求得的解未必是可行解,为了构造可行解,在算法设计时采用了一个“贪婪”过程。这个过程主要是依据两个原则:

1)用于配送作业的车辆数应该尽量少;

2)一个客户被分到某个组时(相当于 CCLP 中一个终端被连接到某个集中器),就可以求出这个组的最小费用,当将这个客户依次分到所有的组中,就会得到一个最小费用的集合。如将这个集合中最小值与次最小值的差值称为节约费用,则最先被连接的客户应该是节约费用最大的客户。

3.2　算法实现步骤

通过上面的分析,可以详细地描述算法实现的总过程:

步骤 1:针对每个客户,计算 $T_i^{\max},e_i$;

步骤 2:问题转化。分别确定 v_j、$c_{ij}(i=1,\cdots,n;j=1,\cdots,m)$;

步骤 3:给出最大迭代次数 N;最优目标值 f^*(一足够大的数);规模系数 α;$k:=0$;$iter:=0$;初始乘子向量 $\lambda^{(k)}$;

步骤 4:令 $\bar{c}_{ij}=c_{ij}+\lambda_i^{(k)}$,求解 m 个背包问题,同时根据每个背包问题的目标值与 v_j 的差值确定是否该在位置 j 处设置集中器,这个差值越小表示这个位置越好;

步骤 5:如果步骤 4 所得的解满足约束条件式(1),应用

TS 算法对之进行改进，转步骤 7；否则，对之按升序排列，用“贪婪”过程构造一个对应于 $\lambda^{(k)}$ 的可行解，并应用 TS 算法对此可行解进行改进，同时计算相应的目标值 $f^{(k)}$，如 $f^{(k)} < f^*$，则 $f^* := f^{(k)}$；

步骤 6：如 $iter < N$，则 $iter := iter + 1$，$\alpha = \frac{\alpha}{2}$，按式(4)计算 $\lambda^{(k+1)}$，$k := k + 1$，转步骤 4；

步骤 7：转化 CCLP 的解为所研究问题的解，即如果在 CCLP 中 $y_j = 1$(对于任意的 j)，假设用 v_i 表示客户 i，应有 $S_j = Y_{1 \leqslant i \leqslant n} v_i$ if $x_{ij} = 1$，对于一组非空的 $\{S_j\}$，可以求得目标值 $f^{(k)}$，如 $f^{(k)} < f^*$，则 $f^* := f^{(k)}$；

步骤 8：求出最优目标值 f^* 对应的对每组客户(由一辆车来送货)的送货时间间隔(周期)、送货量和行驶路线。

4 算例分析

用 VC++ 5.0 编程来验证所设计的算法。程序设计时采用素数模数剩同余数法产生所需的随机数；采用 VC++ 基本类库(MFC)中提供的指针链表类 CPtrList 中的成员函数来处理点的各种运动(实际上是点的连接和归属关系的变化)，每个客户相当于链表中一个节点；用数组类 CUIntArray 来存储动态变化的每辆车所服务的客户序列。关于这两个类的介绍可参见文献[8]；采用结构体(struct)来表示每个客户的属性，如客户的需求、坐标和编号等。

在按照前面的步骤具体编程实现时，采用如下一组数据：取均匀分布于[0,100]平方公里内的 50 个随机点形成模拟客户集合，取所有客户每个周期可以接受的最大缺货概率 p 相同且等于 0.02，客户的最大存储能力 C 在 30～50 之间随机取值。送货车辆的载重量取为 200；配送时间间隔的最小值 T^{LB} 为 2；中央仓库的位置取在(50,50)这点。对于在计算拉格朗日乘子向量变化的步长时(见式(4))用到的目标值上界 $\overline{Z}$，取其等于第一次迭代时用“贪婪”过程得到的目标值加上一个常数，这个常数取值过小会使算法收敛速度很慢，取值过大又可能收敛于局部最小值，通过不同取值的比较最终令其等于 1000。取单位商品单位时间保管费 $h = 1.0$ 和每辆车执行任务时产生的固定费用 $F = 20$。

考虑到启发式算法仅是寻求系统的近似最优解，要综合平衡目标值和计算时间的关系。对于本文所设计的算法，影响算法性能的一个关键因素是搜索的步长，如果步长过小可能使算法仅获得局部最优解，但如果步长过长，又会使得计算时间增加和错过全局最优解。基于上面的数据，首先测试了规模系数 α 取值不同时目标值的变化(见图 1)，可见当 $\alpha = 250$ 时，系统的目标值是所有目标值中最优的。下面的算法分析中取 $\alpha = 250$ 来计算步长。

因为在搜索最优解的过程中主要是通过控制拉格朗日乘子向量的变化来逐步改善当前解，同时针对每个乘子向量构造一个可行解，所以决定乘子向量变化的最大迭代次数 N 直接影响程序的运行时间和所得到的最终结果。表 1 是 N 取值从 1 到 20 时所对应的运行时间和目标值，为了便于分析，同时给出了这个变化的关系图(见图 2)。从图中可以看出，当 $N = 11$ 时目标值已经达到了最小，为了验证算法的收敛性，我们逐步增大迭代次数，希望通过对不同种子客户集合的选取来改善目标值，可是即便将迭代次数增加到 20 次，目标值也没有发生变化，但计算时间却从 6s 增加到了 12s。我们又进一步测试了 N 等于 30、40 和 50 时的情况，结果是目标值与 $N = 11$ 时相同，计算时间线性增加，如 $N = 50$ 时的计算时间为 31s。

