文章编号: 1001-0920(2011)02-0271-05

数据仓库ETL任务调度模型研究

宋旭东[1,2], 刘晓冰[2]

(1. 大连交通大学 软件学院，大连 116028；2. 大连理工大学 CIMS中心，大连 116024)

摘 要: 数据仓库系统包含众多的抽取-转换-加载(ETL)任务, 这些任务具有一定的优先约束关系. 在多处理机环境下, 如何高效地调度这些ETL任务是构建数据仓库需要研究的重要问题. 对此, 在对数据仓库ETL任务调度规则进行归纳总结的基础上, 以数据仓库总的ETL完成时间最短为目标, 建立了数据仓库ETL任务调度模型. 同时结合问题的特点, 采用同层划分的思想, 提出基于同层划分遗传算法求解问题的方法. 最后, 通过应用实例验证了所建立模型和求解算法的可行性和有效性.

关键词: 数据仓库；抽取-转换-加载；任务调度；同层划分；遗传算法

中图分类号: TP273 **文献标识码:** A

Study on ETL tasks scheduling model in data warehouse

SONG Xu-dong[1,2], *LIU Xiao-bing*[2]

(1. Software Institute，Dalian Jiaotong University，Dalian 116028，China；2. CIMS Center，Dalian University of Technology，Dalian 116024，China. Correspondent：SONG Xu-dong，E-mail：xudongsong@126.com)

Abstract: Data warehouse system includes many extract-transform-load(ETL) tasks which have some precedence constraint relations. In a multi-processor environment, how to efficiently schedule these ETL tasks is one of the important aspects for constructing data warehouse. On the basis of classification and summary of data warehouse ETL scheduling rules, a data warehouse ETL scheduling model is established to minimize the total ETL execution time. At the same time, based on the characteristics of the problem, adopting the same layer division strategy, a genetic algorithm based on the same layer division is proposed. Finally, the application case of the model is represented, and the case results show the feasibility and effectiveness of this model and its algorithm.

Key words: data warehouse；ETL；task scheduling；same layer division；genetic algorithm

1 引 言

在数据仓库的建立过程中, 核心技术是抽取、转换、装载(ETL), 它为数据仓库提供及时、高质而准确的数据. 由于ETL包括众多的处理任务, 且这些处理任务之间有一定的约束关系, 如何高效地调度和管理这些处理任务是数据仓库ETL实施中非常重要的工作, 也是提高数据仓库开发效率和资源利用率的关键.

关于任务分配与调度问题被认为是NP完全问题[1], 不可能在多项式时间内找到问题的最优解. 近些年, 学者们提出了基于遗传算法的任务分配与调度[2-3], 为求解此类NP完全问题提供了新的途径. 然而, 这些算法基本上都是针对一个任务的无回路图(DAG)的调度, 针对数据仓库ETL多任务调度的研究并不多见. 文献[4]提出了数据仓库系统中任务调度框架, 给出了动态调度、静态调度和同层划分等3种调度策略, 但缺少对调度模型的数学表示, 没有给出求解算法的机理描述; [5]给出了一种基于贪婪算法ETL最优任务调度方法, 但该方法调度的粒度只能限于ETL单个任务, 无法精细到任务中包含的多个操作; [6]提出了ETL过程的“主表衍生”模式, 给出了ETL执行的流水线优化方法, 但该方法是以ETL各活动串行约束为前提, 缺少一定通用性. 其他相关研究还有ETL执行过程优化[7-9], 这些研究工作主要着眼于ETL工作流逻辑转换的过程优化, 通过减少处理过程的数量或改变过程的执行顺序来减少ETL工作流的执行代价, 并没有对ETL活动的分配和调度问题展开深入研究. 为此, 本文针对数据仓库ETL任务调度问题, 建立了数据仓库ETL任务调度模型, 同时

收稿日期: 2009-11-27；**修回日期:** 2010-04-17.

基金项目: 国家自然科学基金项目(70572098)；辽宁省教育厅项目(L2010083)

作者简介: 宋旭东(1969−), 男, 副教授, 博士, 从事数据仓库、决策支持系统等研究；刘晓冰(1956−), 男, 教授, 博士生导师, 从事企业信息化等研究.

