第48卷第4期 中国科学技术大学学报 Vol.48,No.4
2018年4月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Apr. 2018

文章编号:0253-2778(2018)04-0275-09

MapReduce环境下基于概念分层的概念格并行构造算法

蔡 勇[1,2],陈红梅[1,2]

(1.西南交通大学信息科学与技术学院,四川成都 611756;
2.四川省云计算与智能技术高校重点实验室,四川成都 611756)

摘要:概念格是形式概念分析中的核心数据结构.对此提出运用划分分治和分层约束的方法研究MapReduce框架下概念格并行生成算法以有效地构造概念格.将形式背景按对象划分成外延独立子背景后并行计算子背景上的临时概念,融合各节点临时概念形成全局概念.全局概念按照各概念外延基数进行分层,通过分层约束计算概念父子节点的搜索范围和并行搜索各层概念的父子节点,进而构建概念格.算法基于MapReduce框架实现并在公共数据集上进行测试,实验结果表明,基于概念分层方法的概念格并行构造算法能够对大数据形式背景有效地进行处理.

关键词:概念格;概念分层;并行计算;MapReduce

中图分类号:TP311 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2018.04.002

引用格式:蔡勇,陈红梅. MapReduce环境下基于概念分层的概念格并行构造算法[J]. 中国科学技术大学学报,2018,48(4):275-283.

CAI Yong, CHEN Hongmei. A parallel algorithm for constructing concept lattice based on hierarchical concept under MapReduce[J]. Journal of University of Science and Technology of China, 2018,48(4):275-283.

A parallel algorithm for constructing concept lattice based on hierarchical concept under MapReduce

CAI Yong[1,2], CHEN Hongmei[1,2]

(1. *School of Information Science and Technology, Southwest Jiaotong University, Chengdu* 611756, *China*
2. *Key Laboratory of Cloud Computing and Intelligent Technology, Chengdu* 611756, *China*)

Abstract: Concept lattice is the core data structure of formal concept analysis. A parallel algorithm is focused on for constructing concept lattice under the framework of MapReduce using the methods of divide and conquer based on partition and constrains in layers which aim to construct concept lattice effectively. Firstly, sub-formal contexts are formed by partitioning the formal context by objects and the concepts in each sub-formal context are calculated. Then the global concept is formed by merging concepts in different nodes. Next, different layers of concepts are formed by partitioning the global concept. Finally, constraints in different layers are used to compute the scope of search and concept lattice is constructed by searching and merging parent-son nodes in different layers of concepts. The proposed algorithm is realized in the framework of MapReduce. Extensive experiments carried out on public datasets verify the effectiveness of the parallel algorithm based on concept layer to deal with the formal context in big data.

Key words: concept lattice; hierarchical concept; parallel computing; MapReduce

收稿日期:2017-05-23;**修回日期**:2017-06-24

基金项目:国家自然科学基金(61572406)资助.

作者简介:蔡 勇,男,1990年生,硕士生. 研究方向:数据挖掘. E-mail: yongcai@my.swjtu.edu.cn

通讯作者:陈红梅,博士/副教授. E-mail: hmchen@swjtu.edu.cn

0 引言

自从德国数学家 Wille 于 1982 年提出形式概念分析理论[1]以来，形式概念分析得到了快速的发展和应用.概念格作为形式概念分析中的核心数据结构，被广泛地应用于数据挖掘，知识发现，软件工程，信息检索等领域[2-5].概念格的每一个节点是一个形式概念，由内涵和外延两部分组成.格结构表示了概念相互间的特化和泛化关系，并能够通过 Hasse 图直观地体现这种关系.目前已经提出的概念格构造方法主要有两种：增量算法和批处理算法.增量算法是在数据信息不确定或不完整的情况下，当有少量数据变动时，对已经构造的概念格进行更新和维护，典型代表算法有 Godin 算法[6]和 Capineto 算法[7]；批处理算法是在数据较完整的情况下，依据形式背景构造概念格的一种有效方法，典型代表算法是 Ganter 算法[8].

随着形式背景的增大，概念格构造算法的时空复杂度随之急剧增加，传统的概念格构造算法[6-8]暴露的问题越来越突出.为了提高算法的效率，有学者对建格算法提出了改进.谢润等提出了概念格的分层及逐层建格法，在建格过程中不会生成冗余格节点[9].张继福等提出基于背景知识的约束概念格构造算法[10].智慧来等提出了多概念格的合并算法[11]等.随着高性能并行计算和网络并行计算的发展，已有很多学者提出了概念格的并行建格算法.齐红等提出了基于搜索空间划分的概念生成算法，能有效地减少建格过程中的搜索次数[12].胡学钢等提出将批处理算法和增量式算法相结合的并行建格算法，该算法具有较好的时空性能[13].李云提出了基于 MPI 消息传递机制的并行构造算法，但是 MPI 环境使用进程间通信的方式协调并行计算，导致并行效率较低，内存开销大而且难以解决多节点的扩展性问题[14].董辉等提出了基于闭包系统划分的方法并行构造概念格，该方法迭代生成相互独立的子闭包系统，在一定程度上减少了计算依赖性和通信消耗问题[15].王玮提出一种基于网格的概念格分布式构造算法，采用的分发调度策略扩展性不高[16].Xu 提出了 Twister 框架下的形式概念分析并行算法[17]，该方法采用 Ganter 算法[8]的思想实现了并行的计算模型，但是在枚举所有格节点建立格结构的过程计算开销非常大，而且 Twister 框架假设内存足够大，能够存储计算过程中产生的所有的中间数据，而在实际应用中这种假设是不成立的.