表 1 迭代次数不同时目标值和运行时间的变化

迭代次数	目标值(元)	运行时间(s)	迭代次数	目标值(元)	运行时间(s)
1	759.46	0.5	11	533.57	6
2	759.46	1	12	533.57	6
3	759.46	1	13	533.57	7
4	579.14	3	14	533.57	7
5	579.14	3	15	533.57	9
6	579.14	4	16	533.57	9
7	579.14	4	17	533.57	10
8	579.14	5	18	533.57	10
9	579.14	5	19	533.57	11
10	579.14	5	20	533.57	12

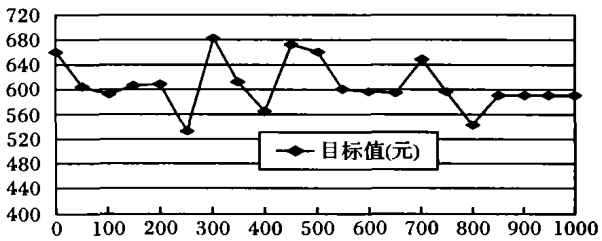

图 1 规模系数取值不同时的目标值变化

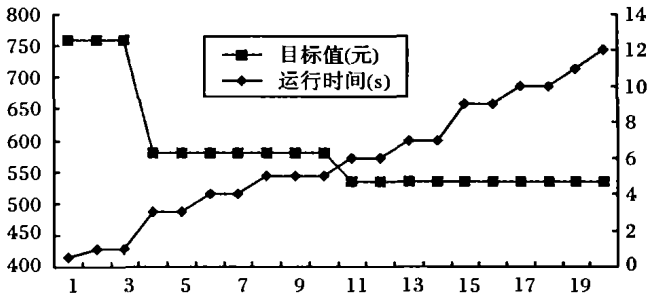

图 2 迭代次数与目标值和运行时间关系图

表 2 两个具体问题的详细运行结果

算法编号	目标值	路线数	路线编号	路线周期	单位时间配送量	客户序列
1	793.32	5	1	4.45	14.10	24 46 6 2 25 20
			2	2.70	73.51	36 5 33 40 31 49 30 8 48 14 17 19 3 45 15
			3	3.51	54.80	34 28 27 22 39 13 50 9 44
			4	3.12	53.79	21 35 41 10 7 4 1 37 18
			5	2.56	67.00	32 42 11 23 12 38 43 6 47 26 29
2	533.57	4	1	2.45	79.99	29 20 36 43 5 49 31 44 30 8 38 26
			2	3.40	57.48	6 33 40 48 9 50 13 39 22 2 15 25
			3	3.13	63.36	24 46 16 45 3 19 14 17 27 10 35 1 37 18 23 11 42
			4	2.83	62.38	47 21 34 28 41 7 4 12 32

为了分析使用 TS 算法对系统目标值的影响,基于同一组数据,分别进行了计算。令问题 1 表示未使用 TS 算法时的系统近似最优解,问题 2 表示应用 TS 算法后的近似最优解。在表 2 中详细给出了每个问题中近似最优解的目标值,客户的分组情况,对每组客户的送货周期和单位时间送货量,以及具体的车辆行驶路线(客户序列),当然每条路线的始终点都是中央仓库(程序中用 0 表示),下表中省略了两个端点。需要指出的是这里列出的车辆行驶路线未必是具体运作时所采纳的,因为本文讨论的是战术层的问题,给出这个具体解只是表明在实际管理中如果采用这个规划方案会使系统的运营费用最小。在表 2 中,为了反映程序的实际运行结果,没有对所求得的对每组客户的配送周期进行取整处理,物流系统实际运作时可以简单的对这组值取整(即四舍五入)得到便于操作的整数周期,当然也可以作些复杂的处理,比如采用 Power-of-Two[9] 策略等。

为了更清楚观察客户分布情况,图 3 和图 4 分别给出了这两个问题的图形输出。

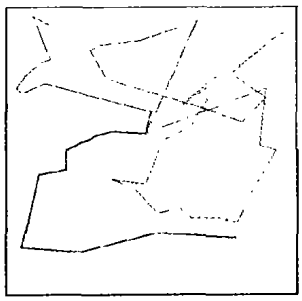

图 3　未应用 TS 算法结果

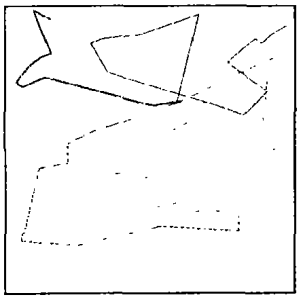

图 4　应用 TS 算法优化结果

4　结语

由于库存控制和运输优化是物流系统管理中最重要的两个环节,将这两个环节集成到一起来研究无疑比单独考虑其中的一个环节更有益于提高物流系统的整体性能,这已经得到了认同,但只有存在优良的算法才会真正促进这个思想在实践中的应用。为此,本文给出了基于 CCLP 问题和禁忌搜索算法所设计的算法,详细介绍了算法的设计过程,并就程序设计中一些要点进行了说明。通过数值模拟,验证了算法的可行性。

参考文献:

[1] LAPORTE G. The Vehicle Routing Problem: an Overview of Exact and Approximate Algorithms[J]. European Journal of Operatioal Research, 1992, 59: 345 – 358.

