一种基于对象相容度的形式背景分割算法

杨韶华[1]，马 骏[1,2]
（1.河南大学 计算机与信息工程学院，河南 开封 475004；
2.河南大学 数据与知识工程研究所，河南 开封 475004）

摘 要：介绍了一种基于对象相容度的形式背景分割算法（OMCP算法）。算法利用提出的对象相容度的定义，根据用户需求对原形式背景进行不同规模的分割，由生成的兼容子背景构造概念格。实验证明，在各对象间的相容度差别较大的情况下，与传统的形式背景预处理方法相比，算法生成的兼容子背景在构造概念格时不但提高了效率，而且使得生成的每个格节点与用户需求相关，提高了概念格的有用性。

关键词：对象相容度；形式背景分割；预处理；兼容子背景；概念格
中图分类号：TP311.1 **文献标识码**：A **文章编号**：1673-629X(2008)02-0027-04

A Context Partition Algorithm Based on Objects-Match

YANG Shao-hua[1], MA Jun[1,2]
(1. College of Computer and Information Engineering, Henan University, Kaifeng 475004, China;
2. Institute of Data and Knowledge Engineering, Henan University, Kaifeng 475004, China)

Abstract: A context partition algorithm based on the objects-match is introduced in this paper. According to the demands of users, this algorithm can construct the lattice from subcontext generated by the original formal context in different scales. Compared with the traditional method of formal context pretreatment, the results show that it not only improve the efficiency when the objects-match is larger difference between different objects, but also makes the concept lattices associated with the user needs more usefulness.

Key words: objects-match; formal context partition; pretreatment; subcontext compatibility; lattices

0 引 言

概念格是基于形式背景的一种形式概念表达方式，它描述了对象和属性间的关系，表明了概念之间的泛化与特化特征，已经在信息检索、软件工程和知识发现等领域得到了广泛的应用。

由于形式背景的规模决定了概念格构造的时间和空间复杂度，因此，不少研究者提出了多种形式背景的约减算法和用于格并行构造的形式背景横向、纵向拆分算法。但是如何有效地缩减概念格的规模使其更符合实际需求，却至今没有非常理想的处理方式。随着信息量的不断增大，如何提高信息有用性也逐渐成为新的问题。对于海量信息来说，其对应的形式背景非常庞大，那么构造概念格的时间复杂度也非常高。而对于一个用户来说，海量信息并不是都有用的，如果不进行筛选，而是直接提供给用户一个概念格，那么很多的格节点对于用户来说用处不大或者根本没有用。因此，首先提出了对象相容度（Objects-Match）的定义，并在此基础上提出了一种基于对象相容度的形式背景分割算法（Objects-Match Context Partition algorithm，OMCP算法）。算法根据用户需求对海量信息进行处理，只形成原形式背景的一个兼容子背景，再从兼容子背景构造概念格，从而不但缩小了格的规模，也提高了格节点的有用性。

收稿日期：2007-05-16
基金项目：河南省高校杰出科研人才创新工程项目资助（2007KYCX018）；河南大学科研基金（05YBZR008）
作者简介：杨韶华（1979-），女，河南平顶山人，硕士研究生，研究方向为知识发现和分布式处理；马 骏，副教授，硕士生导师，研究方向为分布式计算、知识发现及网络应用。

1 形式背景与概念格

形式背景是一个三元组 $K=(G,M,R)$，其中 G 表示对象的集合，M 表示属性的集合，R 表示 G 和 M 之间的一个二元关系，即 $R\subseteq G\times M$。每个形式背景都可以用一个数据表来表示，它描述了对象及其特征之间的自然分组和关系的有序集。表1列出了一个形式背景的例子。

在形式背景 $K=(G,M,R)$ 中，G 的幂集 $P(G)$

和 M 的幂集 $P(M)$ 之间可以定义一个运算符"′"满足：$\forall A \subseteq G: A' = \{m \mid \forall a \in A \cdot (aRm)\}$ 和 $\forall B \subseteq M: B' = \{g \mid \forall b \subseteq B \cdot (gRb)\}$ 两个条件。例如在表 1 的形式背景中，$\{a,b\}' = \{5,6\}$，$\{3,5\}' = \{c\}$。

表 1 一个形式背景的例子

对象 / 属性	a	b	c	d
1	√			
2		√		
3			√	
4				√
5	√	√	√	
6	√	√		√

如果二元组 $(A,B) \in P(G) \times P(M)$ 满足：$A = B'$ 且 $B = A'$，则称之为形式背景 K 的一个形式概念，简称概念，记为 $C = (A,B)$，其中 A 和 B 分别被称为概念 C 的外延和内涵。$(\{3,5\},\{c\})$ 就是表 1 所示形式背景的一个概念。K 的所有形式概念的集合被标记为 $CS(K)$。$CS(K)$ 上的偏序关系可定义为子概念 - 超概念关系（又称为例化 - 泛化关系），即如果 $A_1 \subseteq A_2$（即 $B_2 \subseteq B_1$），则形式概念 (A_1,B_1) 是形式概念 (A_2,B_2) 的亚概念，记为 $(A_1,B_1) \leqslant (A_2,B_2)$。

