文章编号: 1001-0920 (2014) 11-1935-08 DOI: 10.13195/j.kzyjc.2013.0986

基于负载均衡的模糊概念并行构造算法

张 卓[1], 杜 鹃[2], 王黎明[1]

(1. 郑州大学 信息工程学院，郑州 450001；2. 黄河水利职业技术学院 信息工程系，河南 开封 475003)

摘 要: 提高模糊概念格直接构造效率是形式概念分析领域的主要问题之一, 而当前基于模糊伽罗瓦联系的闭包运算仍是构造模糊概念的主要计算负荷, 为此, 提出一种基于负载均衡的并行构造模糊概念算法. 该算法使用树状结构组织, 遍历由自然数区间简化的搜索空间, 逐级并行产生模糊概念、缩减搜索区间, 并通过重新划分子搜索空间, 实现各个计算节点负载均衡. 实验结果表明, 所提出的算法在稀疏数据集上表现优秀, 能够有效地提高模糊概念构造效率.

关键词: 模糊概念构造；自然数区间；完全树；负载均衡；并行算法

中图分类号: TP311 **文献标志码:** A

Load balance-based algorithm for parallelly generating fuzzy formal concepts

ZHANG Zhuo[1], *DU Juan*[2], *WANG Li-ming*[1]

(1. School of Information Engineering，Zhengzhou University，Zhengzhou 450001，China；2. Department of Information Engineering，Yellow River Conservancy Technical Institute，Kaifeng 475003，China. Correspondent: ZHANG Zhuo，E-mail：iezhangzhuo@zzu.edu.cn)

Abstract: Directly constructing the fuzzy concept lattice is one of most important issues for formal concept analysis(FCA). However, most construction algorithms for fuzzy concept lattice are based on closure operation of fuzzy Galois connection. Therefore, a parallel algorithm based on load balance is proposed to improve the efficiency of building all fuzzy concepts. It utilizes the structure of complete tree to organize and parallelly breadth-first traverse search space which is represented with nature number interval. Along the height of the complete tree, the algorithm checks and reduces current sub-search spaces parallelly, meanwhile fuzzy concepts are produced. At the end of each iterations, search space is redivided so that all computing nodes share computing load fairly. Experiment results show that the algorithm has excellent performances on sparse data set, and it can effectively improve the efficiency of the construction of all fuzzy concepts.

Key words: fuzzy formal concept construction；natural number interval；complete tree；load balance；parallel algorithm

0 引 言

概念格[1]作为有效的概念分析方法, 已广泛地应用于数据挖掘[2-3]、本体构造[4]、多属性决策[5]等诸多领域, 其中大多数基于形式概念分析的应用任务都是以概念构造为基础[2-5]. 目前, 人们提出了许多方法将传统形式概念扩展到模糊形式概念. 它们之间的主要区别在于模糊伽罗瓦联系定义的不同, 从而导致模糊形式概念定义、模糊概念格构造方法也各不相同. 这些方法大致可分为以下两类:

1) 阈值裁截法 (α-cut 方法). 该方法通过用户设定的阈值来确定模糊形式背景中哪些属性对对象产生影响. 这种方法的优点是能够处理模糊形式背景中的连续隶属度值情况. Tho 等[4]将该方法用于模糊本体构造; 刘宗田等[6]采用两个阈值确定模糊形式背景中属性与对象之间关系, 给出了相应的渐近式构造算法; 胡明涵等[7]关注于阈值裁截法的数学特性, 给出了计算模糊概念之间相似性的方法; 许佳卿等[8]则将阈值裁截法应用到软件系统的演化分析领域.

2) 对传统伽罗瓦联系进行模糊扩展的方法[9]. 该方法最初由 Burusco 等[10]提出. 范世青等[11]分别在交

收稿日期: 2013-07-17；**修回日期:** 2013-12-10.

基金项目: 国家青年科学基金项目(61303044)；中国博士后科学基金项目(2013M541993)；国家科技支撑计划项目(2013BAH23F01)；河南省博士后科研项目(2012015)；河南省教育厅科学技术研究重点项目(13A520414).

作者简介: 张卓(1978−), 男, 讲师, 博士, 从事形式概念分析及其应用的研究；王黎明(1963−), 男, 教授, 博士, 从事分布式人工智能等研究.

换伴随对与对合剩余格的条件下, 对模糊概念格的4种定义形式进行了讨论; 孙士保等[12]使用下半连续三角模生成的剩余蕴涵建立了模糊概念格理论. 目前, Belohlavek[13]所提出的L-模糊概念格理论, 因具有兼容经典概念格的数学特性而成为主流方法, 该方法有效地扩展了传统形式概念分析信息处理的能力.

目前, 基于Belohlavek所定义的模糊概念格直接构造算法[14-15]并不多. 文献[14]采用NextClosure思想[1]构造L-模糊概念, 但并没有产生格结构; 文献[15]借鉴Lindig算法[16]思想直接构造了L-模糊概念格. 由于概念格的固有属性使其构造过程具有指数级的时间复杂度, 为了提高L-模糊概念构造效率, 笔者在前期工作[17]中已经实现了均分模糊集合搜索空间的并行构造方法. 进一步研究发现: 算法[14-15, 17]都是基于模糊伽罗瓦联系闭包运算的批处理构造算法; 由于闭包计算需要多次扫描数据库, 面对大规模数据库时, 闭包计算耗时过长已成为影响构造效率的主要因素. 显然, 如果以闭包计算为最小计算粒度进行并行任务划分, 相对于均分模糊集合搜索空间, 更能够达到计算负载均衡的目的. 本文以此为出发点, 通过证明模糊集合组合空间、树状搜索空间、自然数区间同构, 进而使得本文所提出的基于负载均衡的模糊概念并行算法能够以自然数表示模糊集合, 以树状搜索空间组织遍历次序, 逐层搜索模糊概念; 并且在每层搜索开始前预计闭包计算次数, 然后均匀分配该计算负载, 从而在充分利用计算资源的基础上, 有效地提高模糊概念格构造效率. 作为基于模糊概念格应用任务的基础算法, 本文方法具有一定的通用性.