[2] ANILY S, FEDERGRUEN A. One warehouse multiple retailer systems with vehicle routing costs[J]. Management Science, 1990, 36 (1): 92 – 114.

[3] CHAKRAVARTY AK, ORLIN JB, ROTHBLUM UG. Consecutive optimizers for a partitioning problem with applications to optimal inventory groupings for joint replenishment[J]. Operations Research, 1985, 33: 820 – 834.

[4] BRAMEL J, SIMCHI-LEVI D. A location based heuristic for general routing problems[J]. Operations Research, 1995, 43: 649 – 660.

[5] PIRKUL H. Efficient algorithms for the Capacitated Concentrator Location Problem[J]. Computers and Operations Research, 1987, 14 (3): 197 – 208.

[6] TAILLARD DÉ, BADEAU P, GENDREAU M, *et al*. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows [J]. Transportation Science, 1997, 31(2): 170 – 186.

[7] HELD M, WOLFE P, CROWDER HP. Validation of subgradient optimization[J]. Mathematical Programming, 1974, 23(6): 62 – 88.

[8] KRUGLINSKI DJ, *et al*. 希望图书创作室译. Visual C++ 6.0 技术内幕[M]. 北京: 北京希望电子出版社, 1999.

[9] HAHM J, YANO CA. The economic lot and delivery scheduling problem: power of two policies[J]. Transportation Science, 1995, 29: 222 – 241.

(上接第 961 页)

($Ends(M_2)$ = 20) ≤ ($LCT(M_2)$ = 20); ($Ends(M_3)$ = 29) ≤ ($LCT(M_2)$ = 29),按照 2.2.4 节中(1) 比较得出此调度就是 k-FT 优化调度。

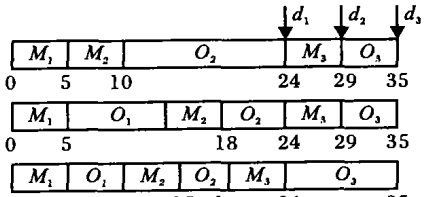

图 2　三种情况的 2-FT 调度

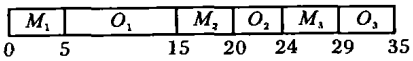

图 3　2-FT 优化调度

4　结语

本文陈述了实时应用中实时性和精确性平衡的必要性,阐述了 $LCT()$ 和 $lct()$ 函数的计算方法,探讨了在 k 错误模式及时间限制条件下的 k-FT 优化调度的属性、产生规则、产生过程以及调度是否为 k-FT 优化的判定方法,分析了此模式下非精确任务的容错优化调度的产生方法,在此基础上提出了 k 错误模式的容错优化调度的形式算法,并通过实例给出了具体分析过程。然而需要指出的是,本文采用的回报函数只是线性关系,在 k-FT 优化算法中,针对的也只是非抢断条件下的独立任务而没考虑满足继承限制条件任务的 k-FT 优化调度。因此,对满足继承限制条件任务的 k-FT 优化调度需要进一步研究。

参考文献:

[1] AYDIN H, MELHEM R, MOSSE D. Incorporating Error Recovery into the Imprecise Computation Model[A]. Sixth International Conference on Real – Time Computing Systems and Applications[C]. Hong Kong, China, 1999.

[2] AYDIN H, MELHEM R, MOSSE D. Tolerating Faults While Maximizing Reward[A]. Euromicro Conference on Real Time Systems (ECRTS00)[C]. Stockholm, Sweden, 2000.

[3] AYDIN H, MELHEM R, MOSSE D. Optimal Scheduling of Imprecise Computation Tasks in the Presence of Multiple Faults[A]. Proceedings of the Seventh International Conference on Real-Time Systems and Applications[C]. 2000. 1530 – 1427.

[4] CHEN Y, XIONG GZ. Imprecise Computation Fault-Tolerant Rate-Monotonic Scheduling[A]. Proceedings of the 5th International Conference on Algorithm and Architecture for Parallel Processing 0-7695-1512-6[C]. 2002.

[5] AYDIN H, MELHEM R, MOSSE D, *et al*. Optimal Reward-Based Scheduling of Periodic Real-Time Tasks[J]. IEEE Transactions on Computers, 2001, 50(2).

[6] SON HS, THURESSON M, HANSSON J. Imprecise Task Scheduling and Overload Management Using OR-ULD[A]. Proceedings of 7th International Conference on Real-Time Systems and Applications [C]. 2000. 1530 – 1427.