面对数据量快速增长的趋势，除了硬件不断发展外，以 MapReduce[18]为代表的大数据并行计算框架正在兴起.MapReduce 是由 Google 公司提出的一种处理海量数据的并行编程模型，已广泛应用于工业界和学术界，如黄龙涛等提出的基于 MapReduce 的并行 Web 服务自动组合[19]，Kim 等提出的基于 MapReduce 环境的高效聚类算法研究[20]和张钧波等设计的基于粗糙集的并行增量知识更新算法[21]等.

针对基于大数据形式背景的概念格构造问题，结合 MapReduce 计算框架的特点，本文提出了一种 MapReduce 环境下基于概念分层方法的概念格并行构造算法.其思想是：采用分而治之的思想，将形式背景拆分成若干个外延独立的子形式背景，集群中的各节点对子形式背景并行计算得到所有概念，然后将概念按照概念外延基数进行分层，最后各节点并行地搜索满足父子关系分层中的所有概念以确定当前概念的父子概念从而形成完备的概念格.实验证明，该算法能够有效处理大数据形式背景且具有良好的并行性能.本文提出的算法的创新点主要有两点：①基于 MapReduce 平台设计了完全并行的概念格构造算法；②提出了概念分层的思想，在建立格关系的过程中能够有效提升运算效率.

1 概念格理论

本节将介绍概念格相关理论.

定义 1.1[8] 设三元组 (G,M,I) 是一个形式背景，其中 $G=\{g_1,g_2,\cdots,g_t\}$ 为对象集，每个 $g_i(1\leqslant i\leqslant t)$ 称为一个对象；$M=\{m_1,m_2,\cdots,m_s\}$ 为属性集，每个 $m_j(1\leqslant j\leqslant s)$ 称为一个属性；I 为 G 和 M 之间的二元关系，$I\subseteq G\times M$. 若 $(g,m)\in I$，则称对象 g 具有属性 m，表示为 gIm.

对于形式背景 (G,M,I)，在对象集 $X\subseteq G$ 和属性集 $Y\subseteq M$ 上定义如下运算：

$$X^*=\{m\mid m\in M,\forall g\in X,gIm\},$$

$$Y'=\{g\mid g\in G,\forall m\in Y,gIm\}.$$

$\forall g\in G$，记 $\{g\}^*$ 为 g^*；$\forall m\in M$，记 $\{m\}'$ 为 m'. 若 $\forall g\in G,g^*\neq\varnothing,g^*\neq M$，且 $\forall m\in M,m'\neq\varnothing,m'\neq G$，则称形式背景 (G,M,I) 是正则的.本文研究的形式背景都是正则的，用“1”表示 $(g,m)\in I$，否则用“0”表示.

定义 1.2[8] 设 $K=(G,M,I)$ 为形式背景，

$X, X_1, X_2 \subseteq G, Y, Y_1, Y_2 \subseteq M$ 对 $X \subseteq G, Y \subseteq M$. 如果满足 $X^* = Y$ 并且 $X = Y'$ 则称 $C = (X, Y)$ 是一个形式概念，简称概念；其中 X 称为形式概念的外延，记为 Extension(C)，Y 称为形式概念的内涵，记为 Intension(C). 形式背景 $K = (G, M, I)$ 的全部概念记为 $L(G, M, I)$；$\forall (X_1, Y_1)$，$(X_2, Y_2) \in L(G, M, I)$，记 $(X_1, Y_1) \leqslant (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2$，$Y_1 \supseteq Y_2$，符号“$\leqslant$”表示 $L(G, M, I)$ 上的偏序关系. 如果 $L(G, M, I)$ 满足：$(X_1, Y_1) \wedge (X_2, Y_2) = (X_1 \cap X_2, Y_1 \cup Y_2)$ 和 $(X_1, Y_1) \vee (X_2, Y_2) = (X_1 \cup X_2, Y_1 \cap Y_2)$，则 $L(G, M, I)$ 是完备格，称为概念格.

定义 1.3[8]　设 $K = (G, M, I)$ 为形式背景，$\forall g \in G$，$\forall m \in M$，称 $(g^{*\prime}, g^*)$ 为对象概念，(m', m'^*) 为属性概念.

定义 1.4[8]　设 $C_1 = (X_1, Y_1)$ 和 $C_2 = (X_2, Y_2)$ 是概念格 $L(G, M, I)$ 的两个不同节点，则 $C_1 < C_2 \Leftrightarrow X_1 \subset X_2 \Leftrightarrow Y_1 \supset Y_2$. 如果不存在 $C_3 = (X_3, Y_3)$ 使得 $C_1 < C_3 < C_2$ 成立，则称 C_1 是 C_2 的子概念，或称 C_1 是 C_2 的直接后继，记为 $C_1 = \text{child}(C_2)$；称 C_2 是 C_1 的父概念，或称 C_2 是 C_1 的直接前驱，记为 $C_2 = \text{father}(C_1)$.

2　概念格并行构造原理

本文设计的概念格并行构造算法主要分为：形式概念生成、概念分层和建立概念关系 3 个部分，接下来将对这 3 部分进行分析.