1 构造算法

本文算法按照一定次序计算模糊概念的外延或者内涵. 根据模糊伽罗瓦联系的对偶性, 可获得完整的模糊概念. 为了便于表述, 本文以计算模糊内涵的方式进行阐述.

1.1 搜索空间的表示

假设对于属性集合$Y=\{0,1,\cdots,n\}$, 真值集合$L=\{0=a_0<a_1<\cdots<a_k=1\}$, 则$Y$的幂集(大小为$\|L\|^{\|Y\|}$)为本文算法的搜索空间. 令属性之间存在序关系, 即$0<1<\cdots<n$, $n=\|Y\|-1$, 则属性模糊集合组合空间为

$$S_L^Y=\left\{\left(\frac{v_0}{y_0},\frac{v_1}{y_1},\cdots,\frac{v_n}{y_n}\right)\right\}. \tag{1}$$

其中: $v_j\in\{a_0,a_1,\cdots,a_k\}$, $0\leqslant j\leqslant n$. 它可以使用一颗$\|L\|$叉完全搜索树表示.

定义 1(树状搜索空间) 对于模糊集合组合空间S_L^Y, 其树状搜索空间为一棵高度为$\|Y\|$的$\|L\|$叉完全搜索树, 表示为T_L^Y; 根节点高度为0; 第i级高度树枝对应于属性y_i, 其中$0\leqslant i\leqslant n$; 其每个非叶子节点拥有$\|L\|$个分支, 第i级第j个分支可表示为b_i^j, 分支之间存在序关系, 即$b_i^0<b_i^1<\cdots<b_i^{\|L\|-1}$, 第$j$个分支拥有权重$a_j\in L$.

一棵高度为$\|Y\|$的$\|L\|$叉完全搜索树共有$\|L\|^{\|Y\|}$个叶子节点, 与模糊集合大小$\|L\|^{\|Y\|}$相同. 因此, 可以使用一条从树T_L^Y的根到叶子节点的路径表示一个模糊集合.

定义 2(路径) 由树状搜索空间T_L^Y的根节点到其叶子节点所经过的所有分支构成一条路径, 表示为$p=\{b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n}\}$, 其中$m_0,m_1,\cdots,m_n\in\{0,1,\cdots,\|L\|-1\}$. 树状搜索空间$T_L^Y$的所有路径表示为$P_L^Y$.

根据定义2, 一条路径的权重序列可以表示为一个模糊集合.

定义 3(函数θ) $P_L^Y\to S_L^Y$, 树状搜索空间的路径到模糊集合的映射, 其中$a_{m_0},a_{m_1},\cdots,a_{m_n}\in L$, 有

$$\theta((b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n}))=\left(\frac{a_{m_0}}{y_0},\frac{a_{m_1}}{y_1},\cdots,\frac{a_{m_n}}{y_n}\right). \tag{2}$$

定义 4(函数ω) $S_L^Y\to P_L^Y$, 由模糊集合的映射到树状搜索空间的路径, 即

$$\omega\left(\left(\frac{a_{m_0}}{y_0},\frac{a_{m_1}}{y_1},\cdots,\frac{a_{m_n}}{y_n}\right)\right)=(b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n}). \tag{3}$$

命题 1 树状搜索空间T_L^Y的所有路径P_L^Y与模糊集合组合空间S_L^Y同构.

证明 1) 根据函数θ的定义, 任意两条不同的路径, 它们不同的分支选择分别反映到模糊集合相应维度的向量中, 使得模糊集合也不同, 因此函数θ是单射的.

2) 一棵高度为$\|Y\|$的$\|L\|$叉完全搜索树共有$\|L\|^{\|Y\|}$个叶子节点, 故其共有$\|L\|^{\|Y\|}$条非相同路径; 而模糊集合组合空间S_L^Y的大小为$\|Y\|$的$\|L\|$, 故函数θ是满射的.

3) 根据1)和2), 函数θ是双射函数, 故树状搜索空间T_L^Y的所有路径P_L^Y与模糊集合组合空间S_L^Y之间同构, 即$P_L^Y\cong S_L^Y$. □

树状搜索空间有利于对本文所提出的负载计算和划分方法的理解. 但是在算法实际应用中不便于搜索空间的表示和子任务分配等工作, 因此, 本文进一步将一条路径的向量组合转换为一个自然数表示, 定义如下.

定义 5(函数ρ) $P_L^Y\to N$, 树状搜索空间路径P_L^Y到自然数N的映射

$$\rho(p)=m_0\times\|L\|^{\|Y\|-1}+m_1\times\|L\|^{\|Y\|-2}+\cdots+m_{n-1}\times\|L\|^1+m_n\times\|L\|^0,\quad(4)$$

其中 $p=(b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n})$.

定义 6 (函数 γ)　$N\to P_L^Y$, 自然数 N 到树状搜索空间路径 P_L^Y 的映射

$$\gamma(t)=(b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n}).\quad(5)$$

其中: $t\in N$; $m_i\,(0\leqslant i\leqslant n)$ 由进制转换(除基取余)算法得到, 高位不足时用零补位 (m_n 为低位).

根据函数 γ、ρ, 自然数与路径之间建立了对应关系. 为了便于进一步利用自然数区间来表示模糊集合子搜索空间, 本文使用树状搜索空间 T_L^Y 的分支上所定义的序关系, 即 $b_i^0<b_i^1<\cdots<b_i^{\|L\|-1}$, 其中 i 表示第 i 级分支 (见定义 1). 对树状搜索空间 T_L^Y 进行深度优先遍历, 所得到的路径序列与同规模的自然数序列同序.