结合遗传算法及同层划分的思想, 提出了相应的模型求解算法.

2 数据仓库ETL任务调度问题描述

数据仓库ETL过程通常包含若干独立的ETL任务, 每个ETL任务又由多个有时间顺序的具体ETL操作(如数据抽取、数据清洗、数据变换、数据聚集、数据集成和数据加载等)组成. 不同ETL任务之间, 没有强制性的时间顺序, 可并发执行. 但每个ETL任务的各个具体ETL操作应该按照各自的先后次序约束执行, 没有先后约束的ETL操作可并发执行. 为了提高数据仓库ETL执行效率, 需要对数据仓库ETL所有任务进行合理的分配与调度. 本文的数据仓库ETL任务调度问题满足以下假设: 1) 所有ETL操作一旦开始进行就不能中断; 2) 所有处理机都是相同的, 即每个ETL操作均可在任意处理机上执行, 而且执行时间是相同的.

其调度目标是: 在满足处理机资源约束及ETL各操作先后次序约束条件下, 合理地将多个ETL任务及ETL操作分配到多个处理机上, 并合理调度各操作执行顺序, 使ETL所有任务尽可能地并行执行以使总的ETL完成时间最短.

3 数据仓库ETL任务调度模型

3.1 ETL调度模型建立

为了清晰表示数据仓库ETL任务调度数学模型, 首先给出一些相关定义及表示.

定义 1 数据仓库ETL:

ETL= $\{\mathrm{Task}_1, \mathrm{Task}_2, \cdots, \mathrm{Task}_n\}$, 它由多个相互独立的ETL任务构成.

定义 2 ETL任务:

$\mathrm{Task}_i = (V_i, R_i)(i = 1, 2, \cdots, n)$ 表示第 i 个ETL任务, 它由多个相互关联的ETL操作组成. 其中 $V_i = \{O_{i1}, O_{i2}, \cdots, O_i m_i\}$ 表示每个ETL任务由多个ETL操作组成; $R_i = \{P(O_{ij}, O_{ik}) | j, k \in (1, 2, \cdots, m), j \neq k\}$ 表示ETL操作之间的先后顺序关系.

定义 3 ETL操作:

$O_{ij}(i = 1, 2, \cdots, n, j = 1, 2, \cdots, m_i)$ 表示第 i 个ETL任务中的第 j 个操作.

定义 4 ETL操作关系:

$$P(O_{ij}, O_{ik}) = \begin{cases} 1, \ O_{ij} \text{是} O_{ik} \text{的前驱操作;} \\ 0, \ O_{ij} \text{不是} O_{ik} \text{的前驱操作.} \end{cases}$$

它表示两个ETL操作的相互关系, 若 O_{ij} 是 O_{ik} 的前驱操作, 则在 O_{ij} 操作完成之前, O_{ik} 操作不能提前执行.

定义 5 处理机资源:

CPU= $\{C_1, C_2, \cdots, C_l\}$ 表示可调度的处理机集合, C_i 为第 i 个处理机.

定义 6 调度决策变量:

$$x_{ij}^k = \begin{cases} 1, \ O_{ij} \text{分配到第} k \text{个处理机上执行;} \\ 0, \ O_{ij} \text{没分到第} k \text{个处理机上执行.} \end{cases}$$

定义 7 ETL操作执行时间:

$T(O_{ij}^k) = t_e(O_{ij}^k) - t_s(O_{ij}^k)$ 表示第 i 个任务的第 j 个操作 O_{ij} 在第 k 个处理机上的执行时间. 其中: $t_s(O_{ij}^k)$ 表示 O_{ij} 操作的执行开始时间, $t_e(O_{ij}^k)$ 表示 O_{ij} 操作的执行结束时间. 这里, 假设所有处理机性能相同, 即不同处理机完成同一个ETL操作的时间相同, 因此 $T(O_{ij}^k)$ 也可简写为 $T(O_{ij})$.