2.1　并行生成形式概念原理

为了直观简单地描述算法，本文引入了外延独立子背景，外延独立单元子背景，临时概念和最终概念等概念. 下面首先给出外延独立子背景，外延独立单元子背景，临时概念和最终概念的定义及相关定理.

定义 2.1　设 $K = (G, M, I)$ 是一个形式背景，$G = \{g_1, g_2, \cdots, g_t\}$ 为对象集，$G = \bigcup_{i=1}^{e} G_i$，$G_j \cap G_k = \varnothing$，$\forall j, k \in \{1, 2, \cdots, e\}$ $(j \neq k)$，则 K 按照对象划分成 e 个互不相交的子形式背景，称 $K_i = (G_i, M, I_i)$ $(i \in \{1, 2, \cdots, e\})$ 是 K 上的一个外延独立子背景. 如果 $|G_i| = 1$，称 K_i 为 K 的一个外延独立单元子背景.

定义 2.2　设 $K = (G, M, I)$ 是一个形式背景. 如果 K 的子背景 $K_i = (G_i, M_i, I_i)$ 满足 $G_i \subseteq G$，$M_i \subseteq M$，$I_i \subseteq I$，令 $C_t(X, Y) \in L(G_i, M_i, I_i)$，则称 $C_t(X, Y)$ 为形式背景 K 的一个临时概念. 设 $C_f(X, Y) \in L(G, M, I)$，称 $C_f(X, Y)$ 为形式背景 K 的一个最终概念.

定理 2.1　设 $K = (G, M, I)$ 是一个形式背景，K_i 是 K 的一个外延独立单元子背景，$M = \{m_1, m_2, \cdots, m_s\}$ 为属性集，并设 K_i 的所有子背景个数为 n，K_i 的一个子背景 $K_{is} = (g_i, M_s, I_{is})$ $(M_s \subseteq M, I_{is} \subseteq I_i, 1 \leqslant s \leqslant n)$，则 $L(K_{is}) = (g_i, g_i^*)$ $(g_i^* = \{m_i \mid m_i \in M_s, (g_i, g_i^*) \in I_{is}\})$，$K_i$ 的所有子背景的概念集合表示为 $L_i = \bigcup_{s=1}^{n} L(K_{is})$. 记 K_i 中存在关系的属性集合 $A_i = \{m_i \mid m_i \in M, (g_i, m_i) \in I_i\}$，$A_i$ 的幂集 $B_i = \{a \mid \forall a \subseteq A_i\}$，则 $L_i = \bigcup_{p=1}^{q} (g_i, B_{ip})$，其中 $B_{ip} \in B_i$ $(q = |B_i|, 1 \leqslant p \leqslant q)$.

证明　由定义 1.2 可证.

定理 2.2　设形式背景 $K = (G, M, I)$，K 的外延独立单元子背景 K_i 表达的概念集合表示为 L_i，设概念集合 $L_{\text{step}-1} = \bigcup_{i=1}^{T} L_i$，$L_{\text{step}-1}$ 中内涵相同的概念集合个数为 q，定义概念集合 $L_{\text{tmp}}^p = \{(X, Y) \mid \forall (X, Y) \in L_{\text{step}-1}, \forall \text{Intension}(X, Y) = \text{Intension}(C), C \in L_{\text{step}-1}\}$ $(1 \leqslant p \leqslant q)$，对 L_{tmp}^p 中的所有概念进行外延取并集操作可以生成新的概念 $C_{\text{new}}^p = (\bigcup_{i=1}^{n} X_i, Y_i)$，其中 $(X_i, Y_i) \in L_{\text{tmp}}^p$，$n = |L_{\text{tmp}}^p|$，则对 $L_{\text{step}-1}$ 中所有内涵相同的概念进行外延取并集操作后生成概念集合 $L_{\text{step}-2} = \bigcup_{p=1}^{q} C_{\text{new}}^p$. 同理，将 $L_{\text{step}-2}$ 中所有外延相同的概念进行内涵取并集操作后生成的概念集合记为 $L_{\text{step}-3}$，则 $L_{\text{step}-3}$ 就是形式背景 K 的最终概念 $L(K)$.

证明　设 $C_1 = (X_1, Y_1)$ $(C_1 \in L(K))$ 是形式背景 K 的一个最终概念，假设 $C_1 \notin L_{\text{step}-3}$，则一定存在概念 $C_2 = (X_2, Y_2)$ 使得 $X_1 \subset X_2$ 或 $Y_1 \subset Y_2$. 如果 $X_1 \subset X_2$，$Y_1 = Y_2$，则一定存在 g，$g \in X_2$ 且 $g \notin X_1$，此时通过外延取并集会产生新的概念 $(X_1 \cup g, Y_1)$；如果 $X_1 = X_2$，$Y_1 \subset Y_2$，则一定存在 m，$m \in Y_2$ 且 $m \notin Y_1$，通过内涵取并集会产生新的概念 $(X_1, Y_1 \cup m)$；如果 $X_1 \subset X_2$，$Y_1 \subset Y_2$，在计算的时候必然会经过 $X_1 \subset X_2$，$Y_1 = Y_2$ 和 $X_1 = X_2$，$Y_1 \subset Y_2$ 两个过程，最后会产生新的概念 $(X_1 \cup g, Y_1 \cup m)$. 此时 $C_1 \in L_{\text{step}-3}$ 与假设相矛盾. 定理得证.

