

有限集上构建拓扑的集簇式递推算法

赵婷婷, 甘健侯, 梁 立

(云南师范大学 信息学院(民族教育信息化教育部重点实验室), 云南 昆明 650500)

摘 要: 为了解决有限集上的拓扑计数问题, 对拓扑的结构性质进行了分析研究; 针对穷举法中庞大的计算量, 提出了一种集簇式递推算法, 以 $n-1$ 元集合的拓扑为基础, 以扩展子集为线索来构建 n 元集合的拓扑; 与传统的穷举法相比, 大大降低了时间复杂度, 减少了计算时间, 并通过实验验证了算法的有效性。

关键词: 有限集; 扩展子集; 拓扑; 递推

中图分类号: TP301.6

文献标识码: A

文章编号: 1000-7180(2014)02-0110-04

Recursive Algorithm to Construct Topologies on a Finite Set

ZHAO Ting-ting¹, GAN Jian-hou², LIANG Li¹

(College of Information Science and Technology, Yunnan Normal University, Key Laboratory of Educational Informalization for Nationalities, Ministry of Education, Kunming 650500, China)

Abstract: This study focuses on the solution to the topologies counting problem on a finite set from the perspective of topological structure properties. Based on the topologies on a set of $n-1$ elements, a recursive algorithm which constructs the topologies on a set of n elements is put forward following the extended subset. Compared with the traditional method, the algorithm expedites and simplifies the computing. Most importantly, its effectiveness has been verified through experiments.

Key words: finete set; extended subset; topology; recursion

1 引言

目前, 三角网格被广泛用来表示三维物体, 其中拓扑数据描述顶点之间的连接关系, 在进行数据压缩时, 拓扑压缩是其中一个组成部分, 文献[1]对均匀的三角网格得到了理想的压缩效果; 文献[2]通过自适应提高预测准确率, 实现了拓扑信息的高效压缩; 在层次网络中, 拓扑压缩的作用是用尽可能少的空间存储路由计算使用的拓扑信息, 有效的算法可以参考文献[3]; 文献[4]将遗传算法应用于求解网络拓扑的多目标优化模型来解决拓扑优化问题; 文献[5]论述了拓扑结构对网络稳定性的影响。

拓扑计数一直是数学家们关心的的问题。 n 元集合上子集的个数为 2^n , 集簇的数量为 2^{2^n} , 由于计算量大, 起初人们主要研究三点集和四点集^[6]上的

计数问题。后来随着计算机的出现, 使得该问题的深入研究成为可能。文献[7]利用偏序结构和拓扑结构之间的等价性, 分析了 n 元有限集上互不同胚拓扑的计数问题, 提出了一种新思路。

2 基本概念

本文所指的集合都是有限集合。记集合 $X_n = \{x_1, x_2, \dots, x_n\}$ 。

(1) 集簇: 以集合 X 的子集为元素的集合称为 X 的集簇^[8]。例如: $X = \{x_1, x_2, x_3\}$, 则下面两个都是 X 的集簇: $\{\emptyset, \{x_1\}, \{x_1, x_2\}\}$, $\{\{x_2\}, \{x_1, x_2, x_3\}\}$ 。

(2) 拓扑: 如果 X 上的集簇 T 满足下面两个条件, 则 T 为 X 上的拓扑^[10]。

① $\emptyset, X \in T$;

收稿日期: 2013-04-01; **修回日期:** 2013-05-20

基金项目: 国家自然科学基金项目(61262071); 云南省应用基础研究计划面上项目(2011FB044)

② $\forall A, B \in T$, 均有 $A \cap B, A \cup B \in T$

拓扑的集簇必须包含空集和全集, 因此不妨假设每个集簇隐含了空集和全集, 这样的集簇叫做可行集簇.

3 递推算法

采用递推算法可以减少穷举的数量, 在相对小的范围内筛选拓扑.

子集是拓扑的构成元素, 从子集的角度来研究拓扑的构建是最基本也是最直接的方法.

用一个整数表示一个集合, 其二进制位为 1 表示对应元素的存在. 例如: 3 个元素的集合, $\{a, b, c\}$ 用 $7(111_2)$ 表示; $\{b, c\}$ 用 $6(110_2)$ 表示; $\{a, c\}$ 用 $5(101_2)$ 表示; $\{c\}$ 用 4 表示; $\{a, b\}$ 用 3 表示; $\{b\}$ 用 2 表示; $\{a\}$ 用 1 表示; 空集用 0 表示.

因此, n 元集合的子集可用 $0 \sim 2^n - 1$ 的整数来表示.

3.1 可行集簇间的递推

X_n 上的可行集簇隐含了空集 (0) 和全集 ($2^n - 1$) 两个子集, 由 $2^n - 2$ 个子集构成, 分别是 $1 \sim 2^{n-1} - 1, 2^{n-1} \sim 2^n - 2$, 显然, 其中 $1 \sim 2^{n-1} - 1$ 是 X_{n-1} 子集, $2^{n-1} \sim 2^n - 2$ 是与新元素 x_n 相关的子集. 将 X_n 上的可行集簇分为两个部分:

(1) 不包含与新元素 x_n 相关的子集

① 可行集簇如果包含子集 X_{n-1} , 就在 X_{n-1} 上的可行集簇中加入 X_n .