定义 8 ETL操作执行关系:

$$Q(O_{ij}^k, O_{i'j'}^k) = \begin{cases} 1, \ O_{ij} \text{先于} O_{i'j'} \text{在} k \text{上执行;} \\ 0, \ O_{ij} \text{不先于} O_{i'j'} \text{在} k \text{上执行.} \end{cases}$$

上式表示两个不同ETL操作在同一处理机上执行的先后关系. 若 $Q(O_{ij}^k, O_{i'j'}^k) = 1$, 则表示在处理机 k 上, O_{ij} 的开始执行时间要早于 $O_{i'j'}$ 的开始执行时间, 即 $t_s(O_{ij}^k) - t_s(O_{i'j'}^k) < 0$.

定义 9 处理机完成时间:

$T_{\mathrm{ts}}(C_i)$ 表示在某一调度方案(ts)下, 处理机 C_i 按ETL操作约束关系顺序完成指派到它上的所有ETL操作所要花费的总时间.

根据数据仓库ETL任务调度目标及约束规则, 建立如下调度数学模型:

$$\min T(\mathrm{ts}) = \max T_{\mathrm{ts}}(C_i), \ \forall i(1 \leqslant i \leqslant l), \ \forall \mathrm{ts}. \quad (1)$$

s.t.

$$\sum_{k=1}^{l} \sum_{i=1}^{n} \sum_{j=1}^{m_i} x_{ij}^k = m_1 + m_2 + \cdots + m_n,$$

$$Q(O_{ij}^k, O_{i'j'}^k) = 1 \rightarrow t_s(O_{i'j'}^k) - t_e(O_{ij}^k) \geqslant 0,$$

$$\forall k(1 \leqslant k \leqslant l), \ \neg \exists i = i' \wedge j = j';$$

$$x_{ij}^k x_{ij}^{k'} = 0,$$

$$\forall j(1 \leqslant j \leqslant m_i), \ k \neq k', \ k, k' \in (1, 2, \cdots, l);$$

$$P(O_{ij}, O_{ij'}) = 1 \rightarrow t_s(O_{ij'}^{k'}) - t_e(O_{ij}^k) \geqslant 0,$$

$$\forall i(1 \leqslant j \leqslant n), \ j \neq j', \ j, \ j' \in (1, 2, \cdots, m_i);$$

$$x_{ij}^k = 1 \text{ or } 0.$$

3.2 ETL调度模型求解

遗传算法具有在复杂空间求解问题近似解的能力, 但对于本文中的问题, 应用传统遗传算法随机产生的个体可能会产生违背ETL操作优先约束的解, 即产生不可行解. 为此, 本文借鉴同层划分的思想[4], 采用基于同层划分的遗传算法进行模型求解.

3.2.1 ETL操作同层划分表示

按照广度优先搜索策略, 将ETL所有操作划分

为不同的层次, 没有前驱节点的层次为第1层, 该层的所有ETL操作集记为

$$O(1)=\{O_{ij}|\neg\exists P(O_{ij'},O_{ij})=1,\forall i,j\}.$$

在得到第1层ETL操作集$O(1)$后, 逻辑上将它们从ETL任务图中删除, 得到剩余图$(\{O_{ij}\}-O(1))$, 在剩余图中再确定所有没有前驱的节点, 形成第2层ETL操作集$O(2)$. 这样反复操作, 直到剩余图为空, 最后得到$O(1),O(2),O(3),\cdots,O(h)$, 即ETL同层划分操作集.

3.2.2　染色体编码及适应度函数表示

采用数字串编码方式进行染色体编码, 每个数字串表示一个可能的调度, 它由两部分构成, 第1部分是一个由ETL操作序号组成的子串, 表示ETL操作调度顺序, 称为操作调度子串; 第2部分是一个由处理机序号组成的子串, 表示每个ETL操作所分配的处理机, 称为处理机子串.

定义适应度函数为

$$f(\mathrm{ts})=\sum_i\sum_j(T(O_{ij})-T(\mathrm{ts})). \tag{2}$$

$f(\mathrm{ts})$值越大, 则该染色体对应的调度效果越好.

3.2.3　算法流程

基于同层划分的ETL任务调度求解流程如图1所示.