通过上述分析发现，在每一个外延独立子背景上可以独立地计算出外延独立单元子背景的全部概念，这些概念属于整个形式背景的临时概念. 对所有

的临时概念,采用内涵相同则外延取并集的方式进行处理;再采取外延相同则内涵取并集的方式来处理可以得到整个形式背景的最终概念.计算临时概念可以采用 MapReduce 模型的 Map 任务进行处理,对临时概念进行合并可以采用 MapReduce 模型的 Reduce 任务来进行处理,因此计算概念的问题可以转换成 MapReduce 编程模型来并行化处理.

2.2 概念分层方法

定理 2.3 设 $C_1=(X_1,Y_1)\in L(G,M,I)$,则 C_1 的外延和 C_1 的父子概念的外延满足如下关系:

$$\text{father}(C_1)\subseteq\{C\mid\forall C\in L(G,M,I),\ |\text{Extension}(C)|>|\text{Extension}(C_1)|\};$$

$$\text{child}(C_1)\subseteq\{C\mid\forall C\in L(G,M,I),\ |\text{Extension}(C)|<|\text{Extension}(C_1)|\}.$$

证明 由定义 1.4 可证.

对概念按照外延基数进行分层存储到 HDFS,即具有相同外延基数的概念存入相同的文件,以外延基数标识对应存储的文件名字.在计算确定概念之间的父子关系的时候就可以根据当前概念所在的文件的文件名中自定义的文件名字来约束查找父子概念的搜索范围.相对于完全遍历所有概念确定概念之间的父子关系,分区存储和分区搜索的方法可以约束计算概念间父子关系时的搜索范围并降低运算时间.

2.3 并行构造格结构原理

在生成概念和对概念分层后,要建立概念间的父子关系形成完备的格结构.当一个计算节点处理一个数据分片时,要确定当前数据分片里的概念的父概念和子概念,就要用当前待处理的概念和其他所有满足搜索范围的概念进行比较.为了能够在 MapReduce 编程模型中准确地表示概念间的父子关系,在这里对概念进行编号,在表达一个概念的父概念或子概念时,直接用概念编号表示.并行环境下概念编号的定义如下.

定义 2.3 设概念 $C=(X,Y)$ 所在的文件的文件名中自定义的名字为 extSize,概念 C 在文件中为第 n 个概念,则概念 C 在概念格中的编号 Number(C)定义为"extSize-n",其中"-"为连接符.

性质 2.1 设 $C_1=(X_1,Y_1)$ 和 $C_2=(X_2,Y_2)$ 是概念格 $L(G,M,I)$ 的两个不同的概念.如果 $\text{Extension}(C_1)\subset\text{Extension}(C_2)$ 并且 $|\text{Extension}(C_2)|-|\text{Extension}(C_1)|=1$,则 $C_2=\text{father}(C_1)$.

性质 2.2 设 $C_1=(X_1,Y_1)$ 和 $C_2=(X_2,Y_2)$ 是概念格 $L(G,M,I)$ 的两个不同概念,则 $C_1<C_2\Leftrightarrow X_1\subset X_2\Leftrightarrow Y_1\supset Y_2$,如果在所有文件中自定义名中外延参数介于 $|\text{Extension}(C_1)|$ 和 $|\text{Extension}(C_2)|$ 之间的文件内不存在 $C_3=(X_3,Y_3)$ 使得 $X_1\subset X_3\subset X_2$ 成立,那么 $C_2=\text{father}(C_1)$.

3 概念格并行构造算法

根据第 2 节对形式概念原理的描述,设计了概念格并行构造算法,算法分为两个执行步骤,第一步:计算形式概念;第二步:建立格结构.概念分层方法在形式概念生成过程中得以实现.

3.1 并行生成形式概念算法

并行生成形式概念算法由两个步骤完成,第一步:对所有临时概念采用内涵相同则外延取并集的方式进行处理;第二步:对第一步输出的临时概念采取外延相同则内涵取并集的方式来处理.这两个步骤分别由算法 3.1 和 3.2 来实现.

在算法 3.1 中,为了便于处理用数字顺序编号属性,如用数字"1"表示第一个属性,数字"2"表示第二个属性,以此类推;同时用"0"表示概念的内涵为空或外延为空.为了得到全对象概念 $(G,\varnothing)$,在算法 3.1 的 Map 阶段需要输出 (0,对象编号)形式的临时概念.同时在算法 3.2 的 Reduce 阶段的 Setup 方法输出 $(\varnothing,M)$ 而得到全属性概念.

概念分层的方法在算法 3.2 的 Reduce 阶段实现,在 Reduce 任务执行之前的 Setup 方法中声明一个 MaultipleOutputs 方法,该方法可以创建多个 OutputCollectors,每个 OutputCollector 有自己的 OutputFormat 和键值类型,每个 OutputCollector 对应一个独立的输出文件.所以在算法中通过设置 MaultipleOutputs 的 OutputFormat 就可以达到对概念分层的效果.