② 可行集簇如果不包含子集 X_{n-1} , 就在 X_{n-1} 上的可行集簇中用 X_n 替代 X_{n-1} .

(2) 包含与新元素 x_n 相关的子集

在 (1) 中所得可行集簇中添加子集 $2^{n-1} \sim 2^n - 2$ 的组合.

因此, 生成 X_n 上的可行集簇时, 分为以下三步, 以 $X_3 = \{x_1, x_2, x_3\}$ 为例:

步骤 1: 在 X_{n-1} 的可行集簇中加入 X_n .

例如: 在 $\{\emptyset, \{x_1\}, \{x_1, x_2\}\}$ 中加入 $\{x_1, x_2, x_3\}$ 得到 $\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$.

步骤 2: 在 X_{n-1} 的可行集簇中用 X_n 替代 X_{n-1} .

例如: 在 $\{\emptyset, \{x_1\}, \{x_1, x_2\}\}$ 中用 $\{x_1, x_2, x_3\}$ 替代 $\{x_1, x_2\}$ 得到 $\{\emptyset, \{x_1\}, \{x_1, x_2, x_3\}\}$.

步骤 3: 在步骤 1 和步骤 2 所得可行集簇中添加子集 $2^{n-1} \sim 2^n - 2$ 的任意组合.

例如: 在 $\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$ 中加入 $\{x_3\}$ 得到 $\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_2, x_3\}\}$; 加入 $\{x_3\}, \{x_1, x_3\}$ 得到 $\{\emptyset, \{x_1\}, \{x_1, x_2\},$

$\{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$; 加入 $\{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}$ 得到 $\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$.

步骤 1 ~ 3 由 X_{n-1} 的可行集簇生成了 X_n 上的可行集簇.

引理 1 X_n 上的可行集簇可由 X_{n-1} 上的可行集簇递推而来.

有了可行集簇之间的递推关系, 经过进一步的论证得到拓扑之间的递推关系.

3.2 拓扑间的递推

定理 1 X_n 上的拓扑可由 X_{n-1} 上的拓扑递推而来.

证明 由引理 1 可知 X_n 上的可行集簇可由 X_{n-1} 上的可行集簇递推而来. X_{n-1} 上的可行集簇分为两部分: 是拓扑的可行集簇和不是拓扑的可行集簇. 只要证明 X_{n-1} 上不是拓扑的可行集簇, 经过步骤 1 ~ 3 递推后得到的集簇仍然不是拓扑, 也就证明了 X_n 上的拓扑可由 X_{n-1} 上的拓扑递推而来.

设 J 是 X_{n-1} 上任一不是拓扑的可行集簇, 则必有 $\exists A, B \in J$, 使得 $A \cup B \notin J$ 或者 $A \cap B \notin J$.

令 $X_n - X_{n-1} = \{x_n\}$, 对于步骤 1 ~ 3 中产生的任一新子集 C , 均有 $x_n \in C$. 由于 A, B 均为 X_{n-1} 的子集, 从而有 $x_n \notin A \cup B$ 且 $x_n \notin A \cap B$. 因此可得 $C \neq A \cup B$ 而且 $C \neq A \cap B$.

由 C 的任意性知, 经过步骤 1 ~ 3 递推后 $A \cup B$ 或者 $A \cap B$ 仍然不在集簇中. 即递推后得到的集簇仍然不是拓扑.

3.3 算法描述

由 X_{n-1} 的拓扑生成 X_n 上的拓扑步骤如下:

第 1 步: 求 X_n 上的拓扑基.

拓扑基由两部分构成: 一部分是在原来的拓扑中加入 X_n , 一定都是拓扑, 称为直系拓扑; 另一部分是在原来的拓扑中用 X_n 替代 X_{n-1} , 如果是拓扑称为置换拓扑.

第 2 步: 扩展子集.

求与新元素 x_n 相关的子集 $2^{n-1} \sim 2^n - 2$, 组合后得到扩展子集簇, 满足封闭性的扩展子集簇称为有效集簇.

封闭性: 对于 X 上的集簇 J , $\forall A, B \in J$, 如果满足以下条件:

① $A \cap B \in J$

② $A \cup B \in J$ 或者 $A \cup B = X$

称集簇 J 满足封闭性.

第 3 步: 扩展集簇. 在拓扑基的每个拓扑中添加

有效集簇,然后判断它们是否是拓扑.

4 算法实验

举例说明由 $X_2 = \{x_1, x_2\}$ 的拓扑递推得到 $X_3 = \{x_1, x_2, x_3\}$ 的拓扑.

(1) 生成拓扑基

① 直系拓扑. 在 X_{n-1} 的拓扑中加入 X_n , 全部都是拓扑, 如表 1 所示.