命题 2　由树状搜索空间 T_L^Y 深度优先遍历所得的路径序列与 $[0,\|L\|^{\|Y\|}-1]$ 规模自然数列同序.

证明　1) 按照分支上所定义的序关系, 即 $b_i^0<b_i^1<\cdots<b_i^{\|L\|-1}$, 对树状搜索空间 T_L^Y 进行深度优先遍历, 路径序列的第 1 条路径为 $p_0=(b_0^0,b_1^0,\cdots,b_n^0)$, 则 $\rho(p_0)=0$.

2) 设路径序列中的任意一条路径 $p_l=(b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n})$, 则

$$\rho(p_l)=m_0\times\|L\|^{\|Y\|-1}+m_1\times\|L\|^{\|Y\|-2}+\cdots+m_{n-1}\times\|L\|^1+m_n\times\|L\|^0.$$

此时又分如下 2 种情况:

① 当 $m_n<\|L\|-1$ 时, 依照深度优先遍历规则, p_l 的下一条路径

$$p_{l+1}=(b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n+1}),$$

故 $\rho(p_{l+1})=\rho(p_l)+1$.

② 当 $m_n=\|L\|-1$ 时, 依照深度优先遍历规则, 说明 $b_{n-1}^{m_{n-1}}$ 下的第 n 级分支已经遍历完毕. 如果 $m_{n-1}<\|L\|-1$, 则从 $b_{n-1}^{m_{n-1}+1}$ 的第 n 级第 0 个分支开始遍历, 有

$$p_{l+1}=(b_0^{m_0},b_1^{m_1},\cdots,b_{n-1}^{m_{n-1}+1},b_n^0),$$

故 $\rho(p_{l+1})=\rho(p_l)+1$; 如果 $m_{n-1}=\|L\|-1$, 则以②的开始证明方法进行类推, 同理可证 $\rho(p_{l+1})=\rho(p_l)+1$. 因此, 按照深度优先遍历序列 $p_l<p_{l+1}\Rightarrow\rho(p_l)<\rho(p_{l+1})$.

3) 树状搜索空间 T_L^Y 所有路径的个数为 $\|L\|^{\|Y\|}$. 根据 1) 和 2), 命题得证. □

根据命题 2, 可以使用自然数列对树状搜索空间 T_L^Y 进行表示和划分. 又根据命题 1, 对树状搜索空间的遍历过程即是对模糊集合组合空间 S_L^Y 的搜索过程.

1.2　模糊概念的产生

在遍历树状搜索空间 T_L^Y 过程中, 逐步完成模糊概念的构造.

定义 7 (路径前缀)　对于树状搜索空间 T_L^Y 的任意一条路径 $p=(b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n})$, 其路径前缀为

$$\mathrm{pre}_i(p)=(b_0^{m_0},b_1^{m_1},\cdots,b_n^{m_n})\bigcap(0,\cdots,i)=(b_0^{m_0},b_1^{m_1},\cdots,b_i^{m_i},b_{i+1}^0,\cdots,b_n^0).\quad(6)$$

其中: $b_i^{m_j}$ 表示高度为 i 的第 m_j 分支, $m_j\in\{0,1,\cdots,\|L\|-1\}$, $0\leqslant i<n$.

命题 3　路径前缀 pre_i 对应一个子搜索空间, 可以使用自然数区间 $[\rho(\mathrm{pre}_i),\rho(\mathrm{pre}_i)+\|L\|^{\|Y\|-i-1}-1]$ 表示.

证明　根据路径前缀 pre_i 定义, 其由树状搜索空间 T_L^Y 的前 i 个分支组成 $(0\leqslant i<n)$. 因此, 由该路径前缀组成的路径可以有 $\|L\|^{\|Y\|-i-1}$ 条, 路径前缀 pre_i 对应一个子搜索空间. 根据命题 2 和函数 ρ 的定义, 该子搜索空间可以使用自然数区间表示, 即 $[\rho(\mathrm{pre}_i),\rho(\mathrm{pre}_i)+\|L\|^{\|Y\|-i-1}-1]$. □

命题 4　由路径前缀 pre_i 所确定的模糊概念为 $(\theta(\mathrm{pre}_i)^{\downarrow},\theta(\mathrm{pre}_i)^{\downarrow\uparrow})$.

证明　根据模糊形式概念定义[18], 任意路径前缀 pre_i 所代表的属性模糊集合的模糊概念可由闭包运算求得, 即 $(\theta(\mathrm{pre}_i)^{\downarrow},\theta(\mathrm{pre}_i)^{\downarrow\uparrow})$. □

根据闭包运算的性质, 多个子集可以产生同一闭包. 为了避免模糊概念被重复构造, 需要对有效的路径前缀进行定义.

定义 8 (有效路径前缀)　对于路径前缀 pre_i 所代表的模糊集合, 如果满足

$$\mathrm{pre}_i=\omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow})\bigcap\{1,2,\cdots,i\},$$

其中 $\downarrow$ 和 $\uparrow$ 为模糊伽罗瓦联系 (详见文献 [18]), 则称 pre_i 为有效路径前缀.

如果一条路径前缀是有效的, 则说明其闭包在该路径前缀的搜索区间内; 反之, 如果路径前缀无效, 则说明该前缀闭包在其他路径前缀的搜索区间. 路径前缀的有效性判断, 为模糊概念的产生确定了一个顺序, 使之可以在对搜索树的遍历过程中依次产生, 同时也为搜索空间的缩减提供了理论依据.

1.3　搜索空间缩减

命题 5　如果一个路径前缀 pre_i 无效, 则其代表的搜索空间 $[\rho(\mathrm{pre}_i),\rho(\mathrm{pre}_i)+\|L\|^{\|Y\|-i-1}-1]$ 可以缩减.

证明 1) 假设 pre_i 为一条无效前缀, 即

$$\mathrm{pre}_i \neq \omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow}) \bigcap \{0,1,\cdots,i\},$$

则说明该前缀闭包 $\omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow})$ 不在路径前缀 pre_i 的搜索区间内.