把有序集 $\underline{CS}(K) = (CS(K), \leqslant)$ 称为形式背景 K 的概念格，记为 $L(K)$。图 1 即为表 1 形式背景对应的概念格。

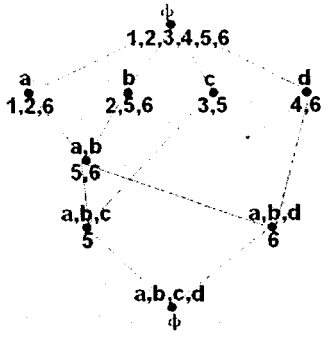

图 1 表 1 形式背景对应的概念

2 几种常见的形式背景预处理方法

为了降低格构造的时间和空间复杂度，通常要先对形式背景进行预处理，常见的预处理方式有：形式背景的约简和形式背景的横向、纵向拆分。

2.1 形式背景约简

对于每个有限格 V，都存在唯一的约简形式背景 $K(V)$ 使得 $V \cong \underline{B}(K(V))$，这个形式背景叫做格 V 的标准形式背景。也就是说每一个有限的形式背景都可以变换到约简形式，而不改变对应概念格的结构。表 3 中的形式背景就是表 2 的约简形式。

表 2 约简前的形式背景

对象 / 属性	1	2	3	4	5
a	√	√	·		
b	√	√	√	√	√
c		√	√	√	√
d		√			
e		√	√	√	

表 3 约简后的形式背景

对象 / 属性	1	2	3	5
a	√	√		
c		√	√	√
e		√	√	

根据这一理论，很多研究者提出了多种形式背景的约简算法，例如基于属性频率函数的粗糙集属性约简算法、基于包含度的属性约简算法[1]、基于决策表的属性约简算法和基于评价指数的属性约简算法等。这些属性约简算法通常都是利用各种定义的度量标准将属性分为必要属性和不必要属性，从而约简形式背景，降低构造格的时间和空间复杂度。

但是，这些约简算法都是遵循不改变格结构的原则，当信息量非常庞大时，由于对应的格结构固定，因此无论如何约简形式背景，格的构造仍然是问题。

2.2 形式背景横向、纵向拆分

为了解决信息量庞大时格构造时间复杂度高的问题，文献[2]中利用高性能并行计算机和网络并行计算的能力提出了概念格的一种并行算法[2]。文献[3]中提出了一种基于同类概念的概念格横向合并算法。这些形式背景的拆分和格的并行构造算法的主要思想是，首先将形式背景利用横向拆分或者纵向拆分方法划分成子形式背景，然后利用 n 阶形式背景核等方法构造子格，再将子格合并，从而生成所需概念格。但是，目前形式背景的拆分只是简单的横向或者纵向拆分，例如将一个 5×5 的形式背景拆分成一个 2×5 和一个 3×5 的背景。如何拆分才能使概念格的并行构造效率更高仍然是个问题。

3 对象相容度

由于子背景的概念格和完整格的子序是同构的，并且兼容子背景的概念格是完整格的同形映射[4]。因此结合这些特性，提出了对象相容度的定义，并以此为基准提出了 OMCP 算法，利用该算法生成兼容子背景。下面介绍涉及的定义。

设形式背景 $K = (G,M,R)$，如果 $M' \subseteq M$，那么 $(G,M',R \cap G \times M')$ 的每个属性的外延也就是 (G,M,R) 的属性的一个外延。这意味着属性的省略等价于外延的封闭系统的粗糙化。同样，对象的省略等价于

内涵的封闭系统的粗糙化。

对于概念格 $\underline{B}(G,M,R)$,如果 $M' \subseteq M$,则映射 $\underline{B}(G,M',R \cap G \times M') \to \underline{B}(G,M,R),(A,B) \mapsto (A,A')$ 是一个 $\wedge$ - 保序包含。对应的有,如果 $G' \subseteq G$,映射 $\underline{B}(G',M,R \cap G' \times M) \to \underline{B}(G,M,R),(A,B) \mapsto (B',B)$ 是一个 $\vee$ - 保序包含。