算法 3.1 并行生成概念算法

```
Map 阶段
Input: key:g,value:K_i
Output: key:Intension(C_i),value:Extension(C_i)
1 function Map(key,value)
2   A_i←∅
3   for each x∈values do
4     if x≠0 then
5       A_i←A_i∪x
6     end if
7   end for
```

```
8    for each Bip ∈ Ai do
9        Emit(Bip key)
10     end for
11     Emit(0 key)
12 end function
```

Reduce 阶段

```
Input: key:Intension(Ct),value:Extension(Ct)
Output: key:Intension(Ct),value:Extension(Ct)
1 function Reduce(key,value)
2    V←∅
3    for each val∈values do
4       V←V∪val
5    end for
6    Emit(key V)
7 end function
```

算法 3.2　并行生成概念算法

Map 阶段

```
Input: key:Intension(Ct),value:Extension(Ct)
Output: key:Extension(Ct),value:Intension(Ct)
1 function Map(key,value)
2    Emit(value key)
3 end function
```

Reduce 阶段

```
Input: key:Extension(Ct),value:Intension(Ct)
Output: key:Extension(Cf),value:Intension(Cf)
1 function Step(M)
2    multipleOutput←new MultipleOutputs(context)
3    Emit(∅,M)
4 end function
1 function Reduce(key,value)
2    V←∅
3    for each val∈values do
4       V←V∪val
5    end for
6    multipleOutputEmit(key V key.length)
7 end function
```

为了对算法进行加速，在每一个 Map 任务执行后进行一次 Combine 操作，Combine 操作相当于在进行 Reduce 操作之前，在各节点先进行一次"min-Reduce"操作，这样能够减少节点间的数据传输量.其具体实现同 Reduce 阶段的算法一致，在算法伪代码中不再单独说明.

3.2　并行建立格结构算法

建立格结构算法由算法 3.3 实现，算法 3.3 中 Setup 方法的作用是：在每一个 Map 任务初始化时获取当前数据分片的名字并找到所有满足计算概念父子关系时搜索范围的文件，用 extSize 标识文件名里的外延基数.在算法 3.3 的 Map 阶段，根据性质 2.1和 2.2 就可以确定每一个概念的父子概念，因为所有的最终概念已经确定，只需要找到各概念的父概念就可以得到完备的概念格，所以在算法实现中只确定各概念的父概念.算法的最终输出形式为：概念编号，概念外延，概念内涵，父概念编号集合.

算法 3.3　并行建立格结构算法

Setup 阶段

```
Input: inputFiles
Output: searchArea
1 function Setup(inputFiles)
2    searchArea←∅
3    dealSplit←inputSplit
4    for each split∈inputFiles do
5       if dealSplit.extSize < split.extSize then
6          searchArea←searchArea∪split
7       end if
8    end for
9    return searchArea
10 end function
```

Map 阶段

```
Input: key:Extension(C),value:Intension(C)
Output: key: Number(C), value: Extension(C), Intension(C),fatherSet
1 function Map(key,value)
2 fatherSet←∅
3 for each file∈searchArea do
4    for each C1 ∈ file do
5       if Intension(key)⊃Intension(C1)&&|Intension(key)|-|Intension(C1)|=1 then
6          fatherSet←fatherSet∪(Number(C1))
7       else
8          flag←true
9          breakFlag:
10         for each file∈searchArea do
11            1for each C2 ∈ file do
12               if Intension(key)⊃Intension(C2)&&Intension(C2)⊃Intension(C1) then
13                  flag←false
14                  break breakFlag
15               end if
16            end for
17            end for
18            if flag=true then
19               fatherSet←fatherSet∪(Number(C1))
20            end if
21:        end if
```

```
22      end for
23    end for
24 Emit(Number(C) key,value,fatherSet)
25 end function
```

3.3 算法示例

以下列举一个例子来描述上述算法的建格过程.给定一个形式背景 $K=(G,M,I)$ 如表 1 所示,其中对象集 $G=\{1,2,3,4\}$,属性集 $M=\{a,b,c,d,e,f\}$.

表 1 形式背景 K

Tab.1 Formal context K

关系(I)		属性(M)					
		a	b	c	d	e	f
对象(G)	1	0	1	1	0	0	1
	2	1	0	0	1	0	0
	3	0	1	1	0	0	1
	4	1	0	1	0	1	0

为了充分展示算法的并行执行流程,将表 1 所给的形式背景 K 划分为 K_1 和 K_2 两个外延独立子背景作为算法 1 的输入数据,将算法 3.1 的输出数据划分为两个数据分片作为算法 3.2 的输入数据.算法 3.1 和 3.2 在形式背景 K 上的 MapReduce 执行流程分别如图 1 和图 2 所示,其执行流程都经过 Map→Combine→Reduce 三个阶段.可以发现,经过两个 MapReduce 任务处理后生成了 7 个形式概念且概念被分成了 5 层. 将生成的形式概念作为算法 3.3 的输入数据,算法 3.3 的 MapReduce 执行流程如图 3 所示,其执行流程经过 Setup→Map 两个阶段,其中在 Setup 阶段确定每一个概念层要搜索的