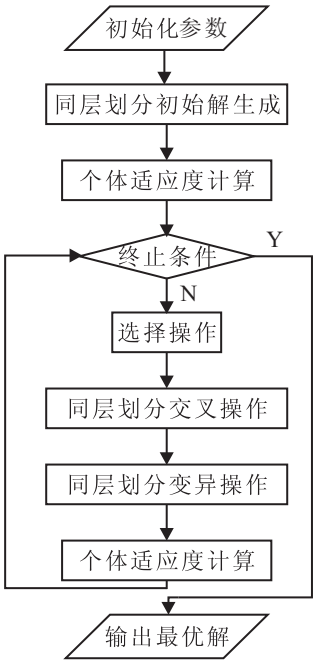

图 1　ETL任务调度求解流程

1) 生成初始种群

Step 1: 按ETL操作层次(即升序), 对每个ETL操作集$O(k)$进行Step 2.

Step 2: 将$O(k)$中的每个ETL操作随机地分配给l个处理机之一.

Step 3: 将所有ETL操作按分配给处理机的先后顺序排列, 形成一个合理调度.

Step 4: 重复Step 1∼Step 3共N次, 得到由N个染色体组成的初始种群.

2) 选择操作

采用基于轮盘赌的比例选择法, 具体如下:

Step 1: 对应每个染色体, 计算被选择的概率

$$p_i=f_i\Big/\Big(\sum_{i=1}^{N}f_i\Big), \tag{3}$$

其中N为种群规模.

Step 2: 在(0, 1]区间产生一个随机数.

Step 3: 若

$$p_0+p_1+\cdots+p_{k-1}<r\leqslant p_0+p_1+\cdots+p_k,$$

则选择第k个染色体.

Step 4: 重复Step 2和Step 3共N次, 得到由N个染色体组成的新一代种群.

3) 交叉操作

设计交叉操作时必须满足ETL操作优先约束关系, 本文构建了两种交叉算子, 一种为DCX, 用于两个父代染色体交叉操作; 另一种为SCX, 用于自身染色体交叉操作.

1) DCX交叉算子操作

Step 1: 选择适应度较高的两个父代染色体, 记为ts_1和ts_2.

Step 2: 随机生成一个整数$x(1\leqslant x\leqslant h$, h为ETL任务的最大层数), 作为ETL操作DCX交叉操作的交叉层, 与x层数相同的所有ETL操作构成待交换的操作基因串. 随机生成一个$y(1\leqslant y\leqslant n$, n为所有ETL操作的个数), 作为处理机交叉操作的交叉位.

Step 3: 对于父代染色体, 在与x层数相同的操作基因串中进行交换操作; 在y交叉位进行处理机单点交叉操作, 生成两个子染色体. 由于此操作并未改变ETL操作的前驱限制关系, 因此可以证明新产生的染色体是两个合理调度.

2) SCX交叉算子操作

Step 1: 选择适应度较高的某个父代染色体, 记为ts_1.

Step 2: 随机生成一个整数$x(1\leqslant x\leqslant h$, h为ETL任务的最大层数), 作为ETL操作SCX交叉操作的交叉层, 设该层ETL操作数为z. 产生两个不同随机数y_1和y_2 $(1\leqslant y_1,y_2\leqslant z)$, 作为两个同层ETL操作交叉点. 产生两个不同随机数w_1和w_2 $(1\leqslant w_1,w_2\leqslant n$, n为所有ETL操作的个数), 作为处理机交叉操作交叉点.

Step 3: 对于父代染色体, 在x层对y_1和y_2两个

基因进行交换操作; 处理机在w_1和w_2进行交叉操作, 生成子染色体. 由于交换两个具有相同高度的操作, 并未改变ETL操作的前驱限制关系, 因此可以证明新产生的染色体是一个合理调度.

3) 变异操作

本文的变异算子SMX是将ETL操作从负载较重的处理机迁移到负载较轻的处理机, 进而提高多处理机的并行性, 并且不破坏ETL操作间的约束关系. 具体变异算子SMX操作如下:

Step 1: 选择适应度较高的某个父代染色体, 记为ts_1.

Step 2: 随机生成一个整数$x(1\leqslant x \leqslant h, h$为ETL任务的最大层数), 作为ETL操作SMX操作的变异层.

Step 3: 计算该变异层每个ETL操作所在的处理机负载. 处理机负载可用分配给该处理机的所有ETL操作执行时间的和来表示. 得到负载最大和负载最小的两个处理机l_i和l_j.