图 2 是子背景概念格的 $\wedge$ - 保序包含的一个例子。

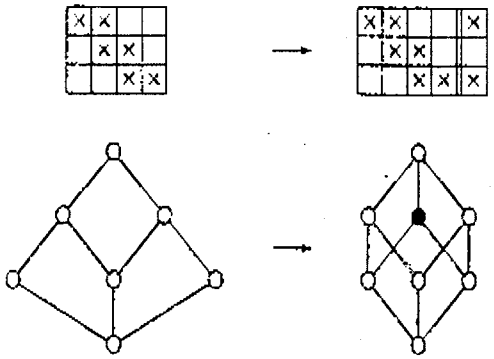

图 2　子背景概念格的 $\wedge$ - 保序包含

若 $G' \subseteq G$ 且 $M' \subseteq M$,则映射 $\underline{B}(G',M',R \cap G' \times M') \to \underline{B}(G,M,R),(A,B) \mapsto (A'',A')$ 是一个保序包含。同样,映射 $(A,B) \mapsto (B',B'')$ 也是一个保序包含[5]。

若每一个概念 $(A,B) \in \underline{B}(G,M,R),(A \cap H, B \cap N)$ 都是子背景的概念,则子背景 $(H,N,R \cap H \times N)$ 称为兼容子背景。背景 $K=(G,M,R)$ 的兼容子背景的概念格是 $\underline{B}(G,M,R)$ 的同形映射。

定义 1　设 g_1,g_2 是 G 中的两个对象,则 g_1,g_2 的对象相容度 $\mathrm{OM}(g_1,g_2)$ 定义为

$$\mathrm{OM}(g_1,g_2)=\frac{|g'_2 \cap g'_1|}{|g'_1|}$$

定义 2　设 g 是形式背景 $K=(G,M,R)$ 中的一个对象,则 g 在 K 中的对象相容度 $\mathrm{OM}(g)$ 定义为

$$\mathrm{OM}(g)=\frac{|o' \cap (\mathrm{ExaC})'|}{|\mathrm{ExaC}'|}+\sum_{i=1}^{n}\left[\frac{|o' \cap (\mathrm{ExaC}_i)'|}{|\mathrm{ExaC}'_i|} \times (1/100)^i\right]$$

其中 ExaC 是精确比对算子,ExpC_i 是第 i 次扩展比对算子,n 是扩展比对的次数。

$\mathrm{OM}(g)$ 的值表明了对象 g 与用户需求的相容度,其值在 0~1 之间,值越高说明相关度越高。在实际应用中,给对象相容度制定合适的阈值(Limit of the OM)是能否生成有效规模的概念格的关键。

4　形式背景分割算法

4.1　算法描述

笔者等提出的 OMCP 算法对原形式背景 $K=(G,M,R)$ 进行分割,首先由用户输入请求 input,根据 input 生成比对算子 ExaC 对原形式背景中的每一个对象 $g(\in G)$ 进行比对,从而生成兼容子背景 $K''=(G',M',R')$。并根据用户要求精确程度的不同修改相容度阈值,生成不同规模的 K'。

算法描述如下:

1) 确定相容度阈值 LimOM,处理用户需求 input。

2) 根据用户需求生成精确比对算子 ExaC。

3) 创建空形式背景 $K'=(G',M',R')$。

4) 如果 G 非空,则从中取出一个对象 g,否则转至 11)。

5) 计算 $\mathrm{OM}(g)$。

6) 如果 $\mathrm{OM}(g) \geqslant \mathrm{LimOM}$,则将对象 g 加入 K' 的对象集 G',否则转至 4)。

7) 如果 g' 非空,则从中取出一个属性 a,否则转至 4)。

8) 如果 $a \notin M'$,则将 a 加入 K' 的属性集 M',否则转至 7)。

9) 生成精确兼容子背景 K'。

10) 如果用户对 K' 的规模不满意,则增大或缩小阈值,并生成扩展比对算子(ExpC)或者二级精简算子(NexC),转至 4)。否则转至 11)。

11) 约简并输出 K'。

4.2　算法示例分析与软件实现

下面结合具体例子说明形式背景分割算法。

表 4　原形式背景 $K=(G,M,R)$

对象 / 属性	1	2	3	4	5	6
a	√				√	√
b	√			√		√
c		√			√	√
d		√		√		√
e		√	√			

表 4 中形式背景 K 所对应的概念格 $L(K)$ 如图 3 所示。

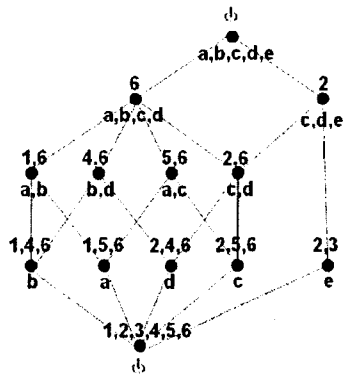

图 3　概念格 $L(K)$

假定用户的请求是 a，则计算出比对算子 $ExaC = \{1,0,0,0,1,1\}$，使用上述算法得到兼容子背景 $K' = (G', M', R')$，如图 4 所示，其中由于对象 e 的对象相关度低于阈值未被加入到 K'，属性 3 和 6 也被约简掉了。