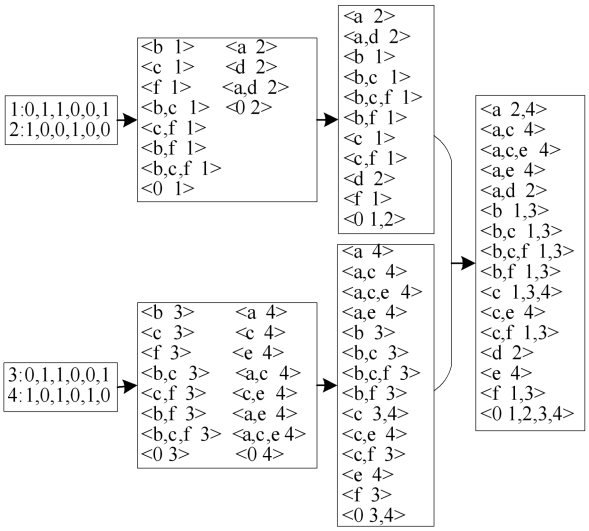

图 1 算法 3.1 的 MapReduce 执行流程

Fig.1 MapReduce execution flow of algorithm 3.1

概念层,在 Map 阶段构建概念父子关系.形式背景 K 经过 3 个 MapReduce 任务处理后得到最终的概念格,概念格 Hasse 图如图 4 所示.通过本文提出的并行算法构建的概念格与串行概念格构造算法构建的概念格完全一致.

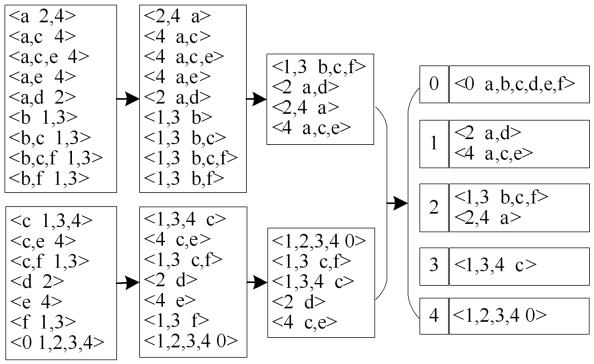

图 2 算法 3.2 的 MapReduce 执行流程

Fig.2 MapReduce execution flow of algorithm 3.2

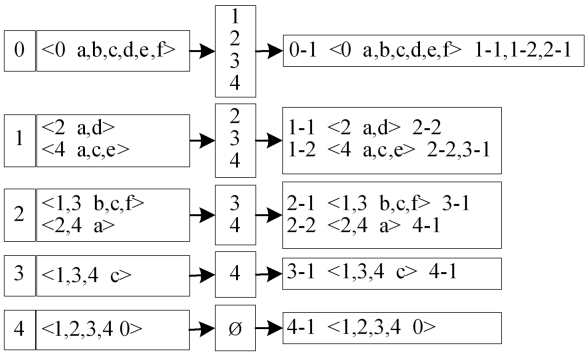

图 3 算法 3 的 MapReduce 执行流程

Fig.3 MapReduce execution flow of algorithm 3.3

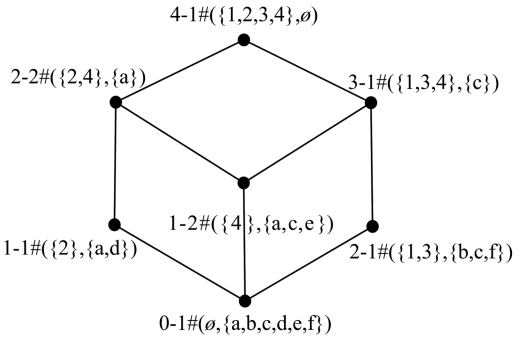

图 4 Hasse 图

Fig.4 Hasse

4 实验及算法性能分析

4.1 实验数据和实验平台

实验选取 2 个数据集作为形式背景,这 2 个数据集都是 UCI 标准数据集经过离散化处理后得到的.KDD 是从 KDDCup99 第一条记录开始间隔 100

条记录提取一次，共提取 20 000 条记录得到的；数据集 Census 是人口收入数据集.形式背景的详细信息如表 2 所示.

表 2　形式背景

Tab.2　Formal context

数据集	对象数	属性数	关系密度
KDD	20 000	42	18%
Census	103 950	133	7.6%

为了较充分地验证本算法，我们搭建了一个拥有 1 个主节点和 16 个从节点的 Hadoop 集群，集群中所有机器(下称“计算节点”)的配置完全一致，计算节点的具体配置信息如表 3 所示.

表 3　计算节点配置信息

Tab.3　Computer configuration information

类别	名称	配置说明/版本信息
硬件	内存	4.0 GB
	处理器	Intel Corei5-2400@3.10 GHz×4
	磁盘空间	500 GB
软件	操作系统	Ubuntu 15.04　64 bit
	JDK	1.7.0_25
	Hadoop	2.6.0

4.2　算法性能分析

为了评价算法的并行性能，测定了算法在保持数据集不变，按比例增加节点数目；保持节点数目不变，按比例增加数据集和数据集同节点数目等比例增长 3 种情况下的运行时间、形式概念数目和概念层数.为了使输入数据在不同节点数目下各个计算节点都能够分配到任务，将输入数据随机打乱并划分成 128 个数据分片.

表 4 和表 5 分别展示了 KDD 和 Census 在不同节点数目和不同输入规模情况下的运行时间，表 6 给出了 2 个数据集在不同规模下所生成的概念数目和概念层数.