2) 假设 pre_{i+1}^j 为 pre_i 的一条路径前缀递增, 则

$$\mathrm{pre}_{i+1}^j \bigcap \{0,1,\cdots,i\} = \mathrm{pre}_i \bigcap \{0,1,\cdots,i\}.$$

根据闭包运算特性, 有

$$\theta(\mathrm{pre}_i) \subset \theta(\mathrm{pre}_{i+1}^j) \Rightarrow \theta(\mathrm{pre}_i)^{\downarrow\uparrow} \subseteq \theta(\mathrm{pre}_{i+1}^j)^{\downarrow\uparrow}.$$

又由 1) 知 pre_i 为一条无效前缀, 因此

$$\mathrm{pre}_{i+1}^j \neq \omega(\theta(\mathrm{pre}_{i+1}^j)^{\downarrow\uparrow}) \bigcap \{0,1,\cdots,i\}.$$

由定义 8 知, pre_{i+1}^j 为无效路径前缀.

3) 根据 1)、2) 和归纳法, 如果一个路径前缀 pre_i 无效, 则其所代表的搜索空间

$$[\rho(\mathrm{pre}_i), \rho(\mathrm{pre}_i) + \|L\|^{\|Y\|-i-1} - 1]$$

内的所有路径前缀递增均无效, 因此该搜索空间可缩减. □

命题 6 如果一条路径前缀 pre_i 有效, 则搜索区间 $[\rho(\mathrm{pre}_i), \rho(\omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow}))]$ 可缩减.

证明 如果一条路径前缀 pre_i 有效, 即

$$\mathrm{pre}_i = \omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow}) \bigcap \{0,1,\cdots,i\},$$

则只需证明搜索区间 $[\rho(\mathrm{pre}_i), \rho(\omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow}))]$ 内无满足路径有效性的闭包, 即可说明该区间可以缩减.

反证法. 假设搜索区间 $[\rho(\mathrm{pre}_i), \rho(\omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow}))]$ 内存在闭包 B, 即 $\mathrm{pre}_i \subset \omega(B) \subset \omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow})$, 则满足

$$\mathrm{pre}_i = \omega(B) \bigcap \{0,1,\cdots,i\},$$

B 是路径前缀 pre_i 的有效递增闭包. 又因为

$$\omega(\theta(\mathrm{pre}_i))^{\downarrow\uparrow} \subset \omega(B)^{\downarrow\uparrow} \subset \omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow}),$$

所以 $\omega(B)^{\downarrow\uparrow} = \omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow})$, 与假设矛盾, 故搜索区间 $[\rho(\mathrm{pre}_i), \rho(\omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow}))]$ 无满足路径有效性的闭包. 另一方面, 根据命题 4, 因路径前缀 pre_i 有效, 模糊概念 $(\theta(\mathrm{pre}_i)^{\downarrow}, \theta(\mathrm{pre}_i)^{\downarrow\uparrow})$ 已被求出, 故该概念所代表的自然数 $\rho(\omega(\theta(\mathrm{pre}_i)^{\downarrow\uparrow})$ 无需再计算. □

根据命题 5 和命题 6, 无论是发现无效路径, 还是求得模糊概念, 均会对搜索空间进行缩减. 显然在完全搜索树越低的层次进行剪枝, 缩减效果越明显. 因此, 本文算法采用广度优先遍历.

1.4 负荷衡量与分配

对于一棵高度为 $\|Y\|$ 的 $\|L\|$ 叉完全搜索树, 初始时, 其搜索空间可以用一个自然数区间表示, 即 $[0, \|L\|^{\|Y\|} - 1]$; 但是, 经过路径前缀有效性检查之后, 缩减后的搜索区间不一定连续, 故需采用区间集合来表示, 定义如下.

定义 9 (区间集合) 由有效搜索区间构成的集合, 表示为 $E = \{e_0, e_1, \cdots\}$. 其中: $e_i = [n_1, n_2]$, n_1、$n_2 \in N$.

目前, 模糊概念的构造依然基于模糊伽罗瓦联系闭包运算. 根据命题 4 和命题 6, 模糊概念可以通过路径前缀有效性检查 (见定义 8) 依次产生, 同时缩减搜索区间集合. 而对于一棵高度为 $\|Y\|$ 的 $\|L\|$ 叉完全搜索树, 每级 i 所需要进行的路径有效性检查次数, 可以通过当前有效搜索区间所包含的第 i 级路径前缀 pre_i 的个数来预测, 其与每级所要进行的闭包运算的次数相同, 故第 i 级计算负荷总量定义如下.

定义 10 (第 i 级计算负荷) 其计算公式为

$$Q_i = \sum_{\forall [n_1,n_2] \in \widetilde{E}_{i-1}} \left(\left\lfloor \frac{\rho(\mathrm{pre}_i(n_2)) - \rho(\mathrm{pre}_i(n_1))}{\|L\|^{\|Y\|-i-1}} + 1 \right\rfloor - a \right);$$

$$a = \begin{cases} 1, \ \rho(\mathrm{pre}_i(\gamma(n_1))) < n_1; \\ 0, \ \text{otherwise}. \end{cases} \tag{7}$$

当 $\rho(\mathrm{pre}_i(\gamma(n_1))) < n_1$ 时, 意味着 n_1 的第 i 级路径前缀 $\mathrm{pre}_i(\gamma(n_1))$ 不在当前搜索区间内, 因此需要减掉. 初始时, 第 0 级的总计算负荷 $Q_0 = \|L\|$. 第 i 级计算负荷 Q_i 由第 $i-1$ 级路径前缀有效性检查并缩减后的搜索区间 $\widetilde{E}_{i-1}$ 内所包含的第 i 级路径前缀 pre_i 的总个数来衡量.