表 1 直系拓扑

原拓扑	加入 $\{x_1, x_2, x_3\}$	是否为拓扑
$\{\emptyset, \{x_1\}, \{x_1, x_2\}\}$	$\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1, x_2\}\}$	$\{\emptyset, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$	$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_2\}, \{x_1, x_2\}\}$	$\{\emptyset, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	是

② 置换拓扑. 在 X_{n-1} 的拓扑中用 X_n 替代 X_{n-1}

表 3 扩展集簇 1

拓扑基	加入 $\{x_3\}$	是否为拓扑
$\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1, x_2\}, \{x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_2\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_2\}, \{x_3\}, \{x_1, x_2, x_3\}\}$	否

表 4 扩展集簇 2

拓扑基	加入 $\{x_3\}, \{x_1, x_3\}$	是否为拓扑
$\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_2\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$	否

最后得到 X_3 上的拓扑共有 29 个(其中拓扑基 7 个).

(4) 结果分析

实验过程中发现,判断一个集簇是否是拓扑所

得到的拓扑,如表 2 所示.

表 2 置换拓扑

原拓扑	用 $\{x_1, x_2, x_3\}$ 替代 $\{x_1, x_2\}$	是否为拓扑
$\{\emptyset, \{x_1\}, \{x_1, x_2\}\}$	$\{\emptyset, \{x_1\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1, x_2\}\}$	$\{\emptyset, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$	$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_2\}, \{x_1, x_2\}\}$	$\{\emptyset, \{x_2\}, \{x_1, x_2, x_3\}\}$	是

因此,拓扑基有 7 个拓扑.

(2) 扩展子集

与新元素 x_n 相关的子集有 3 个: $\{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}$. 扩展集簇共有 $2^3 - 1 = 7$ 个,有效集簇有 6 个,其中集簇 $\{\{x_1, x_3\}, \{x_2, x_3\}\}$ 不满足封闭性.

(3) 扩展集簇

在拓扑基的每个拓扑中分别添加有效集簇,然后判断它们是否是拓扑(表 3 ~ 表 5 举例说明).

需的时间与扩展集簇所需的时间相比,前者几乎可以忽略不计.

穷举法对所有的可行集簇逐一检查是否为拓扑;递推算法对所有的扩展子集簇逐一检查是否为

有效集簇,两者的计算量对比如表6所示.

表5 扩展集簇3

拓扑基	加入 $\{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}$	是否为拓扑
$\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_2\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_2\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$	否
$\{\emptyset, \{x_1\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_1\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$	是
$\{\emptyset, \{x_2\}, \{x_1, x_2, x_3\}\}$	$\{\emptyset, \{x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\}$	是

表6 计算量比较

元素个数 n	穷举法	递推算法
3	64	56
4	16 384	5 760
5	1 073 741 824	16 384 000
6	4 611 686 018 427 387 904	19 441 169 465 344

表6中的数据说明,随着元素个数的不断增加,与穷举法相比,递推算法大大减少了扩展集簇的数量,从而减少了计算时间.实验表明该递推算法是有效的.

5 结束语

本文主要做了两方面的工作:

(1)从算法的角度分析拓扑构建问题.在构建拓扑时,该算法没有重复和遗漏,避免了繁重的查重和查漏工作,而且易于编程实现.

(2)用递推的思想解决构建拓扑问题.本文从分析拓扑的结构特征出发,研究有限集上的拓扑构建问题,以 X_{n-1} 的拓扑为基础得到 X_n 上的拓扑,避免了大规模的穷举,从而降低了计算时间.

本文的算法还有一些需要改进的地方,比如减少扩展子集簇的无效运算.另外,递推的步骤可以进一步细化,以便更好地对拓扑的结构性质进行研究和应用.

参考文献:

- [1] 伏鹏.三角网格压缩算法[D].杭州:浙江大学,2009.
- [2] 刘迎,刘学慧,吴恩华.基于可变模板的三角网格拓扑压缩[J].软件学报,2008,19(4):1016-1025.
- [3] 杨敏,向勇,史美林,等.层次网络中的拓扑压缩算法及性能比较[J].计算机科学,2002,29(12):44-48.
- [4] 张亮,黄曙光,赵文进.基于遗传算法的网络最优拓扑研究[J].微电子学与计算机,2009,26(9):64-66.
- [5] 毛凯.基于拓扑结构的计算机网络稳定性分析[J].微电子学与计算机,2012,29(12):72-74.
- [6] Bouhjar Khalid, Dijkstra Jan J, Mill Jan van. Three-point sets[J]. Topology and its Applications, 2001, 112(2):215-227.
- [7] 张震.有限拓扑中的计数问题[D].北京:首都师范大学,2001.
- [8] 熊金城.点集拓扑讲义[M].4版.北京:高等教育出版社,2011.

作者简介:

赵婷婷 女,(1986—),硕士研究生.研究方向为算法设计与分析.

甘健侯 男,(1976—),博士研究生,副教授.研究方向为智能信息处理、数据库技术等.

梁立 男,(1964—),教授.研究方向为算法设计与分析、图论.