表 4　算法在数据集 KDD 上的运行时间(单位：秒)

Tab.4　Running time of algorithm on KDD(unit：second)

节点不变，数据增加			数据不变，节点增加			节点和数据一同增加		
节点	数据	运行时间	节点	数据	运行时间	节点	数据	运行时间
16	2/16	162	2	16/16	1 885	2	2/16	332
16	4/16	228	4	16/16	1 013	4	4/16	324
16	6/16	254	6	16/16	788	6	6/16	364
16	8/16	316	8	16/16	665	8	8/16	401
16	10/16	379	10	16/16	641	10	10/16	489
16	12/16	431	12	16/16	602	12	12/16	468
16	14/16	455	14	16/16	536	14	14/16	474
16	16/16	496	16	16/16	496	16	16/16	496

表 5　算法在数据集 Census 上的运行时间(单位：秒)

Tab.5　Running time of algorithm on Census(unit：second)

节点不变，数据增加			数据不变，节点增加			节点和数据一同增加		
节点	数据	运行时间	节点	数据	运行时间	节点	数据	运行时间
16	2/16	7 277	2	16/16	150 946	2	2/16	11 750
16	4/16	10 340	4	16/16	81 592	4	4/16	13 662
16	6/16	14 336	6	16/16	51 871	6	6/16	15 671
16	8/16	18 993	8	16/16	43 499	8	8/16	18 952
16	10/16	22 849	10	16/16	39 412	10	10/16	21 364
16	12/16	25 251	12	16/16	36 997	12	12/16	22 595
16	14/16	28 745	14	16/16	32 672	14	14/16	28 659
16	16/16	30 129	16	16/16	30 129	16	16/16	30 129

表 6 数据集在不同规模下的概念数和概念层数

Tab.6 Concept number and concept layer number under different dataset scale

数据规模	不同数据集下概念个数和概念层数			
	KDD		Census	
	概念数	概念层数	概念数	概念层数
2/16	18 494	63	248 532	127
4/16	32 933	83	475 997	145
6/16	48 719	117	715 629	162
8/16	65 142	124	843 391	172
10/16	80 927	141	1 060 013	187
12/16	95 284	149	1 302 708	194
14/16	108 669	153	1 769 801	213
16/16	120 555	167	1 857 342	219

以在 2 个节点上计算所有数据的运行时间作为基准分析加速比，以在 2 个节点上处理 2/16 的数据所用的时间为基准分析算法扩展性，以在 16 个节点上运行 2/16 的数据运行时间为基准分析算法规模增长性.结合表 4 和表 5 的数据绘制了算法在 Census 和 KDD 数据集上的加速比，扩展性和规模增长性如图 5—7 所示.

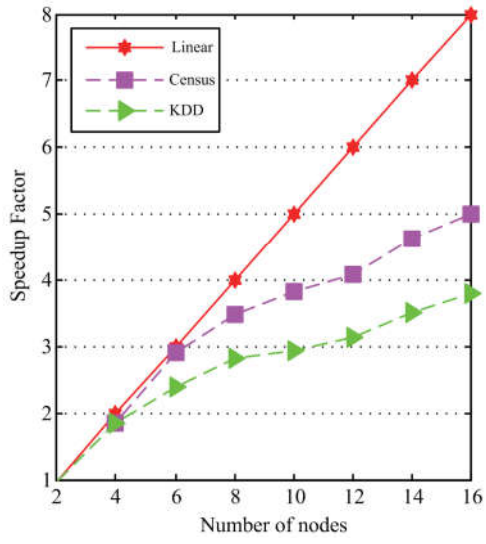

图 5 加速比

Fig.5 Speedup

分析图 5 发现，算法在两个数据集上都表现出较好的加速比，计算节点数目小于 8 时，加速性能明显，当节点数目大于 8 时，加速比增长变缓，变缓的原因是随着节点数目增加，主从节点之间的通信时间消耗变大.数据集 Census 的加速比最终趋近 5，而

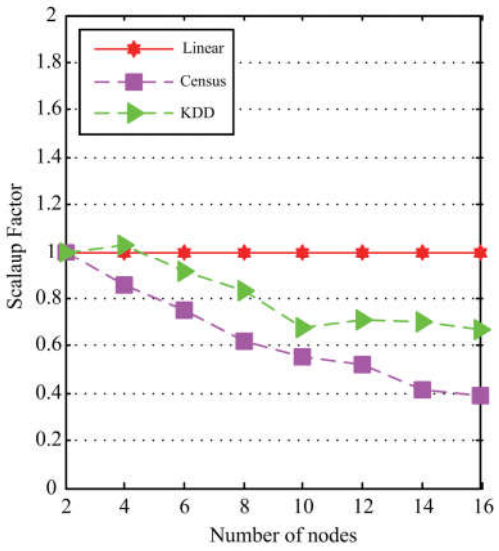

图 6 扩展性

Fig.6 Scaleup

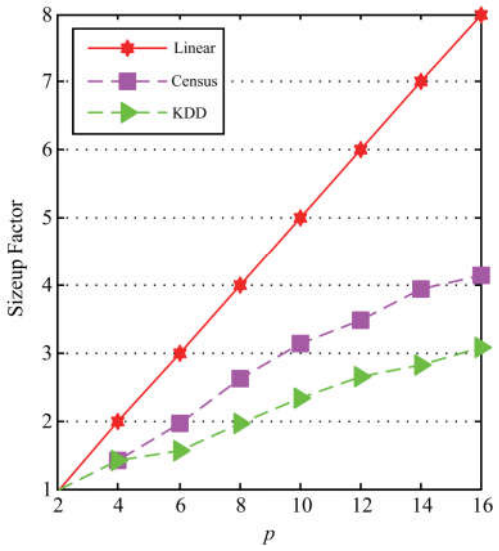

图 7 规模增长性

Fig.7 Sizeup

数据集 KDD 的加速比最终趋近 4，说明数据集越大，算法的加速性能越突出.