已知当前参与运算的计算节点个数为 D, 于是对计算负荷 Q_i 的分配工作是将当前有效搜索区间 $E_i = \widetilde{E}_{i-1}$ 划分成 D 个子搜索区间, 使得每个子搜索区间 $\mathrm{Sub}E_i^j$ ($j \in \{0,1,\cdots,D-1\}$) 包含 Q_i/D 次路径前缀有效性检查任务 (若 Q_i/D 不能被整除, 则需要适当调整, 使得每个计算节点承担的有效性检查次数相差不大于 1. 受篇幅所限, 且因该分配工作属于具体技术问题, 实现方法多种多样, 本文不再详述.

每个计算节点获得各自的子搜索区间后, 并行、独立地对第 i 级的路径前缀进行检查, 缩减搜索空间, 产生模糊概念.

1.5 算法描述

依据上述理论, 基于负载均衡的模糊概念并行构造算法, 简称 PaFuCo, 采用单程序多数据流 (SPMD) 组织方式, 描述如下.

算法 1 PaFuCo.

输入: 模糊形式背景 K, 处理器数量 D;

输出: 全局模糊概念集合 $C(L, X, Y, I)$.

Step 1: 根据输入的模糊形式背景, 搜索空间是高度为 $\|Y\|$ 的 $\|L\|$ 叉完全搜索树 T_L^Y, 全局区间集合 $E = \{[0, \|L\|^{\|Y\|} - 1]\}$.

Step 2: 对完全搜索树 T_L^Y 进行自上而下, 广度优先遍历. 令 i 表示当前高度, $i=0$.

Step 3: 根节点处理器 (ID $=0$) 根据式 (7), 由全局搜索区间集合 E 计算当前高度 i 下的总负荷 Q_i. 根据当前所能支配的处理器数量 D, 按照第 1.4 节所述原则均衡分配计算负荷, 对当前全局搜索区间集合 $E_i=\widetilde{E}_{i-1}$ 进行划分, 获得 D 个局部任务区间集合 $\mathrm{Sub}E_i^j, j\in\{0,1,\cdots,D-1\}$. 非根节点处理器 (ID >0) 则等待获得任务分配.

Step 4: ID $=j$ 的处理器获得局部任务区间集合 $\mathrm{Sub}E_i^j$, 然后独立、并行地执行以下内容:

1) 对 $\mathrm{Sub}E_i^j$ 中的每个搜索区间 e 内的第 i 级路径前缀逐一进行检查. 如果有效, 则根据命题 4 产生模糊概念, 放入局部模糊概念集合 C_{sub}^j 中, 并根据命题 6 缩减当前搜索区间; 如果无效, 则根据命题 5 直接缩减搜索区间. 缩减后的搜索区间均放入局部区间集合 $\mathrm{Sub}\widetilde{E}_i^j$ 中.

2) $i<\|Y\|-1$ 时, 所有非根节点处理器 (ID >0) 将局部区间集合 $\mathrm{Sub}\widetilde{E}_i^j$ 发送到根节点处理器 (ID $=0$), 然后转 Step 6; 根节点处理器 (ID $=0$) 则直接转 Step 5.

3) $i=\|Y\|-1$ 时, 所有节点处理器转 Step 7.

Step 5: 根节点处理器 (ID $=0$) 接收各个节点的局部区间集合 $\mathrm{Sub}\widetilde{E}_i^j$, 汇总并产生新的全局区间集合 $\widetilde{E}_i$.

Step 6: 路径前缀递增, 即 $i=i+1$. 转 Step 3.

Step 7: 汇总各个计算节点所产生的模糊概念 C_{sub}^j 到全局模糊概念集合 C 中. 算法结束.

1.6 复杂度分析

命题 7　对于模糊形式背景 $K=(L,X,Y,I)$, 最坏情况下, 算法 PaFuCo 的时间复杂度为

$$O\Big(\frac{\|L\|^{\|Y\|}}{D}\times(\|X\|\times\|Y\|+\|Y\|\times\|L\|)\Big)+$$
$$O(\|L\|^{\|Y\|-1}),$$

其中 D 为参与计算的节点数量.

证明　1) 最坏情况下, 搜索空间是高度为 $\|Y\|$ 的 $\|L\|$ 叉完全搜索树 T_L^Y 的每个叶子节点均为一个模糊概念. 因此, 无论有多少个节点处理器参与运算, 每次路径前缀有效性检查均有效, 故共进行 $\|L\|^{\|Y\|}$ 次路径检查.

2) 现在考虑每次路径检查的计算复杂度. 每次路径检查涉及到空间转换函数 θ、ρ、γ、ω 和路径前缀函数 pre_i, 以及闭包运算, 故该过程时间复杂度为

$$T_{\mathrm{check}}=$$
$$O(\|X\|\times\|Y\|)+O(\|Y\|\times\|L\|)+O(\|Y\|).$$

3) 通讯量主要由离散的搜索区间数量决定. 最坏情况下, 每个叶子节点均为模糊概念, 依据命题 6 产生概念并缩减区间. 此时, 区间缩减量为 1, 而路径前缀对应一个子搜索区间 (命题 3), 进而原连续搜索区间被划分为两个 (原区间最左值被缩减情况除外). 并行任务划分时, D 计算节点最多产生 D 个新区间. 因此, 整个计算过程搜索区间数量最多为

$$\|L\|-1+(\|L\|-1)\times\|L\|+\cdots+$$
$$(\|L\|-1)\times\|L\|^{\|Y\|-2}+D\times(\|Y\|-1)=$$
$$\frac{(\|L\|-1)\times(\|L\|^{\|Y\|-1}-1)}{\|L\|-1}+D\times(\|Y\|-1),$$

故通讯量复杂度为 $O(\|L\|^{\|Y\|-1})$.