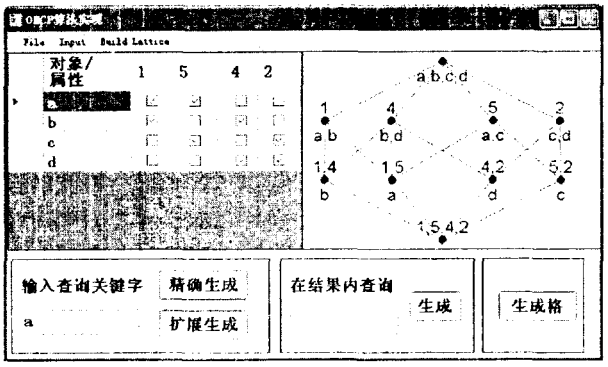

图 4 K' 及 $L(K')$

从图中可以很清晰地看到，$L(K')$ 同构于 K 的一个子序的概念格。如果用户利用精确分割后感觉生成的背景 K' 过小，可选择利用扩展分割算法扩大形式背景的规模。由于本例的特殊性，扩展后的形式背景就是原形式背景 K。在格的构造算法中，首先生成 $L(K')$，然后再通过添加格节点生成 $L(K)$，由于从 K' 到 K 添加的对象的相容度很低，所以新添加的节点对原有格结构的影响很小，这样显然比直接生成 $L(K)$ 的时间复杂度低。

5 总结与分析

利用 OMCP 算法分割形式背景，改变了原有形式背景和概念格一一对应的模式，结合用户需求，使格的生成不仅仅依赖于形式背景的规模，同时还依赖于背景中各对象间的相容度。该算法不但减小了格的规模，而且使得生成的每个格节点都与用户需求相关，从而有效地提高了概念格的有用性，使得生成的格结构更易于应用于实际问题。例如将 OMCP 算法应用于专业搜索引擎，可解决文档分类困难的问题。

为了验证 OMCP 算法的有效性，在 Windows XP 下用 C# 语言实现了 OMCP 算法，在 P4 1.7G 计算机上对随机产生的数据进行了测试。结果表明，当背景中各对象的相容度有较大差别时，算法表现出良好的效果；当对象相容度的差别较小时，算法效果不够理想。

实际应用中，如果对象相容度差别较小，可以将 OMCP 算法与形式背景的横向或纵向拆分算法[4]相结合，从而进一步缩小背景的规模，使其更有利于实现概念格的并行生成。

参考文献：

[1] 李立峰，王国俊. 一种求概念格属性约简的方法[J]. 计算机工程与应用，2006，42(20)：147－149.

[2] 李 云，刘宗田，吴 强，等. 概念格的分布处理研究[J]. 小型微型计算机系统，2005，26(3)：448－451.

[3] 张 磊，沈夏炯，贾培燕，等. 基于同类概念的概念格横向合并算法[J]. 计算机应用，2006，26(8)：1900－1903.

[4] Ganter B，Wille R. Formal Concept Analysis：Mathematical Foundations[M]. Berlin：Springer，1999.

[5] Ho T B. Discovering and Using Knowledge from Unsupervised Data[J]. Decision Support Systems，1997，21(1)：27－41.

(上接第 26 页)

兵”，造成了模型和代码的不同步，在代码不断被修改的同时，模型不会被更新，这样模型就失去了意义。弥补建模和开发之间的鸿沟的关键就在于将建模变为开发的一个必不可少的部分。基于 MDA 的软件开发模式彻底解决了上述问题，并提出了一种创建系统的新途径。

4 结 论

MDA 的出现，为提高软件开发效率，增强软件的可移植性、协同工作能力和可维护性，以及文档编制的便利性提出了新的解决方案。MDA 被面向对象技术界预言为未来几年内最重要的方法学。

当然，软件开发从来都是一个需要创造性思维的工作，软件技术的进步也从来都是渐进的，都是在已有的技术上进行一个新层次的提升，从来都没有什么“银弹”，MDA 也不例外。面对这种思维上的冲击，既要睿智地去迎接新鲜事物，又要冷静地在现有基础上进行提升，这才是一个理智的开发团队本应该具有的素质。

参考文献：

[1] OMG. Model Driven Architecture White Paper[EB/OL]. 2005. http://www.omg.org/mda.

[2] Kleppe A，Warmer J，Bast W. 解析 MDA[M]. 北京：人民邮电出版社，2004.

[3] Frankel D S. 应用 MDA[M]. 北京：人民邮电出版社，2003.

[4] Sawin 软件研发之窗[EB/OL]. 2006. http://www.sawin.cn/doc/SoftMethod/MDA/blueski377.htm.

[5] UML 软件工程组织[EB/OL]. 2006. http://www.uml.org.cn/UMLSearch/1217001.htm.