分析图 6 发现，随着计算节点的增多，算法的扩展性呈下降趋势，当节点数目大于 10 时，扩展性趋于稳定.结合表 6 给定的数据集在不同规模下的概念数和概念层数发现，随着数据集规模增加，概念数和概念层数相应增加，但是增加趋势与数据规模增加的趋势不完全一致，而且不同分层中概念数有很大的差别，这是造成规模增长性下降的主要原因.在节点数为 4 的时候，数据集 KDD 表现出超线性扩展性的情况，原因是计算节点的数目和处理的数据规模是基准节点数目和数据规模的 2 倍，但是概念

分层数目增加，分层粒度更小，运行时间反而更小.

分析图 7 发现，算法的规模增长性趋势较好，增长平缓，且最终算法在数据集 Census 上的增长性超过 4，在数据集 KDD 上的规模增长性也达到 3.数据集越大，算法的规模增长性越好.

综上分析，本文提出的概念格并行构造算法具有良好的并行性能，当数据规模和计算节点数目进一步增加，算法并行性能将会更加突出.

5　结论

概念格的构建是基于形式概念分析进行数据挖掘研究的基础，本文提出的基于 MapReduce 编程框架和概念分层方法的概念格并行构造算法有效地解决了大数据形式背景下概念格生成问题，使计算概念，计算概念之间的父子关系以得到最终概念格都达到完全并行计算的状态，采用对概念进行分层再计算概念之间的父子关系的方法能够有效地提高算法效率.实验结果表明，算法具有良好的并行性能，能够处理大规模的数据.

参考文献(References)

[1] WILLE R. Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts [M]. Newtherland: Springer, 1982: 445-470.

[2] DÍAZ-AGUDO B, GONZÁLEZ-CALERO P A. Formal concept analysis as a support technique for CBR [J]. Knowledge-based Systems, 2001, 14(3): 163-171.

[3] SHIVHARE R, CHERUKURI A K. Three-way conceptual approach for cognitive memory functionalities [J]. International Journal of Machine Learning and Cybernetics, 2017, 8(1): 21-34.

[4] KONECNY J, KRIDLO O. On biconcepts in formal fuzzy concept analysis[J]. Information Sciences, 2017, 375(1): 16-29.

[5] 张磊，张宏莉，韩道军，等. 基于概念格的 RBAC 模型中角色最小化问题的理论与算法[J]. 电子学报，2014，12(42)：2371-2378.

[6] GODIN R, MISSAOUI R, ALAOUI H. Incremental concept formation algorithms based on Galois (concept) lattices [J]. Computational Intelligence, 1995, 11(2): 246-267.

[7] CARPINETO C, ROMANO G. GALOIS: An order-theoretic approach to conceptual clustering [M]// Machine Learning. Morgan: Kaufmann Publishers, 1993, 33-40.

[8] GANTER B, WILLE R. Formal concept analysis. Mathematical Foundations[M]. New York: Springer-Verlag, 1999.

[9] 谢润，李海霞，马骏，等. 概念格的分层及逐层建格法[J]. 西南交通大学学报，2005，40(6)：837-841.

[10] 张继福，张素兰，胡立华. 约束概念格及其构造方法[J]. 智能系统学报，2006，1(2)：31-38.

[11] 智慧来，智东杰，刘宗田.概念格合并原理与算法[J].电子学报，2010，38(2)：455-460.

[12] 齐红，刘大有，胡成全，等. 基于搜索空间划分的并行概念生成算法[J]. 计算机科学，2005，32(4)：55-58.

[13] 胡学刚，张玉红，唐志军，等. 一种新的概念格并行构造方法 [J]. 合肥工业大学学报，2005，28(12)：1523-1527.

[14] 李云，程伟，陈崚，等. 基于消息传递的概念格并行构造[J]. 计算机应用与软件，2006，23(8)：3-5.

[15] 董辉，马垣，宫玺. 一种新的概念格并行构造算法[J]. 计算机科学与探索，2008，2(6)：651-657.

[16] 王玮，张继福. 一种基于网格的概念格分布式构造算法[J]. 太原科技大学学报，2010，31(3)：197-201.

[17] XU B, FREIN R, ROBSON E, et al. Distributed Formal Concept Analysis Algorithms based on An Iterative MapReduce Framework [M]. Berlin: Springer-Verlag, 2012: 292-308.

[18] DEAN J, GHEMAWAT S. MapReduce: Simplified data processing on large clusters[J], Communication of ACM, 2008, 51: 107-113.

[19] 黄龙涛，邓水光，戴康，等. 基于 MapReduce 的并行 Web 服务自动组合[J]. 电子学报，2012，7(40)：1397-1403.

[20] KIM Y, SHIM K, KIM M S, et al. DBCURE-MR: An efficient density-based clustering algorithm for large data using MapReduce[J]. Information Systems, 2014, 42(6): 15-35.

[21] 张钧波，李天瑞，潘毅，等. 云平台下基于粗糙集的并行增量知识更新算法[J]. 软件学报，2015，26(5)：1064-1078.