4) 路径检查被均衡分配到 D 个计算节点上, 根据 1) ~ 3) 的分析, 算法 PaFuCo 的时间复杂度为

$$T_{\mathrm{PaFuCo}}=$$
$$O\Big(\frac{\|L\|^{\|Y\|}}{D}\times(\|X\|\times\|Y\|+$$
$$\|Y\|\times\|L\|+\|Y\|)\Big)+O(\|L\|^{\|Y\|-1}).\qquad\square$$

最坏情况下, 串行算法 Fuzzy NextClosure[15] 的时间复杂度 T_s 和并行算法 ParaFuNec[17] 的时间复杂度 $T_{\mathrm{ParaFuNec}}$ 已在文献 [17] 中分析过. 因此, 本文算法与 ParaFuNec 算法的加速比为

$$S_1=\frac{T_s}{T_{\mathrm{PaFuCo}}}<S_2=\frac{T_s}{T_{\mathrm{ParaFuNec}}}<D.\qquad(8)$$

最坏情况下, 由于搜索空间内均为模糊概念, 使得基于搜索空间的直接划分就能实现闭包运算负荷的均衡分配, 而本文算法的负载均衡方法此时并没有发挥优势. 额外的通讯代价使得本文算法的加速比略小于 ParaFuNec 算法. 但是, 当模糊概念在搜索空间逐渐稀少时, 本文的负载均衡方法的优势就能越来越显现出来, 下面通过实验来进一步说明.

2 实验与结果分析

为进一步验证本文算法的特性, 使用 Java 语言分别实现实验平台和相关算法, 包括: 算法 1、ParaFuNec 算法[17] 和 Fuzzy NextClosure 算法[15]. 算法 1 和 ParaFuNec 算法都使用 MPJ (http://mpj-express.org/) 实现多核并行计算. 本文实验平台及算法运行在单处理器多核 (Intel Xeon 8 核) 计算环境下, 除 Windows 操作系统以外, 无其他程序同时运行.

实验中, 剩余格 L 采用 Lukasiewicz 链[14] 作为伴随算子, 其中

$$a_k\otimes a_l=a_{\max(k+l-n,0)},$$
$$a_k\to a_l=a_{\min(n-k+l,n)}.$$

真值集合 L 选择两种不同精度: 1) $L_3=0,0.5,1$; 2)

$L_5 = 0, 0.25, 0.5, 0.75, 1$. 计算节点个数范围为: 2, 3, 4, 5, 6, 7, 8. 实验采用3组随机产生的数据作为数据集, 分别为数据集1、数据集2和数据集3. 它们都拥有10 000个事务, 10个属性, 区别在于平均非零项分别为20%、40%、80%. 每个数据集分别具有3值和5值属性值, 以对应L_3、L_5两种精度. 各个数据集下所产生的模糊概念数量如表1所示.

表 1 各个数据集下所产生的模糊概念数量

精度	数据集1	数据集2	数据集3
L_3	2 669	8 224	23 613
L_5	51 108	327 472	1 757 263

实验分为两个部分, 完备性实验和并行性能实验. 完备性实验使用本文算法在数据集1上所产生的模糊概念与Fuzzy NextClosure[15]所产生的模糊概念进行比较. 比较内容包括模糊概念个数、概念内涵与外延. 实验结果表明, 本文算法能够产生完备的模糊概念, 同时也说明了本文算法所依据理论的正确性.

并行性能实验通过4个指标: 加速比[19]、可扩展性[20]、计算负荷的均方差和通讯量来刻画本文算法的各种性能. 计算负荷的均方差能够反映各个节点计算任务分配的均衡程度, 方差值越小, 说明每个节点的闭包计算任务越接近于均值. 每个节点完成缩减搜索区间以后, 将有效区间汇总于根节点, 然后再进行下一级的任务分配. 因此, 搜索区间是主要的通信内容. 通讯量由有效自然数区间个数来衡量. 实验结果如图1～图6所示

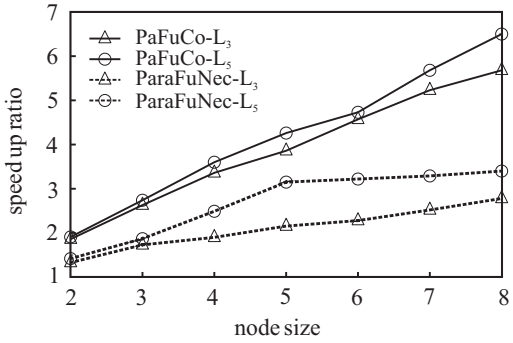

图 1 不同节点情况下的加速比(数据集1)

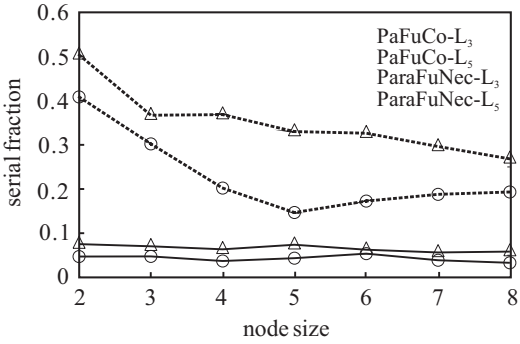

图 2 不同节点情况下的串行比例(数据集1)

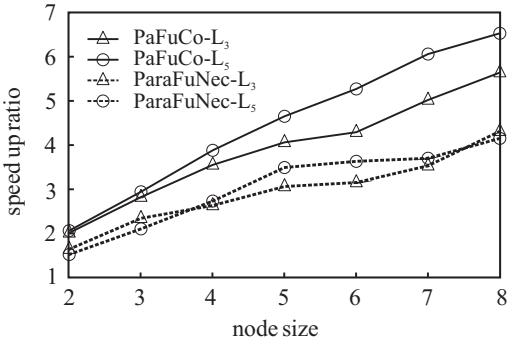

图 3 不同节点情况下的加速比(数据集2)