Step 4: 在x层, 针对处理机负载最大的ETL操作, 实施变异操作, 将其对应的处理机由l_i变异为l_j, 生成新染色体. 变异操作并未改变ETL操作的前驱限制关系, 因此可证明新产生的染色体是一个合理调度.

3.2.4 算法有效性检验和算法复杂度分析

将数据仓库ETL调度问题用于直接比较的实例并不多见. 为了检验算法的有效性, 本文采用文献[10]中提到的例子, 考虑一个具有15个任务, 在3个处理机上进行处理的调度问题, 表1给出了不同算法的性能比较. 由表1可以看出, 本文算法和[10]的算法均可达到最优结果, 但[10]的算法仅能处理具有一个父结点的任务调度问题, 而本文算法不受此限制, 可适应各种情况下的数据仓库ETL任务调度问题.

表 1 各种算法性能比较

	启发式算法	文献[10]算法	本文算法	最佳结果
执行代价	44	35	35	35

本文算法采用多次迭代实现模型求解, 设最大迭代次数为Gen, 种群规模为N, ETL操作个数为T. 每次迭代过程中, 选择操作的时间复杂度为$O(N)$, 交叉操作的时间复杂度为$O(N)$, 变异操作的时间复杂度为$O(N)$, 个体适应度计算的时间复杂度为$O(N*T)$. 因此, 本文算法时间复杂度为$O(\mathrm{GEN}*N*T)$. 算法采用数字串编码方式进行染色体编码, 每个ETL操作对应数字串中的一位, 因此, 本算法空间复杂度为$O(N*T)$.

4 应用实例

本课题源自国内一家大型钢铁企业集团的数据仓库系统构建实际问题, 引入数据仓库ETL任务调度模型进行实例验证. 该钢铁企业集团下设3个股份公司(基地), 在企业集团数据仓库构建过程中, 为支持多个决策主题, 需要定义多个ETL任务, 负责从3个基地抽取数据, 转换并加载到数据仓库中, 形成数据仓库事实表和维表. 以2008年度集团合同跟踪事实表为例, 需要定义订单抽取、各基地入库抽取、各基地发货抽取、各基地抽取数据转换、数据汇总、数据连接、数据加载等13个ETL操作. 实例中本文构建了5个ETL任务, 共计43个操作. 图2给出了这些ETL操作执行的先后次序, 图中圆圈中的数字表示操作的执行时间. 现将这5个ETL任务, 43个ETL操作, 分配在3个性能相同的处理机上进行ETL任务调度.

应用同层划分遗传算法进行求解, 设迭代次数为Gen = 100, 种群规模$N = 5$, 交叉概率$p_c = 0.9$, 变异概率$p_m = 0.1$. 本实验环境为: CPU为Pentium

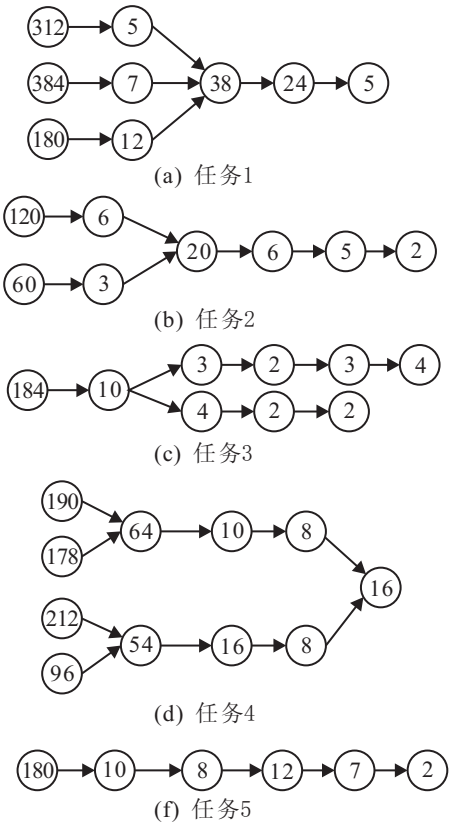

图 2 ETL任务构成实例

IV-3 GHz, 内存为1 GB, PC机操作系统为Windows 2000, 所有程序代码用Visual C++ 6.0编写, 得到最优调度下的$T(\mathrm{ts}) = 827$ s.