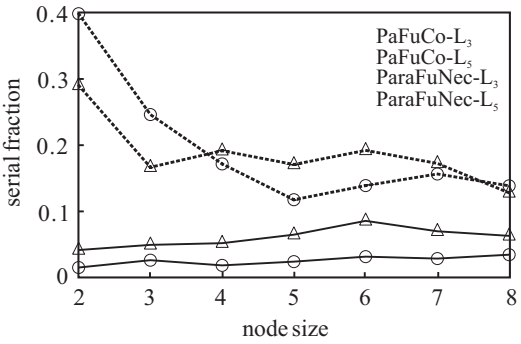

图 4 不同节点情况下的串行比例(数据集2)

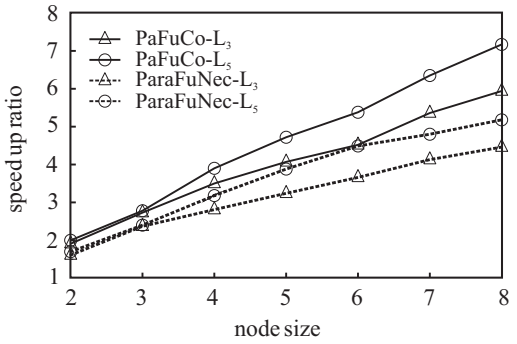

图 5 不同节点情况下的加速比(数据集3)

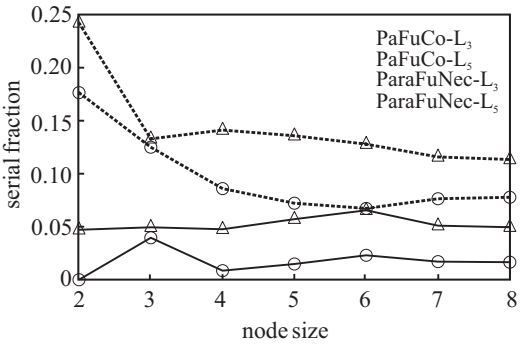

图 6 不同节点情况下的串行比例(数据集3)

从图1、图3、图5中可以看出, 本文算法的加速比在各个数据集上都优于同等情况下的ParaFuNec算法. 而从图2、图4、图6中可以观察到, 本文算法在各种情况下, 各个数据集上其串行比例均小于ParaFuNec算法. 因此, 可以得出本文算法的并行性能优于ParaFuNec算法的结论. 尤其在稀疏数据集上, 表现尤为突出. 这是由于该算法采用广度优先的剪

枝策略, 其有效地缩减了搜索范围. 该算法弥补了ParaFuNec 算法在稀疏数据集上表现不足的缺陷.

表 2 列出了本文算法在精度 L_3 时, 从稀疏数据集到稠密数据集 (数据集 1、数据集 3) 上, 各种并行计算情况下 (节点 2～节点 8 表示并行度) 统计的计算负荷均方差, 同时列出了 ParaFuNec 算法的统计结果以作比较. 可以发现, 在同等情况下本文算法的计算负荷均方差均远小于 ParaFuNec 算法. ParaFuNec 算法所采用的基于搜索区间均分的策略间接地对闭包计算负荷进行了分配, 但是由于模糊概念分布不均, 导致闭包计算负荷分布不均. 这说明本文所提出的基于负载均衡的并行构造模糊概念达到了设计目标. 另一方面, 注意到随着并行度的提高, 均差逐渐减少, 这说明在任务量一定的情况下, 并行度的提高使得每个节点的计算任务相对减少.

表 2　L_3 精度下各个节点计算负荷均方差

节点	数据集1		数据集3	
	FaFoCo	ParaFuNec	FaFoCo	ParaFuNec
2	90.51	2 688.42	82.02	5 110.97
3	70.73	1 615.19	43.84	2 386.85
4	50.22	1 385.70	37.64	2 376.49
5	40.31	1 121.92	33.92	1 881.55
6	35.57	960.92	30.53	1 700.39
7	32.25	828.87	32.93	1 339.41
8	28.88	713.78	31.05	1 183.79

表 3 列出了本文算法和 ParaFuNec 算法在精度 L_5 时, 数据集 1、数据集 3 上各种并行计算情况下统计的计算负荷均方差. 整体变化趋势与表 2 相同, 区别在于均方差值都变大了. 这是由于模糊概念的特性, 精度的提高导致模糊概念数量呈指数倍增加, 计算负荷也随之增加, 所以相对于每个计算节点, 闭包计算量的不同也累计增多.

表 3　L_5 精度下各个节点计算负荷均方差

节点	数据集1		数据集3	
	FaFoCo	ParaFuNec	FaFoCo	ParaFuNec
2	1 793.93	60 203.78	5 786.96	523 225.1
3	1 281.49	39 522.48	4 131.33	337 358.6
4	854.94	26 608.36	3 166.81	256 684.7
5	710.10	18 080.02	2 515.96	175 751
6	595.12	18 340.54	2 063.65	160 484.7
7	501.92	16 514.05	1 770.39	142 534.9
8	450.02	15 513.25	1 512.67	130 472.1

表 4 显示了本文算法采用精度 L_3、L_5 在数据集 1～数据集 3 上, 不同并行度下最大通讯量 (搜索区间个数) 的变化情况. 可以发现, 通讯量随着数据集稠密度的增加而大幅增加. 结合图 1、图 3、图 5 中加速比随着数据集密度增加的变化情况可以发现, 在稠密数据集上, 本文算法的加速比逐渐接近于相同情况下的 ParaFuNec 算法. 这是由于模糊概念密度的增加, 使得 ParaFuNec 算法所采用的简单搜索空间划分方法间接导致闭包计算任务的自然划分; 另外, 通信量的提高和同步代价进一步抵消了并行计算所带来的性能增益. 但是, 从实验结果整体来看, 本文算法的并行性能、扩展性和计算负载均衡方面均优于 ParaFuNec 算法, 有效地提高了模糊概念构造效率.

表 4　不同数据集下最大通讯量

精度	数据集 1	数据集 2	数据集 3
L_3	8 293	17 634	35 529
L_5	127 783	534 843	1 703 376

3　结　论

本文算法通过将模糊集合搜索空间、搜索树和自然数区间三者之间建立联系, 使得模糊概念构造过程中的闭包计算任务得以衡量和均分, 有效地提高了模糊概念的构造效率. 未来工作主要是在并行构造模糊概念的同时产生格结构, 并进一步提高模糊概念格的构造效率.

参考文献(References)

[1] Ganter B, Wille R. Formal concept analysis: Mathematical foundations[M]. Berlin: Springer-Verlag, 1999: 66-68.

[2] 王黎明, 张卓. 基于 Iceberg 概念格并置集成的闭频繁项集挖掘算法[J]. 计算机研究与发展, 2007, 44(7): 1184-1190.
(Wang L M, Zhang Z. An algorithm for mining closed frequent itemsets based on apposition assembly of iceberg concept lattices[J]. J of Computer Research and Development, 2007, 44(7): 1184-1190.)

[3] 柴玉梅, 张卓, 王黎明. 基于频繁概念直乘分布的全局闭频繁项集挖掘算法[J]. 计算机学报, 2012, 35(5): 990-1001.
(Chai Y M, Zhang Z, Wang L M. An algorithm for mining global closed frequent itemsets based on distributed frequent concept direct product[J]. Chinese J of Computers, 2012, 35(5): 990-1001.)

[4] Tho Q T, Hui S C, Fong A C M, et al. Automatic fuzzy ontology generation for the semantic web[J]. IEEE Trans on Knowledge and Data Engineering, 2006, 18(6): 842-856.

[5] 鞠可一, 周德群, 吴君民. 混合概念格在案例相似性度量中的应用[J]. 控制与决策. 2010, 25(7): 987-992.

(Ju K Y, Zhou D Q, Wu J M. Multi-galois lattice for similarity measurement in case retrieving[J]. Control and Decision, 2010, 25(7): 987-992.)

[6] 刘宗田, 强宇, 周文, 等. 一种模糊概念格模型及其渐进式构造算法[J]. 计算机学报, 2007, 30(2): 184-188.
(Liu Z T, Qiang Y, Zhou W, et al. A fuzzy concept lattice model and its incremental construction algorithm[J]. Chinese J of Computers, 2007, 30(2): 184-188.)

[7] 胡明涵, 张俐, 任飞亮. 模糊形式概念分析与模糊概念格[J]. 东北大学学报: 自然科学版, 2007, 28(9): 1274-1277.
(Hu M H, Zhang L, Ren F L. Fuzzy formal concept analysis and fuzzy concept lattices[J]. J of Northeastern University: National Science, 2007, 28(9): 1274-1277.)

[8] 许佳卿, 彭鑫, 赵文耘. 一种基于模糊概念格和代码分析的软件演化分析方法[J]. 计算机学报, 2009, 32(9): 1-13.
(Xu J Q, Peng X, Zhao W G. An evolution analysis method based on fuzzy concept lattice and source code analysis[J]. Chinese J of Computers, 2009, 32(9): 1832-1844.)

[9] Belohlavek R, Vychodil V. What is a fuzzy concept lattice[C]. Proc of Concept Lattices and Their Applications. Olomouc: CEUR-WS, 2005: 34-35.

[10] Burusco A, Fuentes-Gonzalez R. The study of L-fuzzy concept lattices[J]. Mathware Soft Computer, 1994, 1(3): 209-218.

[11] 范世青, 张文修. 模糊概念格与模糊推理[J]. 模糊系统与数学, 2006, 20(1): 11-17.
(Fan S Q, Zhang W X. Fuzzy concept lattice and fuzzy reasoning[J]. Fuzzy System and Mathematics, 2006, 20(1): 11-17.)

[12] 孙士保, 秦克云. 基于剩余蕴含的模糊概念格构造方法[J]. 西南交通大学学报, 2006, 41(2): 259-263.
(Sun S B, Qin K Y. Construction method of fuzzy concept lattices based on residual implication[J]. J of Southwest Jiaotong University, 2006, 41(2): 259-263.)

[13] Belohlavek R. What is a fuzzy concept lattice?—II[C]. Proc of the 13th Int Conf on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. Berlin Heidelberg: Springer-Verlag, 2011: 19-26.

[14] Belohavek R, De baets B, Outrata B, et al. Computing the lattice of all fixpoints of a fuzzy closure operator[J]. IEEE Trans on Fuzzy Systems, 2010, 18(3): 546-557.

[15] Belohavek R. Algorithms for fuzzy concept lattices[C]. Proc of the 4th Int Conf on Recent Advances in Soft Computing. Nottingham, 2002: 200-205.

[16] Lindig C. Fast concept analysis[C]. Working with Conceptual Structures — Contributions to ICCS. Aachen, 2000: 152-161.

[17] 张卓, 柴玉梅, 王黎明, 等. 模糊概念并行构造算法[J]. 模式识别与人工智能, 2013, 26(3): 260-269.
(Zhang Z, Chai Y M, Wang L M, et al. A parallel algorithm generating fuzzy formal concepts[J]. Pattern Recognition and Artificial Intelligence, 2013, 26(3): 260-269.)

[18] Belohavek R. Fuzzy galois connections[J]. Math Logic Quarterly, 1999, 45(4): 497-504.

[19] 陈国良. 并行计算——结构, 算法, 编程[M]. 修订版. 北京: 高等教育出版社, 2003: 83-87.
(Chen G L. Parallel computing: Structure, algorithm and programming[M]. Revised edition. Beijing: Higher Education Press, 2003: 83-87.)

[20] Karp A H, Flatt H P. Measuring parallel processor performance[J]. Communications of the ACM, 1990, 33(5): 539-543.

(责任编辑: 李君玲)