若增加处理机个数, 则可缩短ETL任务完成时间. 对上述5个ETL任务, 43个ETL操作, 分别在3~6个性能相同的处理机上进行ETL任务调度, 应用基于同层划分的遗传算法进行求解, 参数设置同上, 结果如表2所示.

由于本数据仓库ETL任务调度模型是针对一般

表 2　多个处理机下的ETL任务调度结果

处理机个数	最优调度结果	10 次平均调度结果
3	827	833
4	642	653
5	580	585
6	484	488

数据仓库的ETL调度问题进行设计的, 模型具有一定的通用性; 同时, 本文提出的基于同层划分的模型求解算法具有较好的适应性, 可满足多处理机ETL任务调度问题求解. 因此, 本文所提出的方法能适应一般数据仓库ETL任务调度应用.

5　结　论

数据仓库ETL任务调度是一个十分复杂的优化问题, 但对于该问题的研究还不多见. 本文以数据仓库总的ETL执行时间最短为调度目标, 建立了ETL任务调度模型, 提出了基于同层划分的遗传算法进行模型求解的算法流程, 并通过应用实例验证了模型及求解方法的可行性和有效性, 为数据仓库ETL任务调度问题的研究提供了宝贵经验.

参考文献(References)

[1] Hironori Kasahara, Seinosuke Narita. Practical multiprocessor scheduling algorithms for efficient parallel processing[J]. IEEE Trans on Computers, 1984, 33(11): 1023-1029.

[2] Edwin S H Hou, Nirwan Ansari. Genetic algorithm for multiprocessor scheduling[J]. IEEE Trans on Parallel and Distributed Systems, 1994, 5(2): 113-120.

[3] 钟求喜, 谢涛, 陈火旺. 基于遗传算法的任务分配与调度[J]. 计算机研究与发展, 2000, 37(10): 1197-1203.
(Zhong Q X, Xie T, Chen H W. Task matching and scheduling by using genetic algorithms[J]. J of Computer Research and Development, 2000, 37(10): 1197-1203.)

[4] 史捷, 鲍玉斌, 刘运涛, 等. 数据仓库系统中任务调度策略研究[J]. 控制与决策, 2005, 20(1): 109-112.
(Shi J, Bao Y B, Liu Y T, et al. On task scheduling strategy in data warehouse system[J]. Control and Decision, 2005, 20(1): 109-112.)

[5] 王珊, 陈琨. ETL中基于贪婪算法的任务调度方法研究[J]. 微电子学与计算机, 2009, 26(7): 130-133.
(Wang S, Chen K. Research on task scheduling method based on the greedy algorithm in ETL[J]. Micro Electronics and Computer, 2009, 26(7): 130-133.)

[6] 韩京宇, 徐立臻, 董逸生. ETL执行的流水线优化[J]. 小型微型计算机系统, 2005, 26(6): 134-138.
(Han J Y, Xu L Z, Dong Y S. Optimization of ETL execution by pipelining method[J]. Mini-micro System, 2005, 26(6): 134-138.)

[7] 吴远红. ETL执行过程的优化研究[J]. 计算机科学, 2007, 34(1): 81-83.
(Wu Y H. The research of optimizing ETL execution process[J]. Computer Science, 2007, 34(1): 81-83.)

[8] 姚全珠, 赵双瑞. 基于状态空间搜索的ETL过程优化[J]. 计算机工程与应用, 2007, 43(26): 169-173.
(Yao Q Z, Zhao S R. Optimizing ETL processes based on state-space search[J]. Computer Engineering and Applications, 2007, 43(26): 169-173.)

[9] Simitsis A, Vassiliadis P, Sellis T. Optimizing ETL processes in data warehouses[C]. Proc of 21st Int Conf on Data Engineering(ICDE). Tokyo, 2005: 564-575.

[10] 张炎, 裘聿皇. 基于遗传算法的考虑优先约束和负载平衡的多任务调度[J]. 计算机工程与应用, 2003, 39(12): 86-88.
(Zhang Y, Qiu Y H. Multi-task scheduling with precedence constraint and load balance based on genetic algorithm[J]. Computer Engineering and Applications, 2003, 39(12): 86-88.)